
Get Up, Running, and Integrated with Cloud Manager for Experience Manager

Table of Contents

- [Lab Overview](#)
- [Lesson 1 - Pipeline Setup](#)
- [Lesson 2 - Pipeline Execution](#)
- [Lesson 3 - Fixing Issues](#)
- [Lesson 4 - Create a Webhook](#)
- [Lesson 5 - Webhook Setup](#)
- [Lesson 6 - Testing it Out](#)
- [Next Steps](#)
- [Additional Resources](#)

Scenario Overview

Welcome to the scenario "Get Up, Running, and Integrated with Cloud Manager for Experience Manager"! In this scenario, we will look at how Cloud Manager helps our customers and partners quickly deploy their custom applications to Adobe-managed Experience Manager environments while ensuring that custom code adheres to both Adobe and industry best practices. We will also explore how external systems (in this case Microsoft Teams) can be integrated with Cloud Manager using our API.

Cloud Manager, first introduced at Summit 2018, is a self-service cloud application which enables our Managed Services customers to deploy, update, monitor, and manage their Experience Manager environments. This lab will be focused on the Continuous Integration / Continuous Delivery (CI/CD) aspect of Cloud Manager. There are other sessions at Summit which address some of the other capabilities of Cloud Manager as well as full reference documentation linked to from the [Additional Resources](#) section below.

Key Takeaways

- Learn how to set up a CI/CD pipeline in Cloud Manager.
- See how Cloud Manager identifies coding errors.
- Integrate Cloud Manager with an external system using Adobe I/O.

Scenario Roadmap

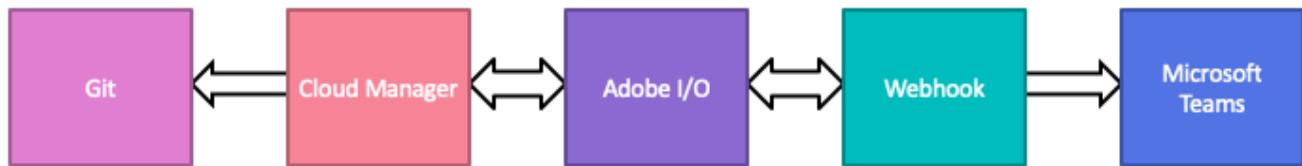
In this scenario, we'll be using a number of different tools, so before we get started, let's do a quick overview of what each of these tools does and how we will be using them.

- Cloud Manager
 - In this scenario, we'll be setting up a CI/CD pipeline in Cloud Manager and executing it a few times. Cloud Manager will be used in Lessons 1, 2, 3, and 6.
- Git
 - Git is a version control system. Every Cloud Manager customer is provided with a git repository. This scenario won't be teaching you the ins and outs of using git; just enough to get by. Git will

be used in Lesson 3.

- Adobe I/O
 - Adobe I/O is Adobe's centralized API Gateway through which customers and partners can integrate with the entire Adobe product portfolio. Adobe I/O will be used directly in Lesson 5.
- Microsoft Teams
 - Microsoft Teams is a collaboration platform which includes chat (both group and one-on-one), voice, video, and other types of communication. In this lab, we are using Microsoft Teams as a notification channel in Lessons 5 and 6.

Lab Touchpoints



© 2019 Adobe. All Rights Reserved. Adobe Confidential.
adobe.com/go/fairuse/2019-f122



The last item in this picture is a webhook. This is a small piece of software which you'll be configuring, running, and (if you want) modifying which will act as a notification broker between Adobe I/O and Microsoft Teams. Your webhook (created in Lesson 4) will receive notifications *from* Adobe I/O and then send notifications *to* Microsoft Teams. In addition to receiving notifications from Adobe I/O, the webhook will make API calls *to* Adobe I/O to augment the information provided in the initial notification.

Prerequisites

As a LiveTrial attendee, you have been provisioned with all of the necessary access and software needed to participate in this scenario. If you are using this workbook outside of the LiveTrial, you'll need the following:

- Access to Cloud Manager with the Deployment Manager role.
- The System Administrator role for your Organization in the Adobe Admin Console.
- A git client (either the command line client or as part of an Integrated Development Environment).

Lesson 1 - Pipeline Setup

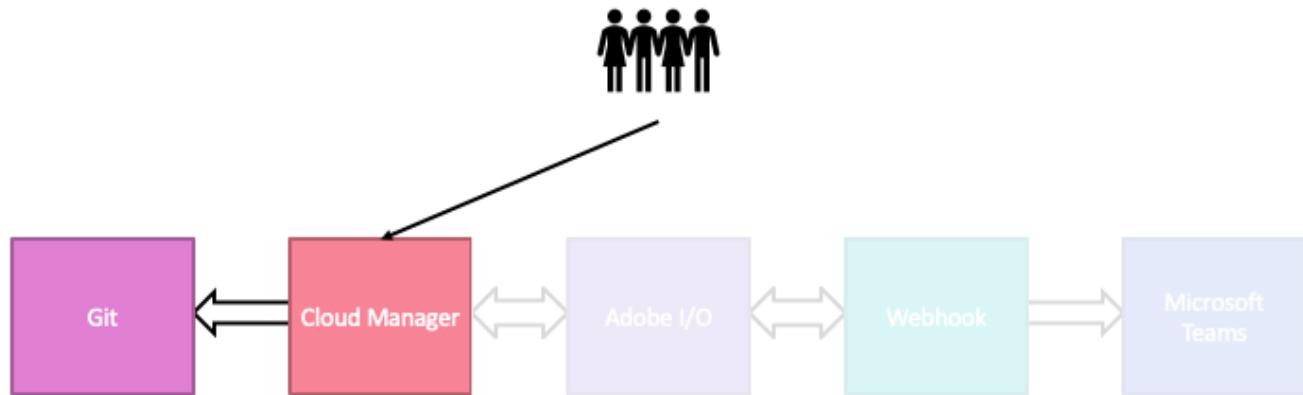
Objectives

1. Log into Cloud Manager
2. Create a Code Quality Pipeline

Lesson Context

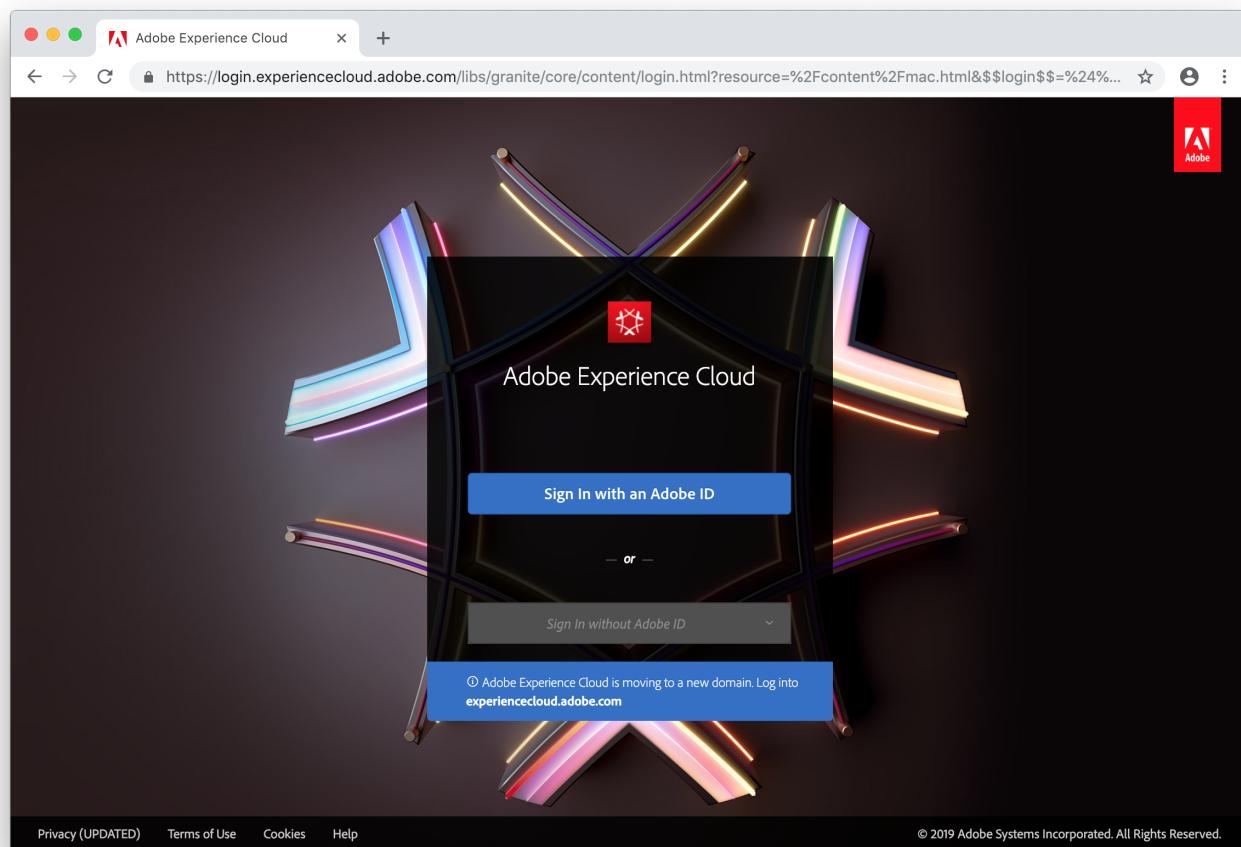
This lesson will start by logging into Cloud Manager through the Experience Cloud. Once logged in, you will create a simple CI/CD pipeline which builds an application project and evaluates the code quality. Pipelines like this can be used throughout the development process to get early and fast feedback.

Lab Touchpoints – Lesson 1



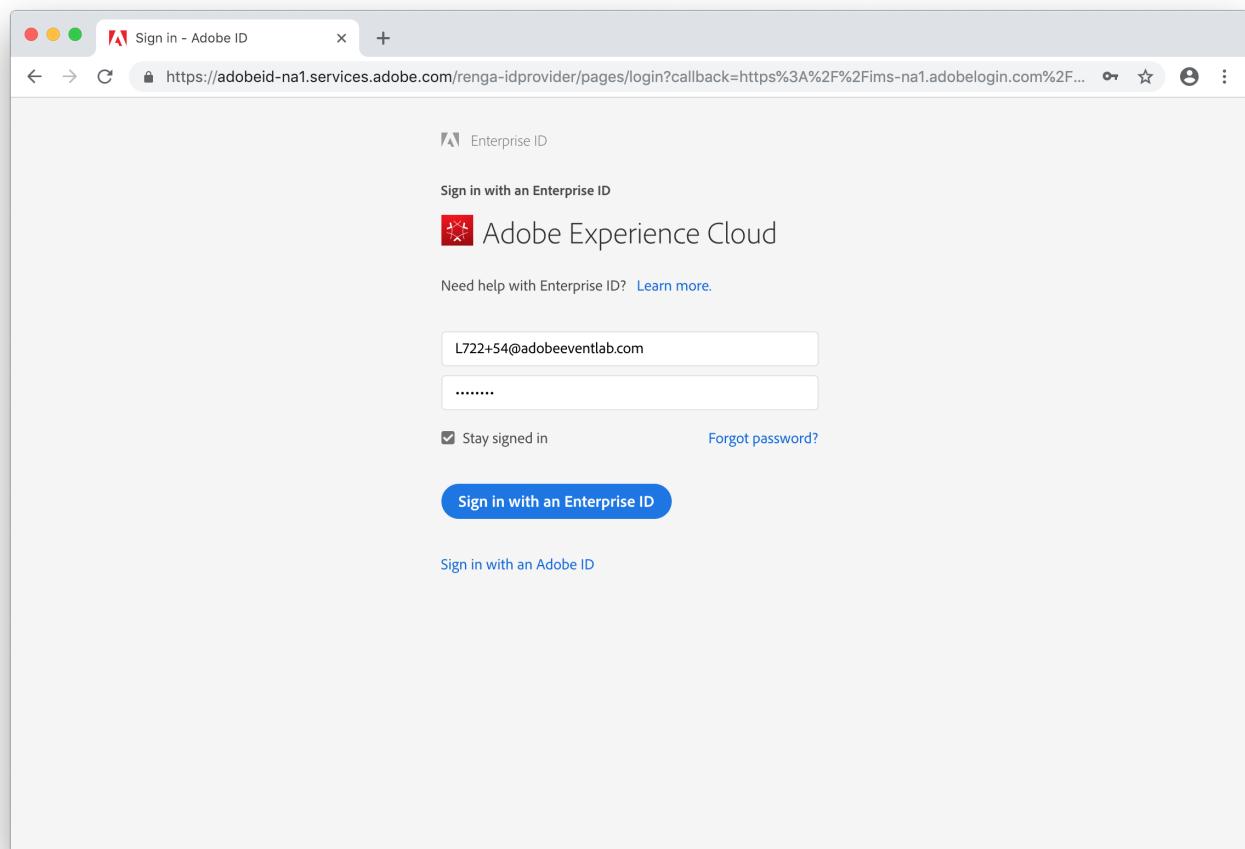
Exercise 1.1

Our lab starts with logging into the Cloud Manager web interface. To start, open Google Chrome. It should automatically present you with the Experience Cloud login page. If not, navigate to <https://aem65lt.experiencecloud.adobe.com>.



On this page, click the *Sign In with an Adobe ID* button.

Each attendee has a unique login assigned to them. You should see this on your workstation. Enter that email address and password on the login form and click the *Sign in with an Enterprise ID* button.



To navigate to Cloud Manager, click the Solution Switcher in the header navigation (the icon is three rows of three squares) and select Experience Manager.

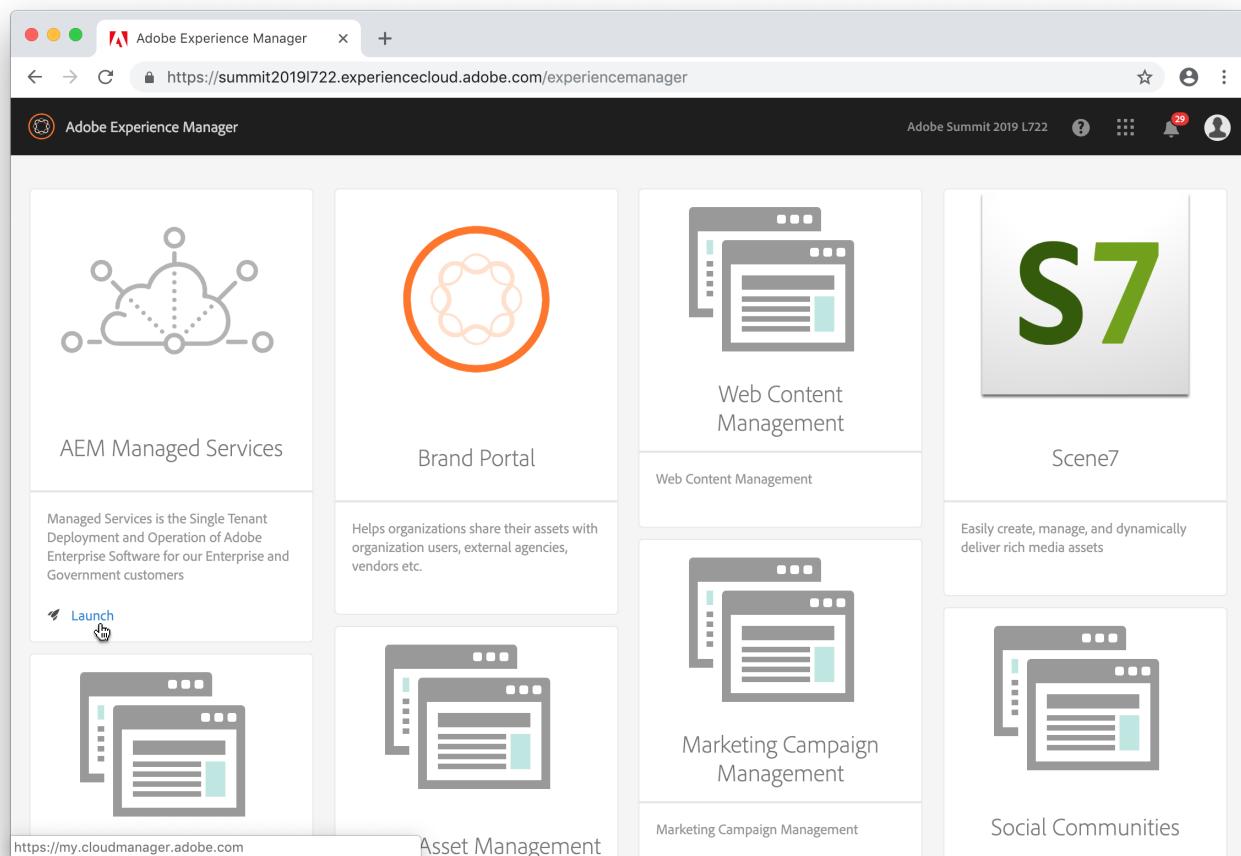
The screenshot shows the Adobe Experience Cloud home page. At the top, there's a navigation bar with links for "Adobe Experience Cloud", "Profile & Password", "Preferences", "Adobe Summit 2019 L722", and user icons. Below the navigation is a grid of service cards:

ADVERTISING CLOUD	MARKETING CLOUD	ANALYTICS CLOUD	PLATFORM
Media Optimizer	Campaign	Analytics	Activation
	Experience Manager	Audience Manager	Assets
	Primetime		Exchange
	Social		Experience Cloud Home
	Target		Feed
			Mobile Services
			Offers
			People

At the bottom, there are two sections of user cards:

ORG ADMIN	EMAIL	ORG ADMIN	EMAIL	LINK
Adobe Summit 2019 L722	L722 01	Adobe Summit 2019 L722	Adobe Experience Manager Home	Launch Experience Manager
ORG ADMIN	L722+01@adobeeventlab.com	ORG ADMIN	L722 02	Launch Activation
	L722+02@adobeeventlab.com			

On this screen, click the *AEM Managed Services* card.



We refer to this screen as the *Program Switcher*. Many of our Managed Services customers have multiple *programs* under a single enterprise organization. For example, you might have one set of environments for an Enterprise DAM, another set for your public-facing websites, and another set for an employee or partner portal. Cloud Manager groups these into separate programs. For the purpose of this lab, we only have a single program (We.Retail Intranet). To navigate to the Cloud Manager overview page for that program, hover over the card and click on the Cloud Manager icon (it will be the first icon on the left).

The screenshot shows the Adobe Experience Cloud interface, specifically the "Managed Services - Programs" section. A single program card is displayed:

- EXPERIENCE MANAGER We.Retail Assets**: Features a server icon and a "Launch" button.

At the bottom of the page, there are links for "Help", "Terms of Use", "Privacy Policy", "Language: English", and "© 2019 Adobe. All Rights Reserved".

If you click on the card itself, you will get an error message stating that the AEM Link cannot be found. Under normal circumstances, clicking on the card will navigate to the production AEM author instance. For the purpose of this lab, there is no production author instance, so this link is non-functional.

Exercise 1.2

Cloud Manager supports multiple types of CI/CD pipelines. The *primary* pipeline builds a customer application project, runs a series of code quality checks, deploys that project to a staging environment, runs security and performance tests, and finally deploys to the production environment. Additional pipelines may be created to deploy to other non-production environments, e.g. a development environment. There is also support for what we refer to as "Code Quality Only" pipelines which only execute the build and code quality checks. These pipelines are useful to verify code quality during a development cycle and are especially relevant for customers who do not have a separate development environment hosted by Adobe.

For the purpose of this lab, we will be using a Code Quality Only pipeline primarily for the purpose of speed -- it does the least and thus takes the least amount of time.

That said, if you are using this workbook outside of Summit and have the environments available, feel free to use a different pipeline type; it doesn't actually impact the lab content significantly.

To start, click the Add button in the Non-Production pipelines card.

The screenshot shows a card titled "Non-Production Pipelines" with the sub-instruction "Manage pipelines which deploy to non-production environments". Below the title, there is a section labeled "PIPELINE" containing a single item: "Code Quality on Master" under the "master" branch. At the bottom of the card, there is a dark button with the word "Add" and a hand cursor icon, and a "Learn More" link.

Provide a name for your pipeline which includes your attendee number. For example, if your attendee number is 5, name it *Pipeline 5*. Make sure that *Code Quality Pipeline* is the selected Pipeline Type (it should be the only option available).

The git repository for this program has been set up with 100 branches, one for each attendee. This is not really typical -- most programs only have a handful of branches. Select the branch which corresponds to your attendee number.

The screenshot shows a web browser window titled 'Add Non-Production Pipeline'. The URL is <https://my.cloudmanager.adobe.com/nppipelinesetup.html/organization/C80E31365C1030E80A495CFC@AdobeOrg/program/1>. The page has a light gray header with the title and a back/forward button. Below the header is a navigation bar with a lock icon, a search bar, and user profile icons.

Pipeline Name
Provide a name for the Pipeline
Pipeline 54

Pipeline Type
The Pipeline can be a Code Quality type or a Deployment type

Code Quality Pipeline
A Pipeline that handles builds, runs unit tests and evaluates Code Quality

Deployment Pipeline
A Pipeline that handles builds, runs unit tests, evaluates Code Quality and deploys to an Environment

Git Branch
The pipeline must be configured with a Git branch.
attendee-054

Cancel Save

Finally, make sure that the Manual trigger is selected under the Pipeline Options section and that the Important Failure Behavior is set to Fail immediately.

The screenshot shows a web browser window titled "Add Non-Production Pipeline". The URL is <https://my.cloudmanager.adobe.com/nppipelinesetup.html/organization/C80E31365C1030E80A495CFC@AdobeOrg/program/1>. The page displays two pipeline types: "Code Quality Pipeline" (selected) and "Deployment Pipeline". Under "Code Quality Pipeline", it says "A Pipeline that handles builds, runs unit tests and evaluates Code Quality". Under "Deployment Pipeline", it says "A Pipeline that handles builds, runs unit tests, evaluates Code Quality and deploys to an Environment". Below this, there is a section titled "Git Branch" with the sub-instruction "The pipeline must be configured with a Git branch." A dropdown menu shows "attendee-054". The next section is "Pipeline Options" with two columns: "Deployment Trigger" (with "Manual" selected) and "Important Failure Behavior" (with "Fail immediately" selected). At the bottom of the page are links for "Help", "Terms of Use", "Privacy Policy", and "Language: English", along with a copyright notice: "© 2018 Adobe. All Rights Reserved."

Finally, click the Save button in the top right corner.

Lesson 2 - Pipeline Execution

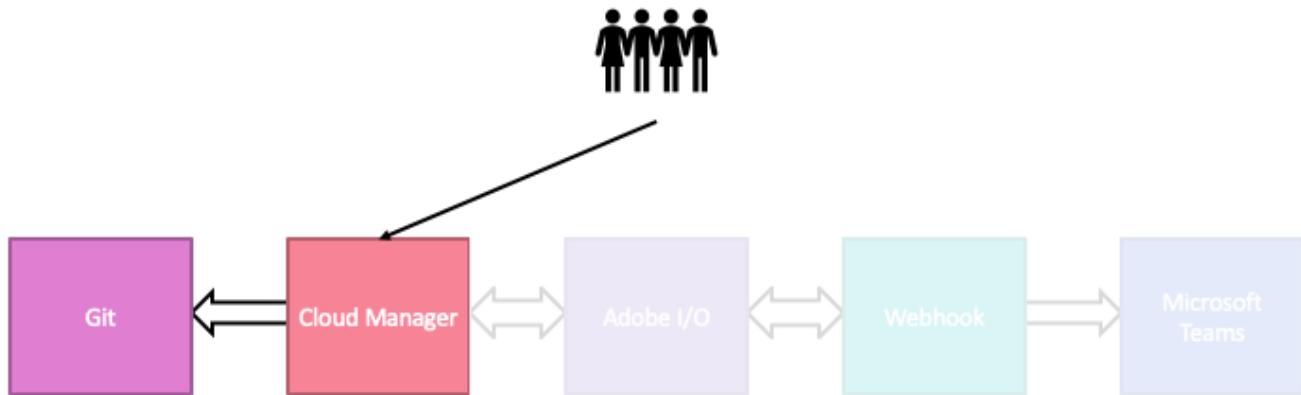
Objectives

1. Execute the Pipeline
2. Understand the Results

Lesson Context

In this lesson, we will execute the pipeline created in the first lesson and observe the results in both summarized and detailed form.

Lab Touchpoints – Lesson 2



Exercise 2.1

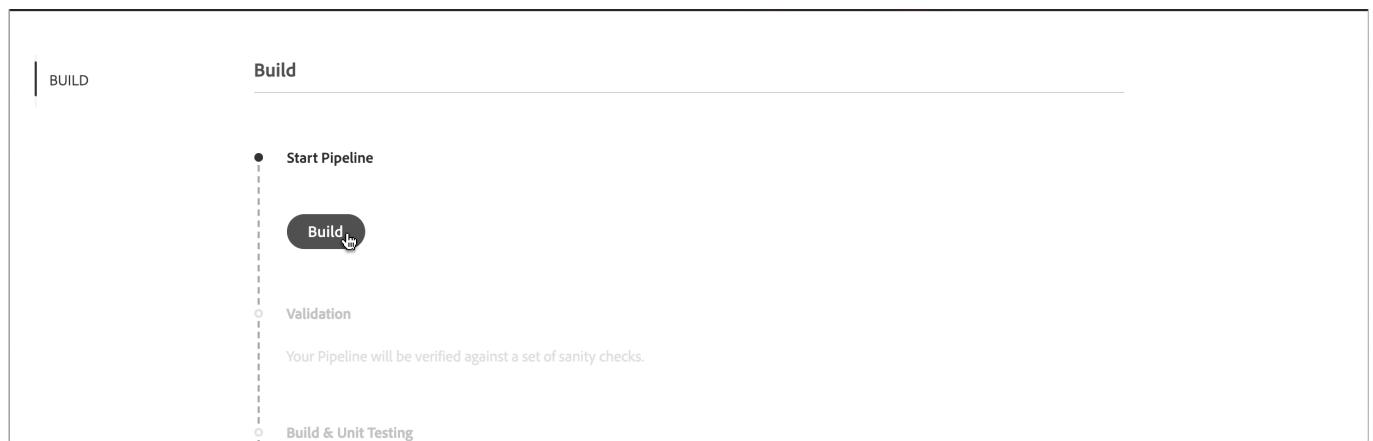
Now that we have our pipeline setup, it is time to execute it.

You should be back on the Cloud Manager Overview page. If not, click the Overview link in the top navigation.

In the Non-Production Pipelines box, find the pipeline you created in the first step. Hover over the row in the table and click the *Build* button. This will take you to the Pipeline Execution page.



On the Pipeline Execution page, click the *Build* button to start the pipeline.



This will take a little time, so grab something to drink.

Exercise 2.2

Uh oh. The pipeline failed. Let's see why. Looking at the execution details screen, you'll see that the Code Quality step has failed. Click on the Review Summary button to open the summary dialog.

The screenshot shows the 'Pipeline Execution' interface in Adobe Experience Manager. At the top, there's a banner with the text 'Pipeline Execution' and a sub-instruction: 'Push your application code into deployment and track progress below. Review results from various deployment steps and validate your application at runtime after successful deployment.' Below the banner, the 'BUILD' section is visible, containing two steps: 'Build & Unit Testing' (status: Succeeded) and 'Code Scanning' (status: Failed). The 'Code Scanning' step includes a note: 'Your application code has been verified against a selected set of rules used for AEM applications.' At the bottom of the BUILD section, there are three buttons: 'Review Summary' (highlighted), 'Download Details' (disabled), and 'Failed' (disabled).

The Code Quality, Security Testing, and Performance Testing steps in Cloud Manager all follow a three-tier structure -- the step produced a variety of metrics and these are classified as either *Critical*, *Important*, or *Informational*. If a Critical metric has not met its threshold, the execution fails immediately. If an Important metric has not met its threshold, the execution is paused (by default, although in this case, we specified the pipeline should fail immediately since it is a Code Quality-only pipeline) until someone overrides or rejects the violation. And Informational metrics are purely Informational.

The Security Test step is slightly different in that each "metric" is actually a separate Health Check and either passes or fails, but the same three-tier concept applies.

Code Scanning Results

Critical ● 0 PASSED ● 1 FAILED

Security Rating is B or better

C ●

Important ● 2 PASSED ● 2 FAILED

Security Rating is A

C ●

Code Coverage is 50% or more

100.0% ●

Maintainability Rating is A

A ●

Reliability Rating is C or better

D ●

Information

Duplicated Lines (%) is 1% or less

0.0 ●

Number of Open Issues is less than 1

3 ●

Close

In this case, we actually have two problems -- the Security Rating is a C and the Reliability Rating is a D. The Reliability Rating is an Important metric, so we could override that, but the Security Rating is Critical so we must fix that in order to proceed. So let's do that. Click the Close button to dismiss the dialog.

Cloud Manager provides a spreadsheet listing the specific coding rule violations which led to the metrics. We'll need that for the next lesson, so click the Download Details button to download it.

The resulting download should open automatically in Microsoft Excel, but if not, go ahead and open it. There will be four rows in the spreadsheet -- a heading row and one row for each of the three violations identified in the project. You may want to adjust the columns for readability.

Lesson 3 - Fixing Issues

Objectives

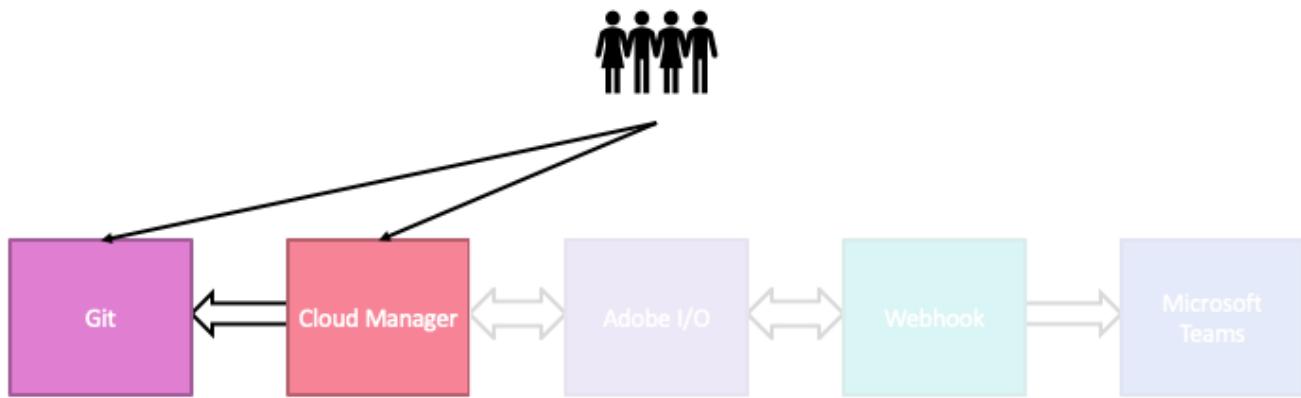
1. Checking out the Project Code
2. Updating the Project Code

3. Re-Executing the Pipeline

Lesson Context

Building on the prior lesson, in this lesson you will download a copy of the codebase built by the Cloud Manager pipeline set up in the first lesson and executed in the second lesson and attempt to resolve the problems identified during the pipeline execution.

Lab Touchpoints – Lesson 3

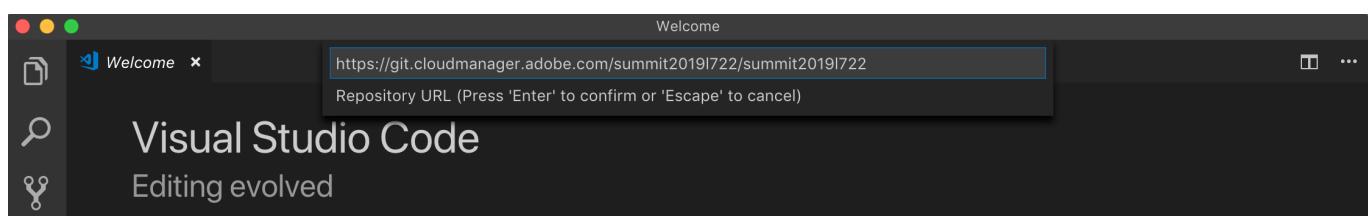


Exercise 3.1

Now that Cloud Manager has told us where the problems in our code lie, let's go ahead and fix them. To do this, you will clone the Cloud Manager-provided git repository for this program and make your changes on your branch.

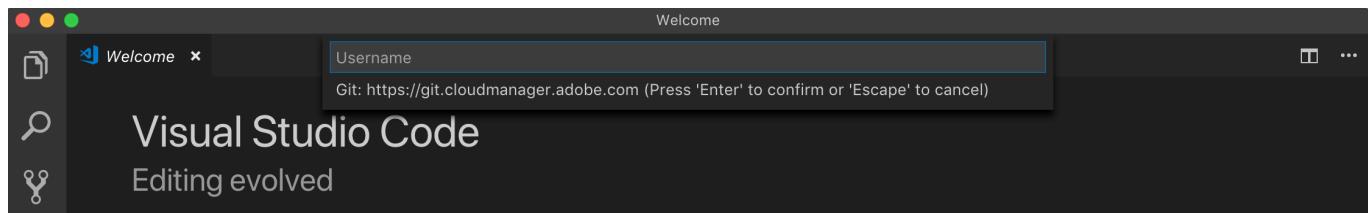
In order to connect to the git repository, you will need a URL, username, and password. These can be found in a file named *git_credentials.txt* on the Desktop. Open this file in TextEdit.

In this lab, we will be using Microsoft Visual Studio Code as our Integrated Development Environment, but if you are doing this lab on your own, you can use any IDE. To start, open Visual Studio Code. Select *Command Palette* from the *View* menu and type *Git: Clone*. Copy and paste the URL from the text file and press *Enter*.

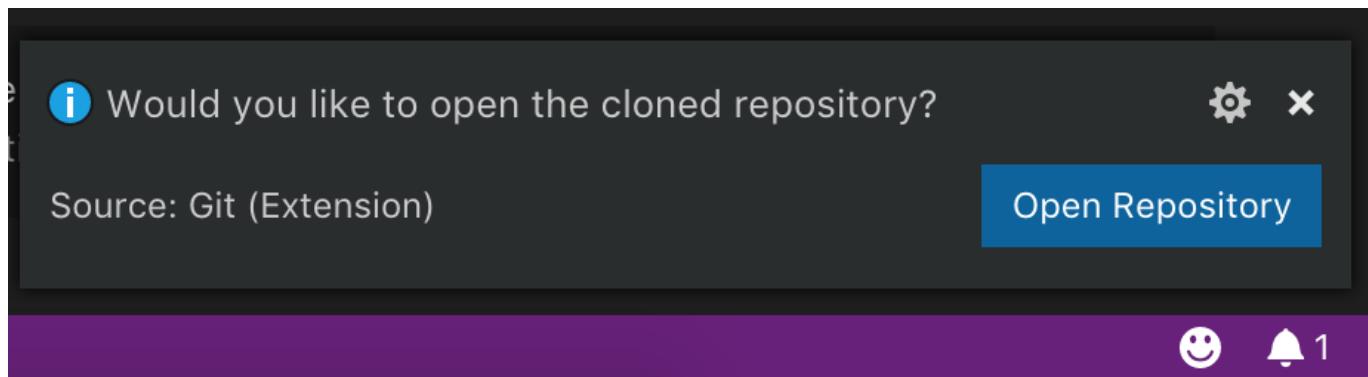


When prompted, select your *Documents* folder and click *Select Repository Location*. Visual Studio Code will automatically create a folder inside the selected folder.

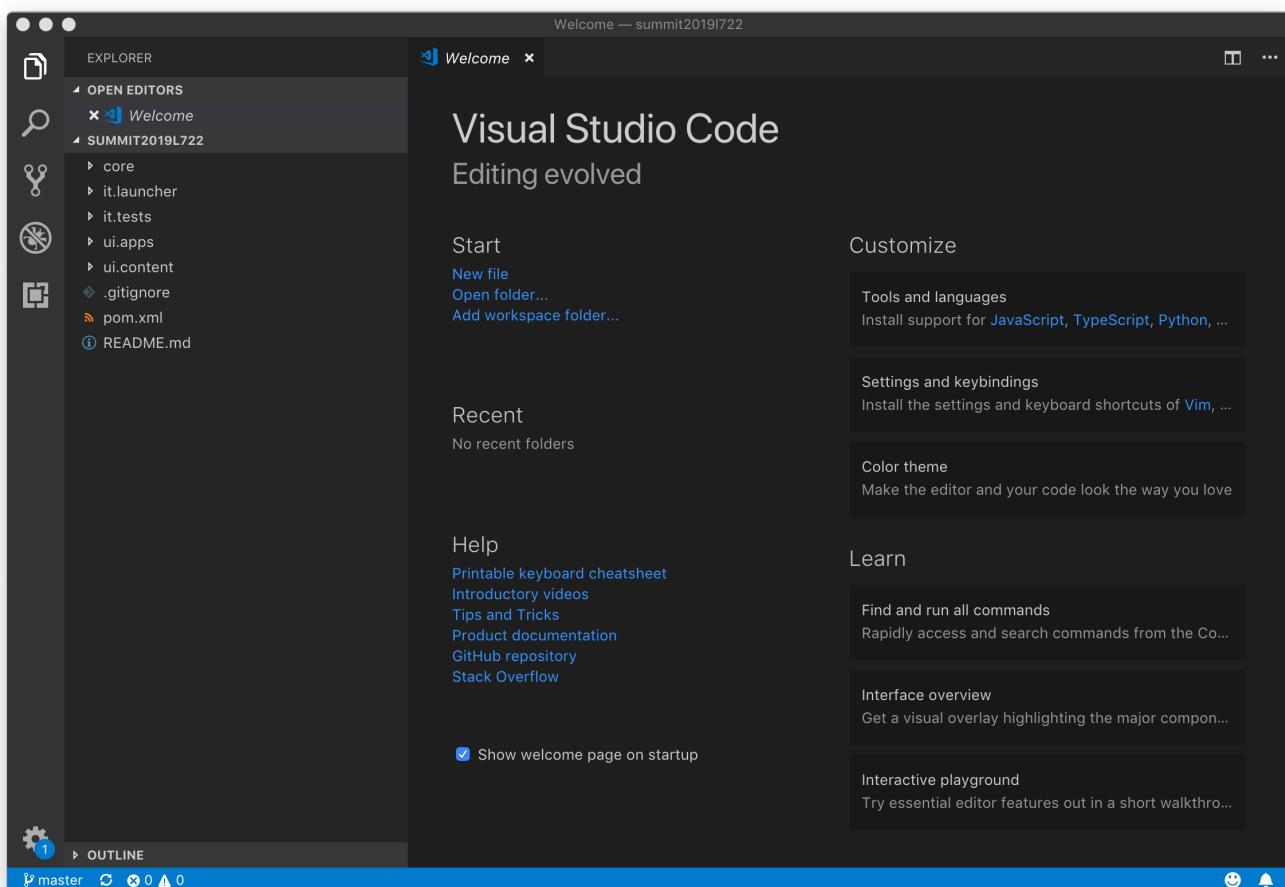
Visual Studio Code will now prompt you for the username and password. As with the URL, copy and paste these values from the text file and press *Enter*.



Once the clone operation is complete, Visual Studio Code will prompt you to open the cloned repository. Click the *Open Repository* button.

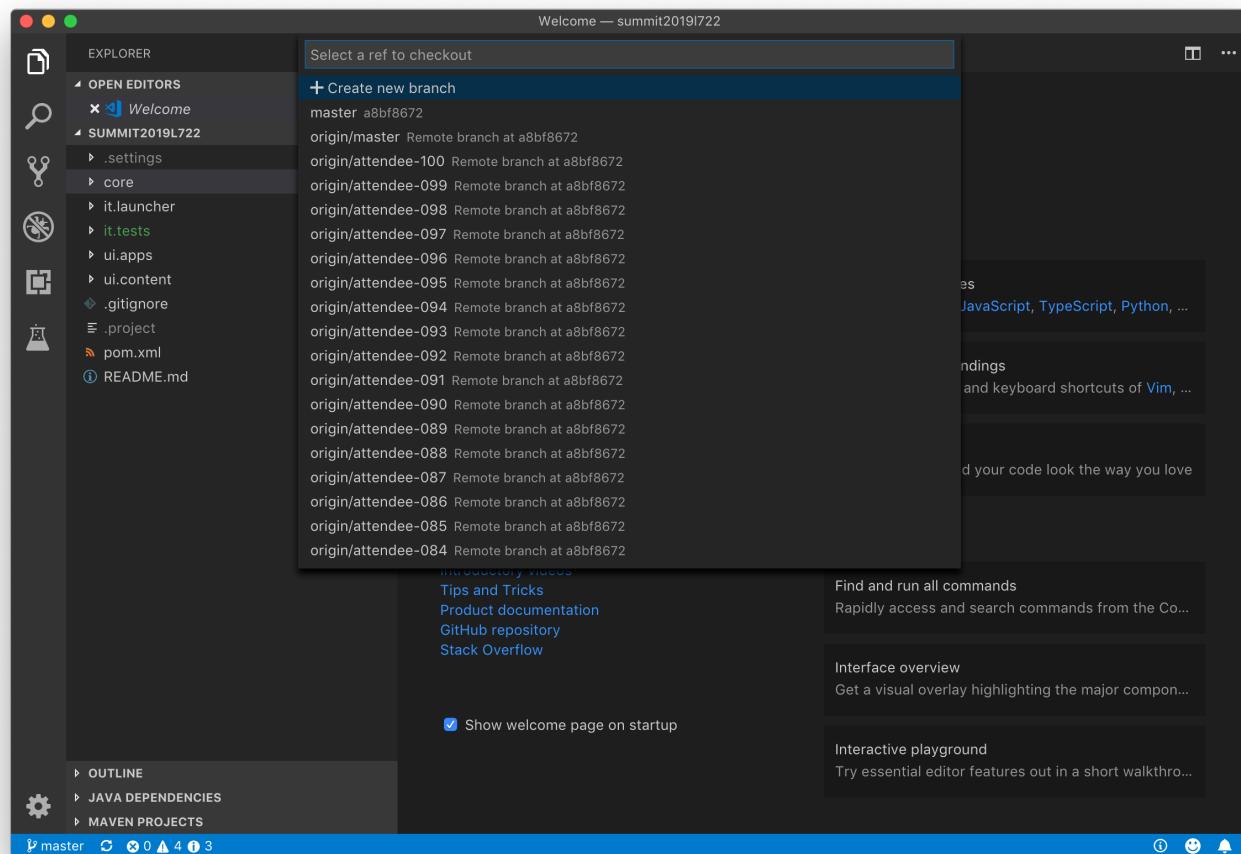


Your Visual Studio Code window should now look like this:



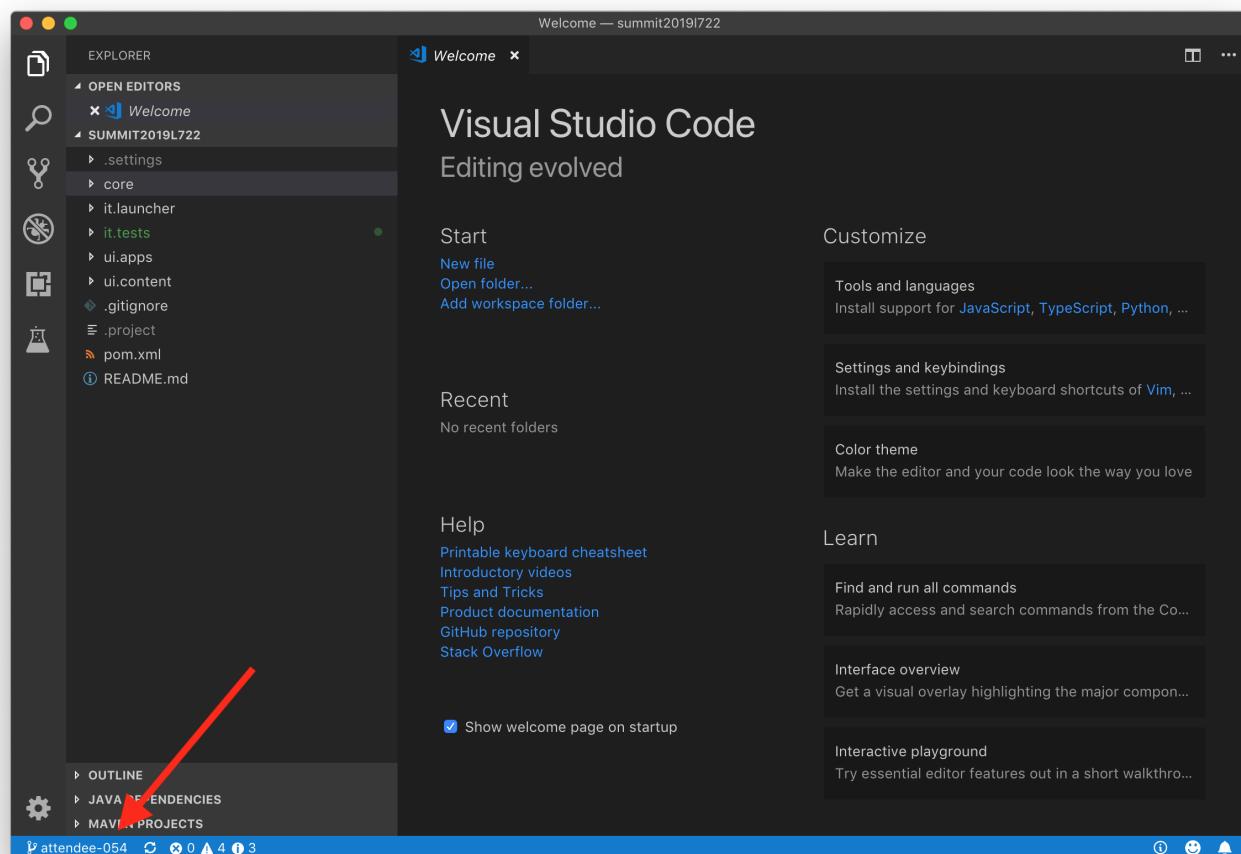
The last thing you need to do before editing code is to checkout your specific branch. As mentioned in [Exercise 1.2](#), each attendee has their own branch. To checkout your branch, open the Command Palette and

select Git: *Checkout to....*



Select your branch and press *Enter*.

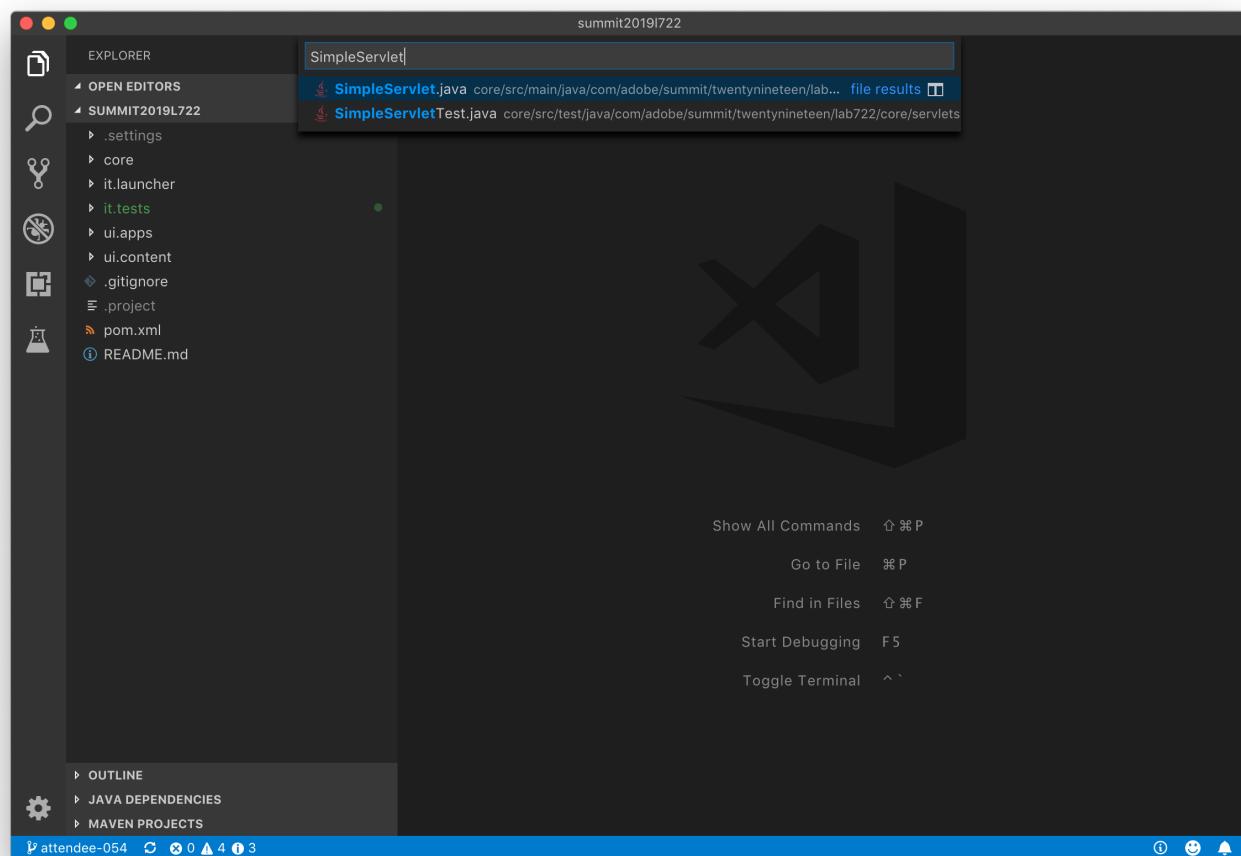
In Visual Studio Code, the current branch is shown in the bottom left-hand corner. Confirm that the proper branch has been selected.



Exercise 3.2

Now that the code is checked out, the two issues identified by Cloud Manager can be fixed. Based on the spreadsheet, you can see that the issues are in the files named *NotFoundResponseStatus.java* and *SimpleServlet.java*.

To easily open these files, open the *Go* menu and select *Go to File....* Then enter the file name.



For both files, the fixes are in the source file, just commented out. Follow the inline instructions as to which lines to add comments to and which lines to uncomment. After making the directed changes, *NotFoundResponseStatus.java* should look like this:

```
* 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 CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.adobe.summit.twentynineteen.lab722.core.servlets;

import com.adobe.cq.sightly.WCMUsePojo;

/**
 * This class is responsible for handling the status for a not found request.
 */
public class NotFoundResponseStatus extends WCMUsePojo {

    @Override
    public void activate() throws Exception {
        getResponse().setStatus(404);
        getResponse().setContentType("text/html");
    }

    public String getResponseText() {
        // uncomment this line and comment out the line below to get rid of the security warning
        return String.format("Could not find %s for request method %s.", getRequest()
            // return String.format("Could not find " + getPathInfo() + " for request method " + getMethod())
    }
}
```

And *SimpleServlet.java* should look like this:

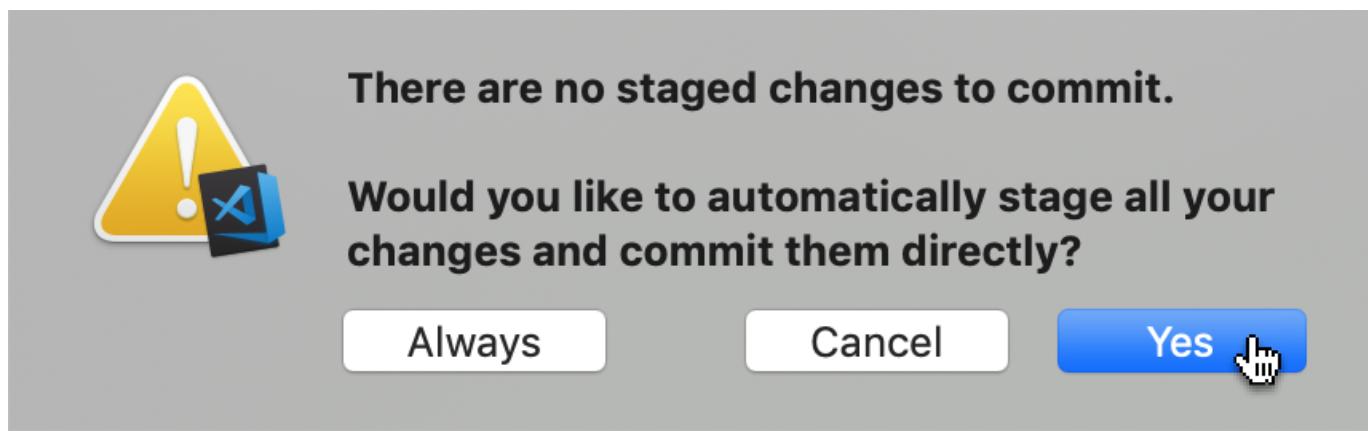
```

SimpleServlet.java — summit2019L722
SimpleServlet.java • NotFoundResponseStatus.java
35 /**
36 * Servlet that writes some sample content into the response. It is mounted for
37 * all resources of a specific Sling resource type. The
38 * {@link SlingSafeMethodsServlet} shall be used for HTTP methods that are
39 * idempotent. For write operations use the {@link SlingAllMethodsServlet}.
40 */
41 @Component(service=Servlet.class,
42             property={
43                 Constants.SERVICE_DESCRIPTION + "=Simple Demo Servlet",
44                 "sling.servlet.methods=" + HttpMethod.GET,
45                 "sling.servlet.resourceTypes=" + "lab-project/components/structure",
46                 "sling.servlet.extensions=" + "txt"
47             })
48 public class SimpleServlet extends SlingSafeMethodsServlet {
49
50     private static final long serialVersionUID = 1L;
51
52     private static final Logger log = LoggerFactory.getLogger(SimpleServlet.class);
53
54     // comment out the line below to get rid of the bug
55     // private transient ResourceResolver resourceResolver;
56
57     @Override
58     protected void doGet(final SlingHttpServletRequest req,
59                          final SlingHttpServletResponse resp) throws ServletException, IOException
60     {
61         // uncomment this line and comment out the line below to get rid of the bug
62         //resourceResolver = req.getResourceResolver();
63         ResourceResolver resourceResolver = req.getResourceResolver();
64         log.info("Request received from {}", resourceResolver.getUserID());
65         final Resource resource = req.getResource();
66         resp.setContentType("text/plain");
67         resp.getWriter().write("Title = " + resource.getValueMap().get(JcrConstants.
68     }

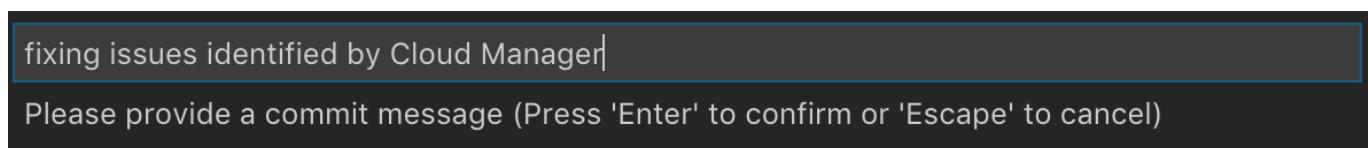
```

Remember to save both files.

These changes now need to be committed to the git repository. Open the Command Palette and select *Git: Commit All*. You'll be prompted to first stage your changes. Click the *Yes* button.



You'll then be prompted for a commit message. Type some useful message and press *Enter*.



Finally, to push this commit to the Cloud Manager git repository, open the Command Palette again and select *Git: Push*.

Exercise 3.3

With the fixes to the issues committed and pushed, the pipeline should successfully execute. To try this, go back to the web browser, navigate to the Cloud Manager Overview page, find your pipeline and build it again.

Lesson 4 - Create a Webhook

Objectives

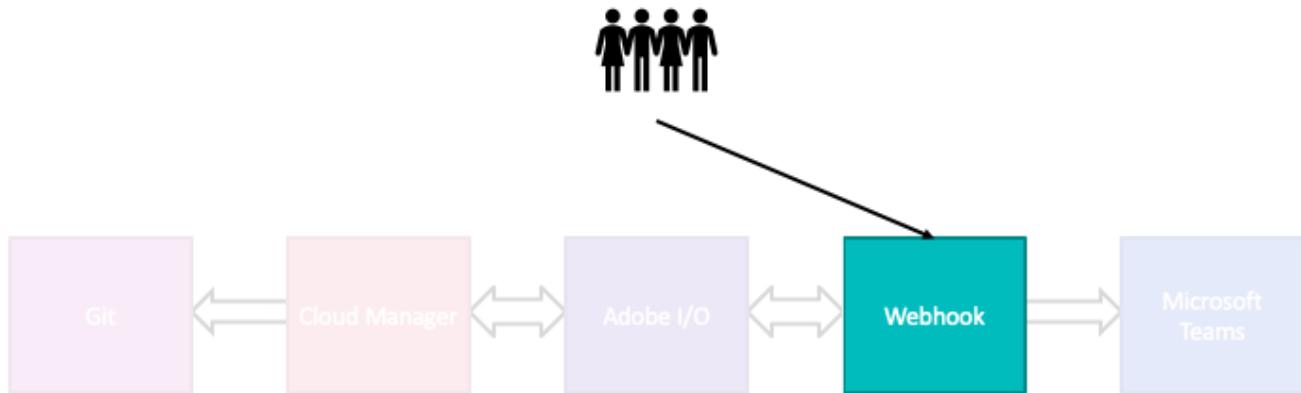
1. Run a Simple Webhook
2. Expose the Webhook using ngrok

Lesson Context

NodeJS is required for this lesson. Please install it if you haven't done so.

In this lesson, you will run a simple web application which illustrates the type of application typically run to receive events from Adobe I/O. You will also use a tool called [ngrok](#) to expose that application to the public internet.

Lab Touchpoints – Lesson 4



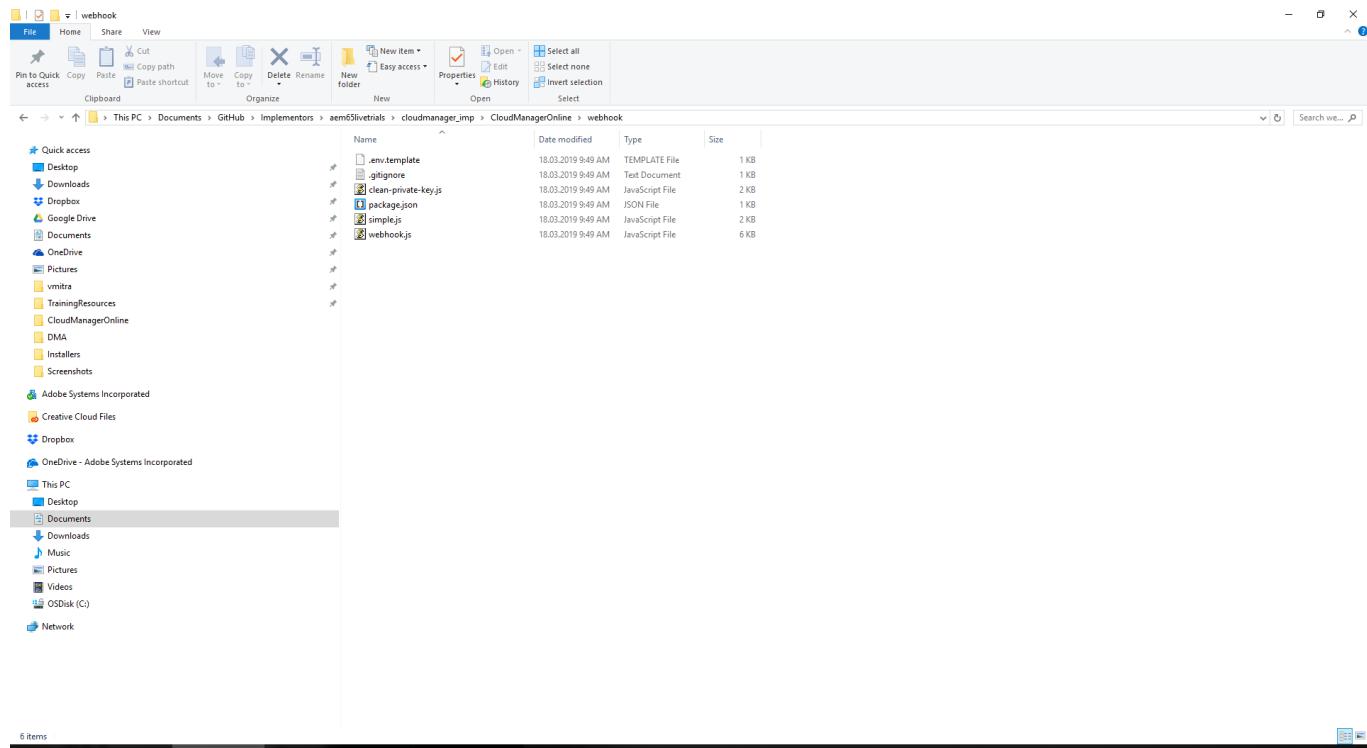
Exercise 4.1

The next two lessons are focused on how to use the Cloud Manager API and Adobe I/O to receive notifications from Cloud Manager when the pipeline starts. To do this, we will create a *webhook*. A webhook is simply a small web application which receives a request from a service when something has happened. In this case, Adobe I/O will invoke the webhook on any pipeline event -- when the pipeline starts, when it ends, when individual steps start and end, etc.

The bulk of the webhook has already been written and in the next lesson we will be configuring the webhook, but before we get to the real webhook implementation, let's start by running a very simple webhook.

To start this:

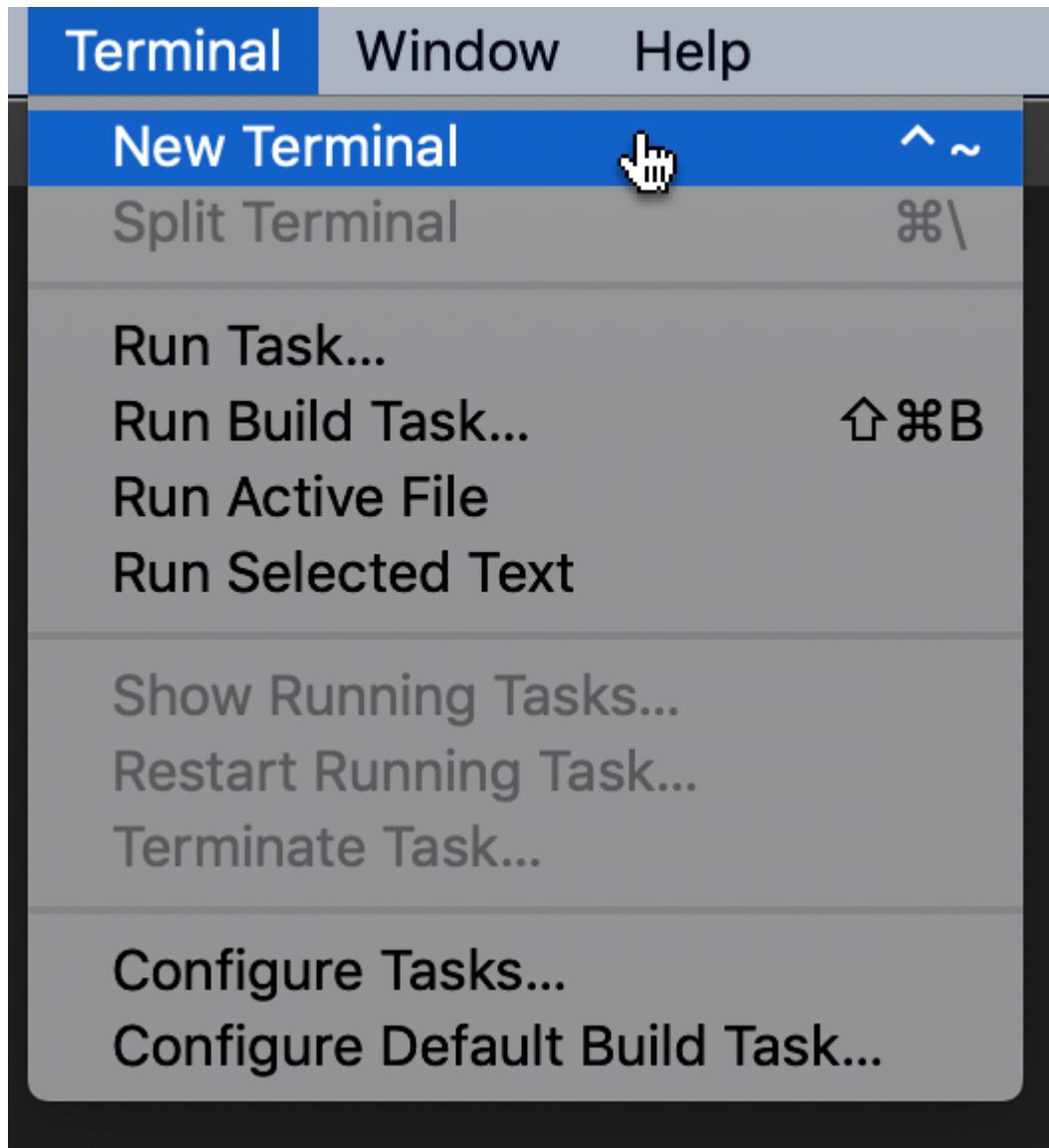
1. Go back to Visual Studio Code.
2. Open the *File* menu and select *Open Folder....*
3. Click on /aem65livetrials/cloudmanager_imp/CloudManagerOnline/webhook.
4. Click the *Open* button.



You may need to run `npm install` to initialize the project

The webhook project is available in a public GitHub repository. See the [Additional Resources](#) section for the link.

This folder contains two webhook implementations written in JavaScript -- `simple.js` and `webhook.js`. We're going to run `simple.js` script first. To do this, open the *Terminal* menu and select *New Terminal*.



This will open a new Terminal panel in Visual Studio Code with a shell prompt. In this panel, run the command

1. `npm install`. This will install the node modules
2. `npm run start-simple`. This will run the `simple.js` webhook.

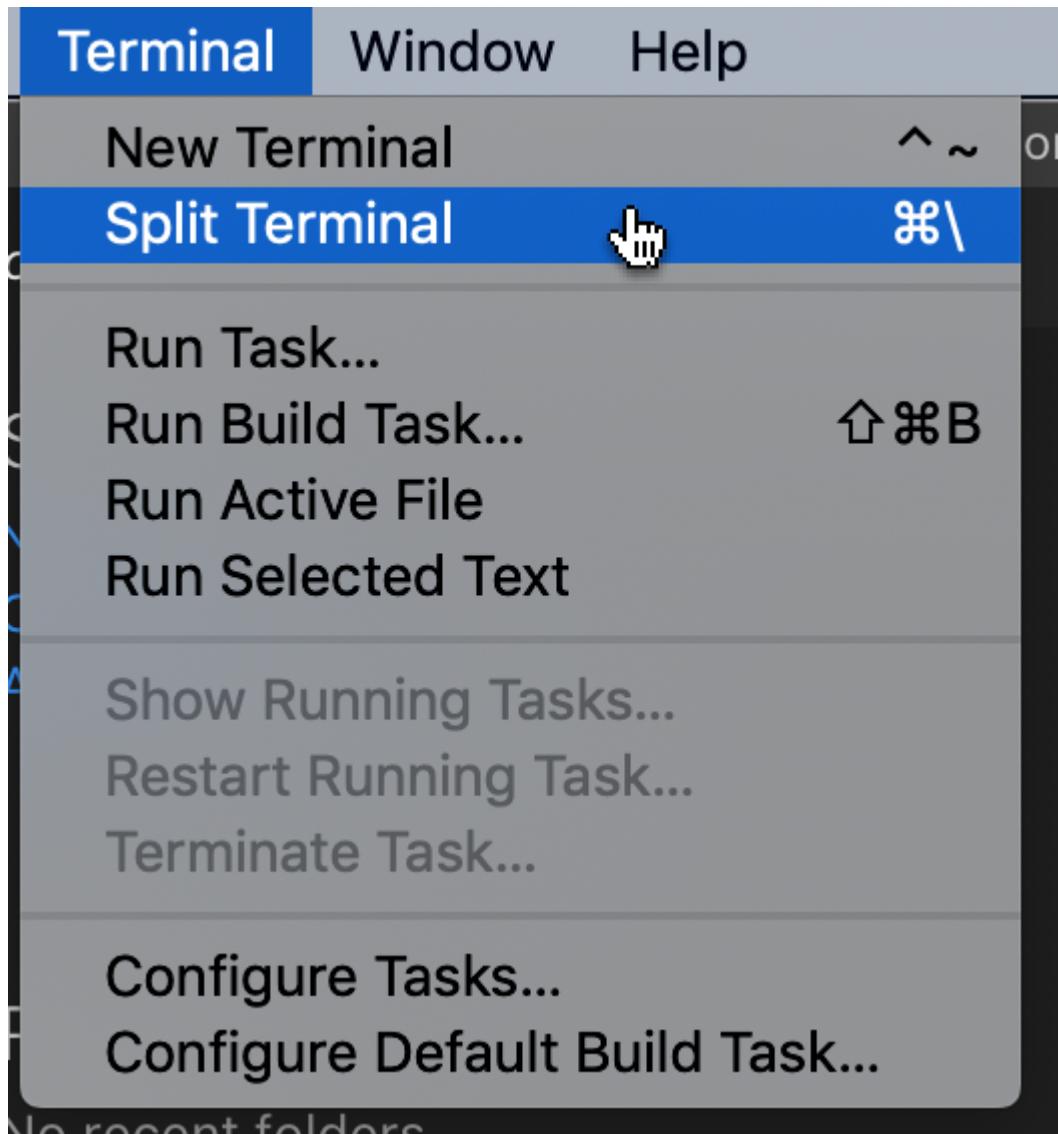
A screenshot of the Visual Studio Code terminal window. The tab bar at the top shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' selected. The terminal window itself has a title bar '1: node' and various control icons. The content of the terminal shows the following output:

```
1722-lab-webhook $ npm run start-simple
> cloudmanager-lab-webhook@1.0.0 start-simple /Users/labuser/Documents/1722-lab-webhook
> node simple.js

Your app is listening on port 3000
```

The terminal window has a dark background and light-colored text.

To test this out, we'll use a second terminal panel. To do this, open the *Terminal* menu and select *Split Terminal*.-- Or just open a new terminal window.



In the second Terminal panel, run the command `curl -X POST http://localhost:[PORT]/webhook`. What you should see is a simple response (`pong`) to the request and a message in the first panel showing that a webhook request was received.

The screenshot shows two terminal panes. The left pane displays the output of a Node.js application running in a terminal window. It shows the command `npm run start-simple` being run, followed by the application's code, and a message indicating it is listening on port 3000. The right pane shows the command `curl -X POST http://localhost:3000/webhook` being run, which returns the response `pong`.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
l722-lab-webhook $ npm run start-simple
> cloudmanager-lab-webhook@1.0.0 start-simple /Users/labuser/Documents/l722-lab-webhook
> node simple.js
Your app is listening on port 3000
webhook received
l722-lab-webhook $ curl -X POST http://localhost:3000/webhook
pong
l722-lab-webhook $
```

Congratulations! You've run a simple webhook.

Exercise 4.2

In order for Adobe I/O to access the webhook, it must be accessible on the public internet. Since your laptops are not publicly accessible, we will use a piece of software named [ngrok](#) to create a tunnel which allows

Adobe I/O to access the webhook.

To do this, go back to the second Terminal panel in Visual Studio Code and click the plus icon to open a second terminal window. In this window, type

```
ngrok http 3000
```

This command instructs *ngrok* to open a tunnel from a randomly generated URL to port 3000 on your lab workstation.

ngrok will start running and output something like this to confirm it is up and running:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
l722-lab-webhook $ npm run start-simple
> cloudmanager-lab-webhook@1.0.0 start-simple /Users/labuser/Documents/l722-lab-webhook
> node simple.js
Your app is listening on port 3000
webhook received
[]
```

ngrok by @inconsreivable (Ctrl+C to quit)

Session Status	online
Session Expires	7 hours, 59 minutes
Version	2.2.8
Region	United States (us)
Web Interface	http://127.0.0.1:4040
Forwarding	http://4830004e.ngrok.io -> localhost:3000
Forwarding	https://4830004e.ngrok.io -> localhost:3000
Connections	ttl opn rt1 rt5 p50 p90
	0 0 0.00 0.00 0.00 0.00

In this screen, the URL generated by *ngrok* is <https://4830004e.ngrok.io>

If you register for an *ngrok* account, you will be able to assign a custom domain name.

If you want to be sure this is working, feel free to open a third split Terminal panel and run `curl -X POST <ngrok URL>/webhook`.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
l722-lab-webhook $ npm run start-simple
> cloudmanager-lab-webhook@1.0.0 start-simple /22-lab-webhook
> node simple.js
Your app is listening on port 3000
webhook received
webhook received
[]
```

ngrok by @inconsreivable (Ctrl+C to quit)

Session Status	online
Session Expires	7 hours, 59 minut
Version	2.2.8
Region	United States (us
Web Interface	http://127.0.0.1:4040
Forwarding	http://4830004e.ngro
Forwarding	https://4830004e.ngro
Connections	ttl opn r
	1 0 0
HTTP Requests	-----
POST /webhook	200 OK

Now that we have a simple webhook running, we can setup and run the real webhook.

Lesson 5 - Webhook Setup

Objectives

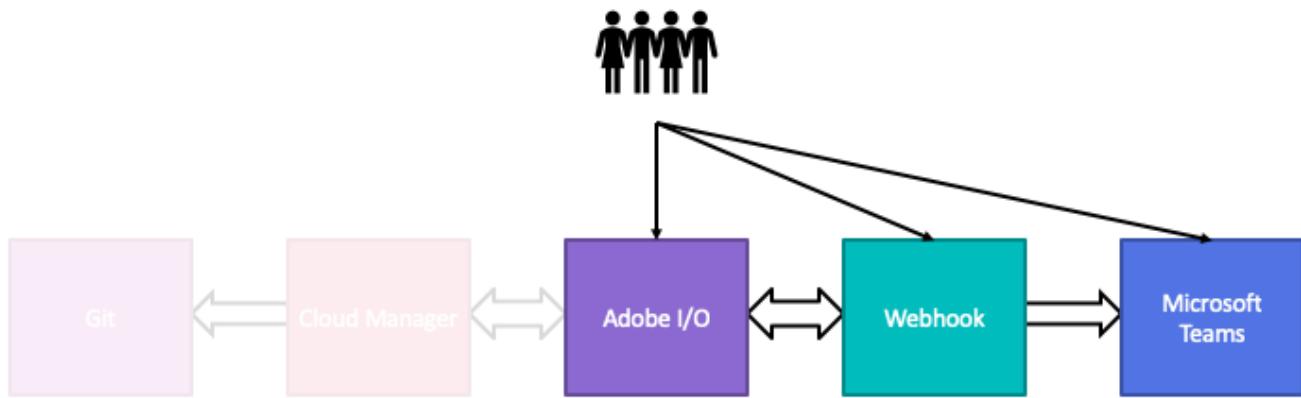
1. Register an API integration with Adobe I/O

2. Set up Webhook for publishing to Microsoft Teams
3. Register your webhook with Adobe I/O

Lesson Context

In this lesson, you will be configuring the actual webhook which will be invoked by Adobe I/O and will, in turn, invoke a Microsoft Teams API in order to send a notification when your Cloud Manager pipeline starts.

Lab Touchpoints – Lesson 5



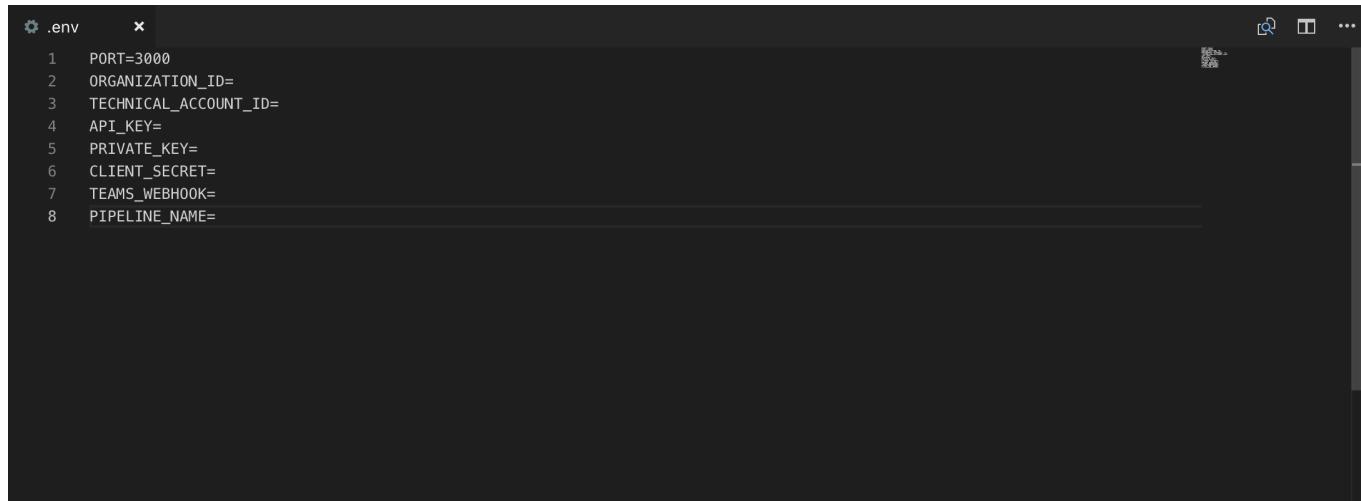
Exercise 5.1

Before proceeding with this exercise, make sure that ngrok is running; the URL generated each time you start ngrok may be different and that URL will be used in this exercise (and the next one). If you need to stop and restart ngrok, you will need to go back to the Adobe I/O console.

The bulk of the real webhook is already written (in the file `webhook.js` in the same directory opened in the prior lesson in Visual Studio Code). It does need to be configured so that it can invoke the Cloud Manager API and so that it can post messages to Microsoft Teams or Slack.

The webhook uses a Node.js library named `dotenv` to read its configuration. As the name suggests, this library uses a file named `.env`. In Visual Studio Code, click on this file in the Explorer to open it.

In the actual git project linked to from the [Additional Resources](#) section, this file is not provided; the file `.env.template` is present and needs to be copied to a file named `.env`.



.env

```
1 PORT=3000
2 ORGANIZATION_ID=
3 TECHNICAL_ACCOUNT_ID=
4 API_KEY=
5 PRIVATE_KEY=
6 CLIENT_SECRET=
7 TEAMS_WEBHOOK=
8 PIPELINE_NAME=
```

We now need to populate the first handful of lines in this file. To do this, we will register an Integration in the Adobe I/O Console. Before doing that, we need to generate a cryptographic certificate in order to securely sign requests to Adobe I/O.

Select the first Terminal panel (i.e. the one where the webhook is running, not the panel in which ngrok is running) and type Ctrl-C.

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout private.key -
out certificate.crt
```

You'll be prompted here for a series of values. The actual values don't really matter for the purposes of this lab, but you must *at minimum* provide an email address.

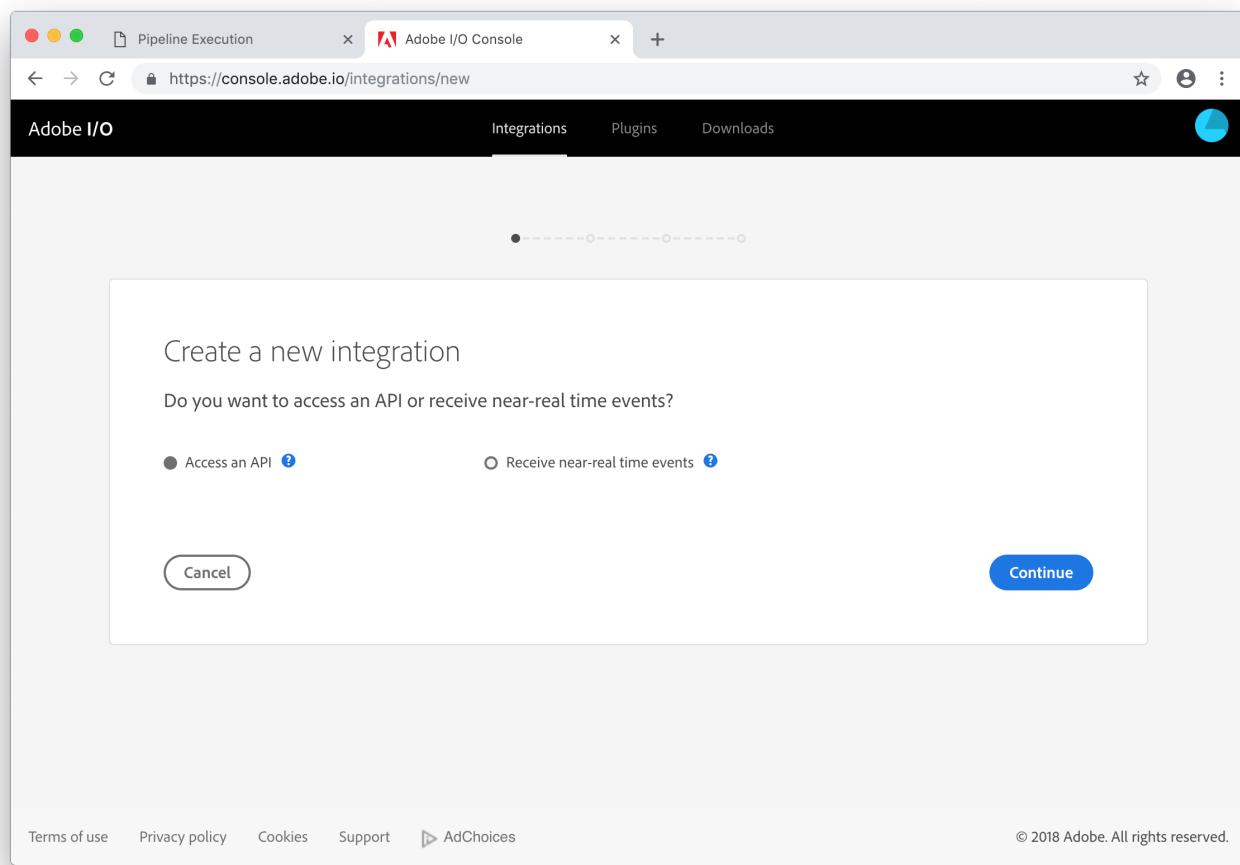
Now, switch back to Chrome. Open a new tab and navigate to <https://console.adobe.io/>

You shouldn't have to log in, but if you do, use the same credentials as used in Lesson 1.

Click the New integration button.

The screenshot shows the Adobe I/O Console interface. At the top, there are tabs for 'Pipeline Execution' and 'Adobe I/O'. Below the tabs, the URL is https://console.adobe.io/integrations. The main content area is titled 'Integrations' and displays a single integration entry: 'General API Access' under 'Service Account — General API Access'. A 'New integration' button is located in the top right corner of this card. Below the card, it says 'Showing 1 of 1 integrations'. At the bottom of the page, there are links for 'Terms of use', 'Privacy policy', 'Cookies', 'Support', and 'AdChoices', along with a copyright notice: '© 2018 Adobe. All rights reserved.'

The integration (which is how Adobe I/O refers to API clients) will both need to receive events and make API calls, so the answer to the question "Do you want to access an API or receive near-real time events" is actually *both*, but Adobe I/O requires this to be a two-step process. Leave *Access an API* selected (it should be the default) and click Continue.



Select Cloud Manager and click the Continue button.

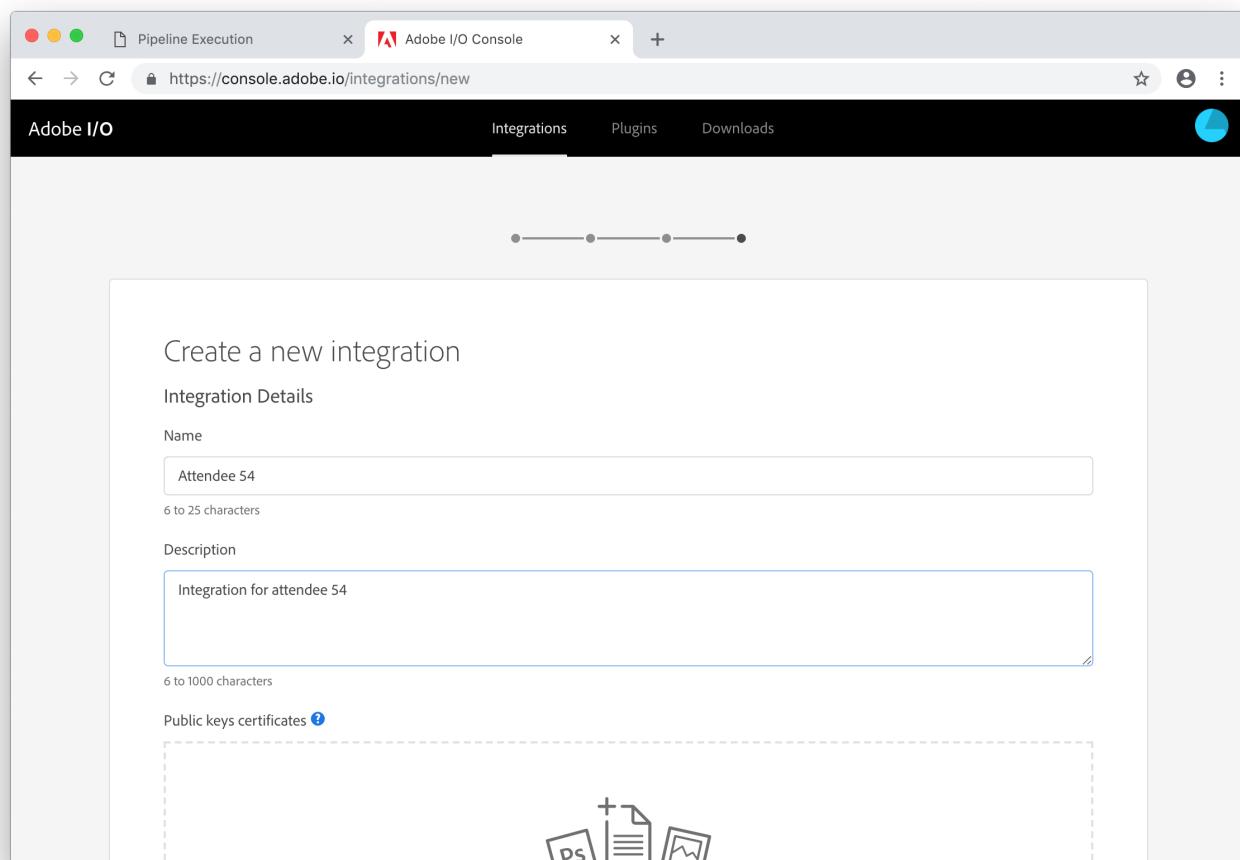
The screenshot shows the 'Create a new integration' page in the Adobe I/O Console. At the top, there are tabs for 'Integrations', 'Plugins', and 'Downloads'. Below the tabs, a progress bar indicates the user is on step 1 of 6. The main section is titled 'Create a new integration' and asks 'Select the Adobe service you wish to integrate with.' A dropdown menu shows 'Adobe Summit 2019 L722'. The page lists several service categories with their respective icons and API options:

- Adobe Experience Platform**: Includes Experience Platform API, Experience Platform Launch API.
- Adobe Sensei**: Includes Content AI (Beta), Visual search by Typekit.
- Adobe Services**: Includes I/O Events, I/O Management API, User Management API.
- Creative Cloud**: Includes Adobe Stock, Creative SDK, Lightroom, Photoshop, Task Queue Manager, Typekit Platform.
- Document Cloud**: Includes PDF Services.
- Experience Cloud**: Includes Adobe Analytics, Adobe Campaign, Adobe Target, AEM Brand Portal, Cloud Manager - Docs (selected), GDPR API, Journeys, Places.

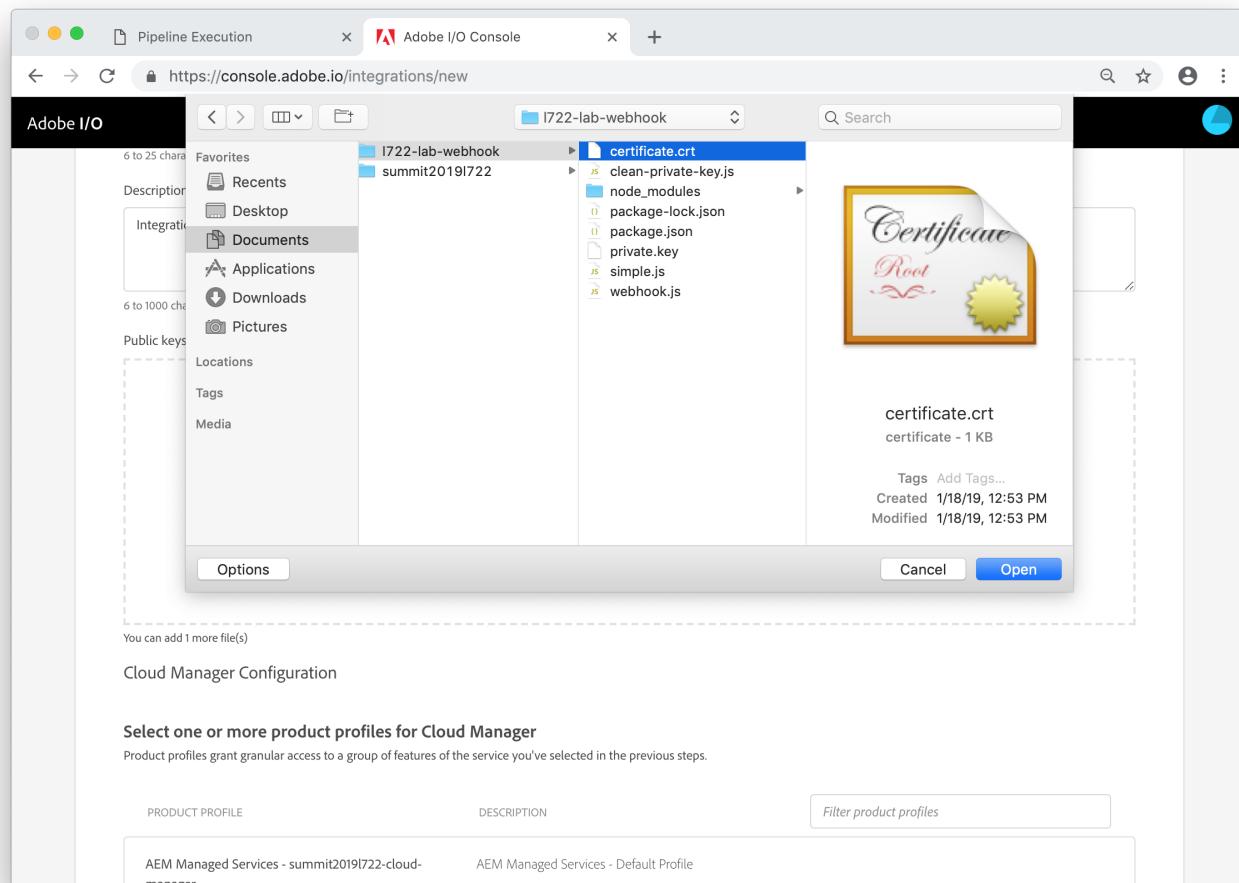
On the *Create a new integration* screen, ensure *New integration* is selected and click the Continue button.

The screenshot shows a web browser window for the Adobe I/O Console at the URL <https://console.adobe.io/integrations/new>. The browser's address bar and tabs are visible at the top. The main content area has a header "Create a new integration". Below it, a message says "You may create a brand new integration for this service, or update an existing one." There are two sections: "Create" (with a selected "New integration" option) and "Update" (which is empty and displays the message "No compatible integrations found. Create a new integration or select a different service."). At the bottom are "Cancel", "Back", and "Continue" buttons. A progress bar is shown above the main form. Below the form, there is a section titled "Select the Adobe service you wish to integrate with." It lists three categories with sub-options: "Adobe Experience Platform" (Experience Platform, Experience Platform API), "Adobe Sensei" (Content AI (Beta)), and "Adobe Services" (I/O Events, I/O Management API). A dropdown menu "Adobe Summit 2019 L722" is also visible.

Provide a name and description for your integration.



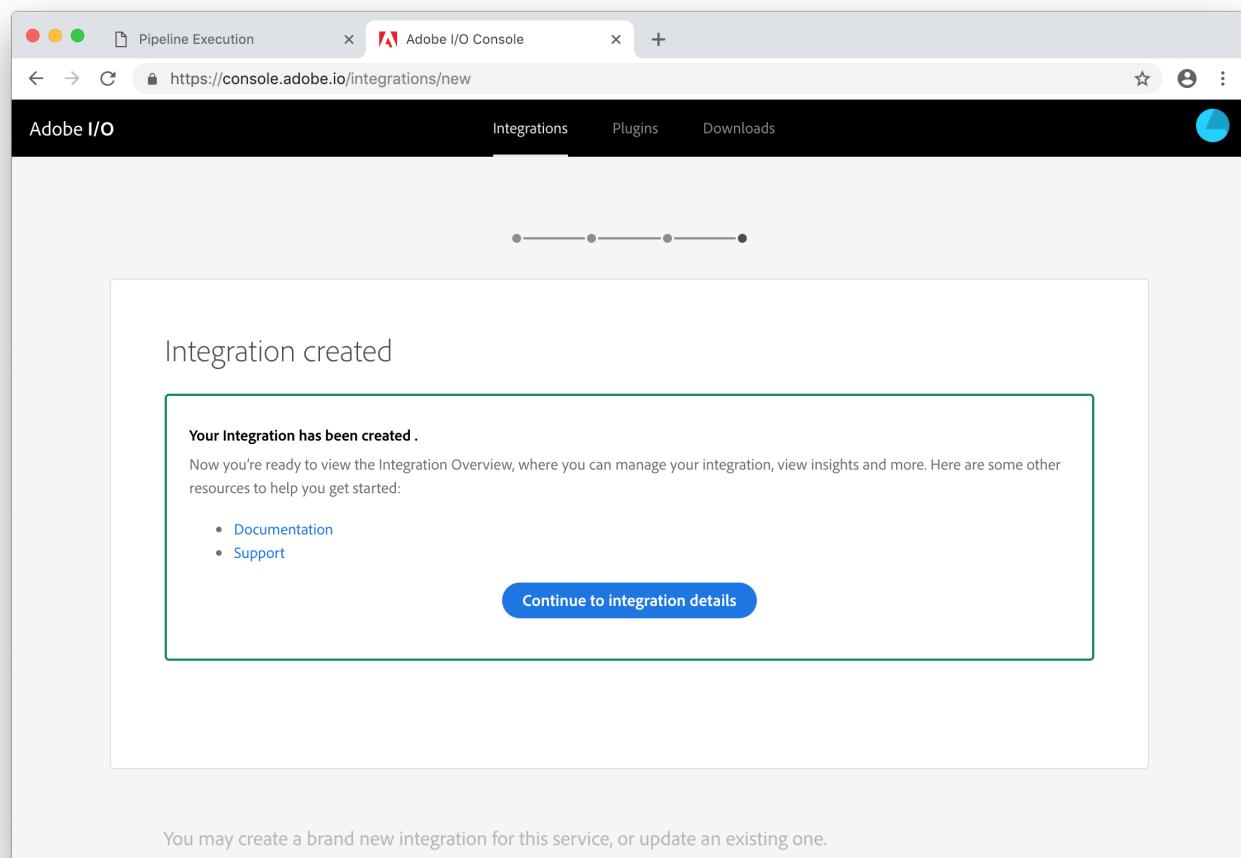
In the *Public keys certificates* box, click the *Select a File* link and open the *certificate.crt* file created by [openssl1](#).



Finally, select the *Cloud Manager - Deployment Manager Role* profile from the product profile list and click the *Create integration* button.

The screenshot shows a web browser window for the Adobe I/O Console at the URL <https://console.adobe.io/integrations/new>. The page is titled "Integrations". A table lists a single item: "certificate.crt" (Size: 0.001 MB) with a "Remove" button. Below the table, the text "Cloud Manager Configuration" is displayed. A section titled "Select one or more product profiles for Cloud Manager" contains a note: "Product profiles grant granular access to a group of features of the service you've selected in the previous steps." A "Filter product profiles" input field is present. A list of profiles is shown, with "Cloud Manager - Deployment Manager Role" selected (indicated by a blue border and checked checkbox). Other options include "AEM Managed Services - summit2019l722-cloud-manager", "Cloud Manager - Business Owner Role", "Cloud Manager - Developer Role", and "Cloud Manager - Program Manager Role". At the bottom are "Cancel" and "Create integration" buttons.

You'll now see a confirmation screen saying that your integration has been created. Click the *Continue to integration details* button to see the details of the created integration.



You now need to copy and paste the values for the API Key, Technical account ID, Organization ID, and Client Secret from this screen into the `.env` file in Visual Studio Code. Use the Copy button next to each value (clicking the *Retrieve client secret* button to show the Client Secret) and paste them into the corresponding variable in the `.env` file.

Attendee 54

Overview Insights Services Events JWT Delete

Client Credentials		Integration Details	
API Key (Client ID)	e3a64cccd7eaa40068111d0b3f2c2a459	Name	Attendee 54
Technical account ID	C1FC25C25C4215420A495EEF@techacct.adobe.com	Description	Integration for attendee 54
Technical account email	f5bbdd47-9991-4138-8e5b-dc0dab34d5e7@techacct.adobe.com	6 to 1000 characters	
Organization ID	C80E31365C1030E80A495CFC@AdobeOrg	6 to 1000 characters	
Client secret	Retrieve client secret	Update	
Public keys			
FINGERPRINT	EXPIRY DATE		

```
.env
1 PORT=3000
2 ORGANIZATION_ID=C80E31365C1030E80A495CFC@AdobeOrg
3 TECHNICAL_ACCOUNT_ID=C1FC25C25C4215420A495EEF@techacct.adobe.com
4 API_KEY=e3a64cccd7eaa40068111d0b3f2c2a459
5 PRIVATE_KEY=
6 CLIENT_SECRET=c2fe0de1-5a3a-4512-bc62-8ccbfa6c379
7 TEAMS_WEBHOOK=
8 PIPELINE_NAME=
```

The `PRIVATE_KEY` variable needs to be set as the value of the generated `private.key` file but *without any line breaks*. There's a helper script which will do this for you. To run this and copy the result to the clipboard, in the terminal panel type:

```
npm run clean-private-key | pbcopy
```

Then you can paste this value into the `.env` file.

The screenshot shows a terminal window with several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active and displays the command `npm run clean-private-key | pbcopy` followed by a prompt. To the right of the terminal, there is a status bar for an ngrok session. The status bar shows "ngrok by @inconshreveable" and "Session Status online". It also includes a note "(Ctrl+C to quit)" and a tab indicator "1: bash, ngrok".

```
1 PORT=3000
2 ORGANIZATION_ID=C80E31365C1030E80A495CFC@Adobe0rg
3 TECHNICAL_ACCOUNT_ID=C1FC25C25C4215420A495EEF@techacct.adobe.com
4 API_KEY=e3a64ccd7eaa40068111d0b3f2c2a459
5 PRIVATE_KEY-----BEGIN PRIVATE KEY-----MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDUL+8niLi7oHAmiXpN1UbCrWY092NveGj f
6 CLIENT_SECRET=c2fe0de1-5a3a-4512-bc62-8ccbfa6c379
7 TEAMS_WEBHOOK=
8 PIPELINE_NAME=
```

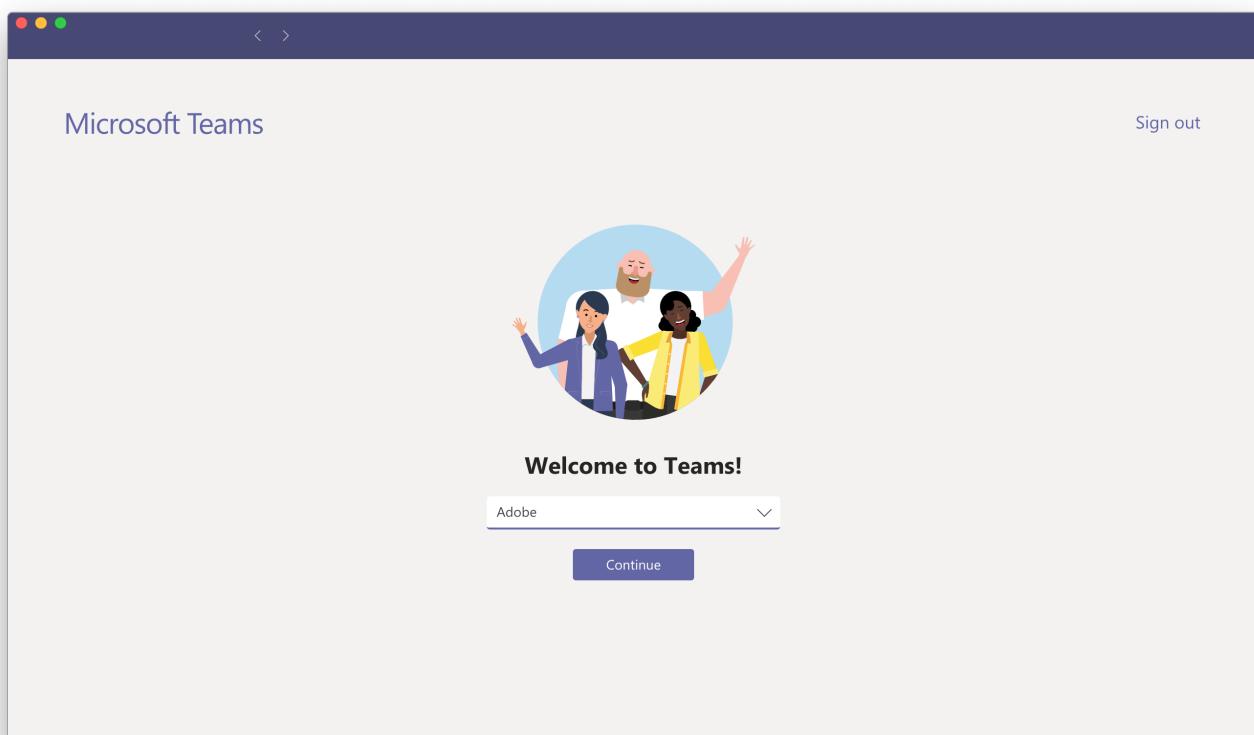
Exercise 5.2

The next value we need is a webhook URL which will be used to post messages to Microsoft Teams or Slack.

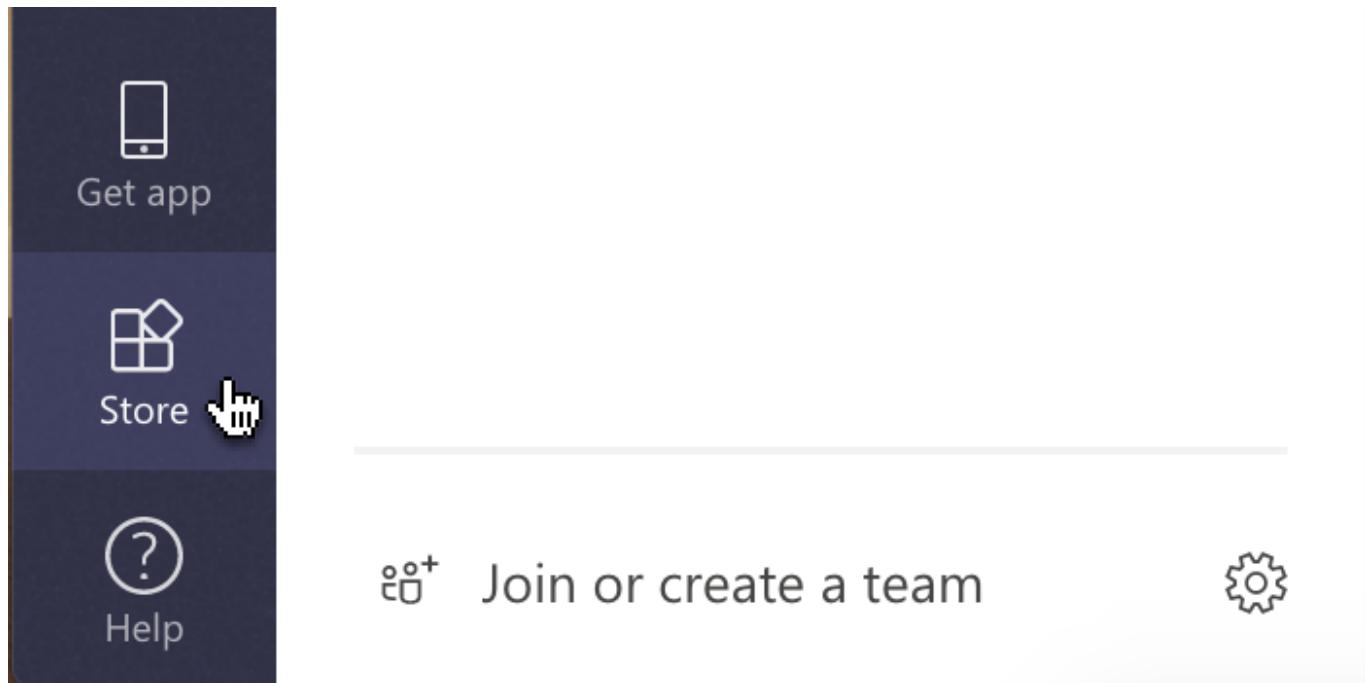
This part is a bit complicated, but essentially Adobe I/O is going to invoke a webhook on your lab machine which *in turn* will invoke a webhook hosted by Microsoft/Slack.

To start, open [Microsoft Teams](#)

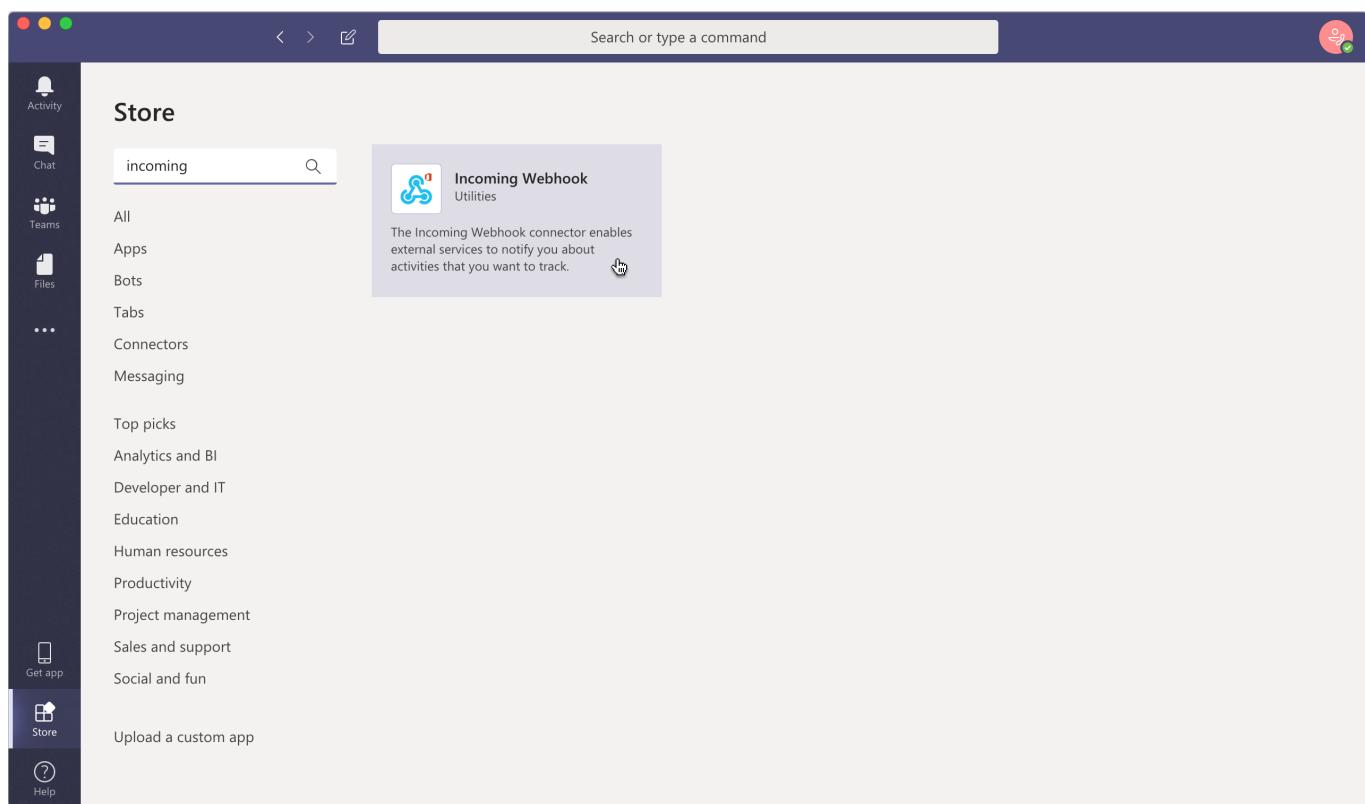
If you see the _Welcome to Teams! screen (see below), click the *Continue* button.



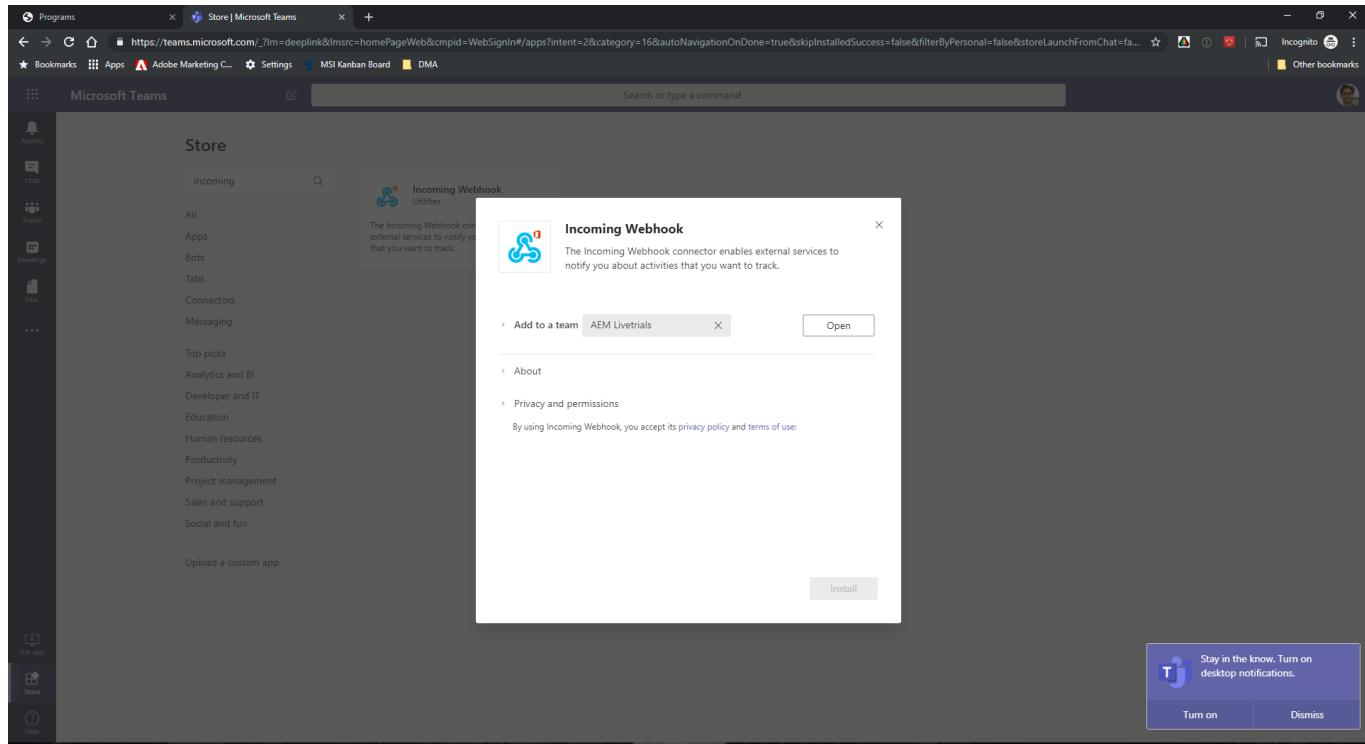
To create an Incoming Webhook integration, click the Store icon.



Search for *incoming* and click on *Incoming Webhook*.



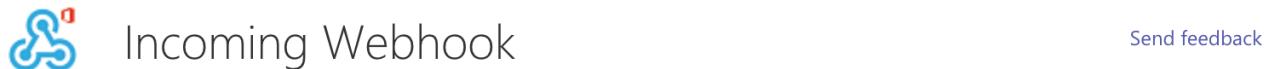
Select *Summit Lab 722* as the team. Since the connector is already installed, click the *Available* link.



The webhook should send messages to the General channel by default. On the next screen, click *Set up*.

You need to provide a name for your webhook. Do this with a name based on your attendee number and click the *Create* button.

Connectors for "General" channel in "Summit Lab 722" team



The Incoming Webhook connector enables external services to notify you about activities that you want to track. To use this connector, you'll need to create certain settings on the other service, which needs to support a webhook that's compatible with the [Office 365 connector format](#).

Enter a name for your IncomingWebhook connection.

Participant 54 Webhook

Customize the image to associate with the data from this Incoming Webhook.

Upload Image



You will now see a URL (it will begin with <https://outlook.office.com/webhook/>). Click the button to copy this to the clipboard and click the *Done* button.

Connectors for "General" channel in "Summit Lab 722" team

Upload Image

Copy the URL below to save it to the clipboard, then select Save. You'll need this URL when you go to the service that you want to send data to your group.

<https://outlook.office.com/webhook/192c0d4f-47e0-4a9b-8a2d-4a2a2a2a2a2a>

**Done****Remove**

Note: If you're a software developer and want to learn more about sending data to Office 365 using Incoming Webhook, see [Get started with Office 365 Connector Cards](#).

Now switch back to Visual Studio Code and paste the copied URL as the value of the **TEAMS_WEBHOOK** variable in the **.env** file.

```
1 PORT=3000
2 ORGANIZATION_ID=C80E31365C1030E80A495CFC@AdobeOrg
3 TECHNICAL_ACCOUNT_ID=C1FC25C25C4215420A495EEF@techacct.adobe.com
4 API_KEY=e3a64ccd7eaa40068111d0b3f2c2a459
5 PRIVATE_KEY-----BEGIN PRIVATE KEY-----MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDUL+8nILi7oHAMiXpN1UbCrWY092NveGj
6 CLIENT_SECRET=c2fe0de1-5a3a-4512-bc62-8ccbbfa6c379
7 TEAMS_WEBHOOK=https://outlook.office.com/webhook/192c0d4f-47e0-4a9b-8a2d-4a2a2a2a2a2a/
8 PIPELINE_NAME=
```

The last value which needs to be populated in the **.env** file is the name of the pipeline you created in Lesson 1. After adding this, make sure save the file in Visual Studio Code.

```
1 PORT=3000
2 ORGANIZATION_ID=C80E31365C1030E80A495CFC@AdobeOrg
3 TECHNICAL_ACCOUNT_ID=C1FC25C25C4215420A495EEF@techacct.adobe.com
4 API_KEY=e3a64ccd7eaa40068111d0b3f2c2a459
5 PRIVATE_KEY=====BEGIN PRIVATE KEY=====MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwgSkAgEAAoIBAQDUL+8niLi7oHAmiXpN1UbCrWY092NveGj f
6 CLIENT_SECRET=c2fe0de1-5a3a-4512-bc62-8ccbbfa6c379
7 TEAMS_WEBHOOK=https://outlook.office.com/webhook/192c2049-91b7-412c-9edb-aebb5bca5708@6925c7e5-8258-4d21-8488-2ee3c9eb14f4/
8 PIPELINE_NAME=Pipeline 54
```

Exercise 5.3

With our webhook fully configured, we can start it and register it with Adobe I/O.

In order to register a webhook with Adobe I/O, it must be running and, as mentioned in Lesson 4, accessible from the public internet.

To start the webhook, in the terminal panel in Visual Studio Code run

```
npm start
```

```
l722-lab-webhook $ npm start
> cloudmanager-lab-webhook@1.0.0 start /Users/labuser/Documents/l722-lab-
webhook
> node webhook.js
Your app is listening on port 3000
```

ngrok by @inconshreveable	
Session Status	online
Session Expires	7 hours, 29 minutes
Version	2.2.8
Region	United States (us)
Web Interface	http://127.0.0.1:4040
Forwarding	http://4830004e.ngrok.io -> localhost:3000
Forwarding	https://4830004e.ngrok.io -> localhost:3000
Connections	ttl opn rt1 rt5 p50 p90
	1 0 0.00 0.00 0.07 0.00
HTTP Requests	-----
POST /webhook	200 OK

Switch back to Chrome (you should still be on the Integration Details page in the Adobe I/O Console) and click the Events tab.

Under the *Add New Event Providers* section, select Cloud Manager and click the *Add event provider* button.

The screenshot shows the Adobe I/O Console interface. The top navigation bar includes tabs for Pipeline Execution, Adobe I/O Console, Integrations, Plugins, and Downloads. Below the navigation is a header with the text "Attendee 54" and a "Delete" button. The main content area has tabs for Overview, Insights, Services, Events (which is selected), and JWT. Under the "Events" tab, there are two sections: "Current Event Providers" and "Event Registration". The "Current Event Providers" section contains a message stating "No event providers have been configured for this integration." The "Event Registration" section contains a message "Add a webhook to receive events in real-time." with a "Add Event Registration" button. At the bottom, there is a "Add New Event Providers" section with a "Experience Cloud" icon and a "Cloud Manager" option, followed by a blue "Add event provider" button.

Then click the Add Event Registration button.

The screenshot shows the Adobe I/O Console interface. The top navigation bar includes tabs for Pipeline Execution, Adobe I/O, Integrations, Plugins, and Downloads. The main content area is titled "Attendee 54". Below this, there are tabs for Overview, Insights, Services, Events (which is selected), and JWT. A "Delete" button is visible in the top right corner of the main content area. The "Events" section is divided into two main sections: "Current Event Providers" and "Add New Event Providers". The "Current Event Providers" section contains a single entry for "Cloud Manager" with a gear icon. The "Add New Event Providers" section indicates "No event providers available." A "Add event provider" button is located at the bottom right of this section. A tooltip "Add Event Registration" with a cursor icon is shown over the "Add Event Registration" button.

Provide a name and description for your webhook and as the URL, use the ngrok-generated URL plus [/webhook](#), e.g. if ngrok displayed <https://4830004e.ngrok.io>, the webhook URL is <https://4830004e.ngrok.io/webhook>. Adobe I/O registrations can receive multiple types of events (even multiple types of events from different providers). For the purpose of this lab, we only *need* to subscribe to the *Pipeline Execution Started* event. Click the checkbox next to this event. Then click the Save button.

The screenshot shows the Adobe I/O Console interface for managing pipeline events. The top navigation bar includes links for Pipeline Execution, Adobe I/O, Integrations, Plugins, and Downloads. The main content area is titled "Event Details" and contains fields for "Event Registration Name" (set to "Webhook Attendee 54"), "Webhook URL (optional)" (set to "https://4830004e.ngrok.io/webhook"), and "Event Description" (set to "Webhook Attendee 54"). A "Journaling" section provides instructions for generating a unique API endpoint. To the right, a table lists available events to subscribe to, with "Pipeline Execution Started" selected. A "Save" button is at the bottom right.

Subscribe	Event name	Provider
<input checked="" type="checkbox"/>	Pipeline Execution Started	Cloud Manager
<input type="checkbox"/>	Pipeline Execution Step Ended	Cloud Manager
<input type="checkbox"/>	Pipeline Execution Step Started	Cloud Manager
<input type="checkbox"/>	Pipeline Execution Step Waiting	Cloud Manager
<input type="checkbox"/>	Pipeline Execution Ended	Cloud Manager

If you've entered the URL correctly, the event registration is now Active.

The screenshot shows the Adobe I/O Console interface. The top navigation bar includes tabs for Pipeline Execution, Adobe I/O Console, Integrations, Plugins, and Downloads. The main content area is titled "Events". Under "Current Event Providers", there is a single entry for "Cloud Manager". In the "Event Registration" section, a table lists one entry: "Webhook Attendee 54" (Status: Active, Webhook URL: <https://4830004e.ngrok.io/webhook>). Buttons for "View" and "Delete" are shown next to the entry. A "Add Event Registration" button is located at the top right of this section. Below these sections, there is a "Add New Event Providers" section with a message stating "No event providers available." and a "Add event provider" button.

However, if you have entered the wrong URL, you will see a message that the verification has failed. If you see this, click the View button and double check the URL.

The screenshot shows the Adobe I/O Console interface. At the top, there's a navigation bar with tabs for 'Integrations', 'Plugins', and 'Downloads'. Below that, the main title is 'Attendee 54'. Underneath, there are tabs for 'Overview', 'Insights', 'Services', 'Events' (which is underlined and has a red exclamation mark icon), and 'JWT'. On the far right, there's a 'Delete' button. The main content area starts with a section titled 'Current Event Providers' which lists 'Cloud Manager'. Below that is a 'Event Registration' section with a table:

Event name	Status	Webhook URL
⚠️ Webhook Attendee 54	Disabled	https://4830004e.ngrok.io/webhooks

Next to the table are three buttons: 'Retry', 'View', and 'Delete'. At the bottom of the main content area, there's a link 'Add New Event Providers'.

Lesson 6 - Testing it Out

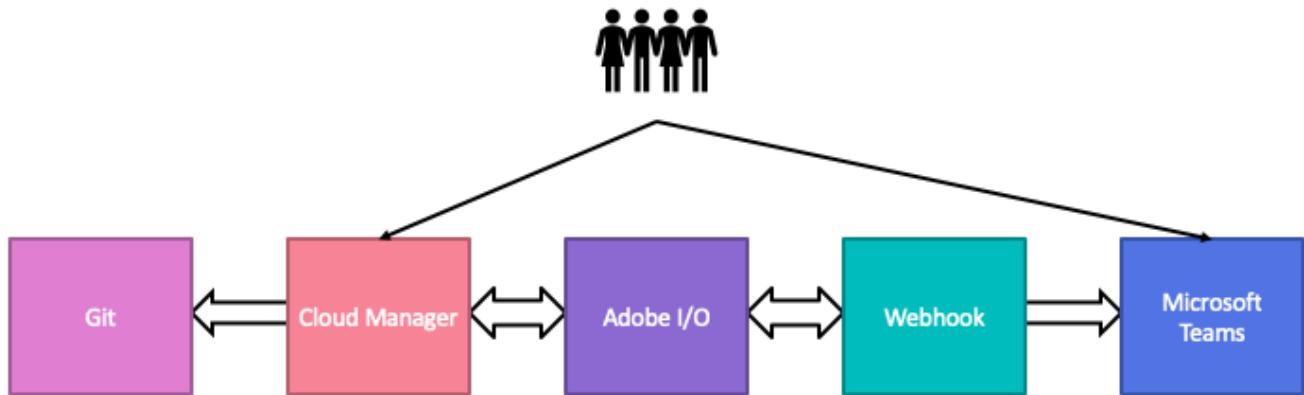
Objectives

1. Re-Execute the Pipeline
2. Watch Notifications in Microsoft Teams

Lesson Context

In this last lesson, you will test the process end to end by starting the pipeline in Cloud Manager and, if everything is working correctly, observe the resulting notification in Microsoft Teams.

Lab Touchpoints – Lesson 6



Exercise 6.1

At this point, your webhook and ngrok should be running and registered with Adobe I/O. So now it is time to run the pipeline again. Go back to the Overview page in Cloud Manager, find your pipeline and start it.

Exercise 6.2

Depending on how quickly everyone in the lab has reached this step, our Microsoft Teams channel is about to get very noisy. Each pipeline start should result in this kind of notification in the channel:

Participant 54 Webhook 1:46 PM

Update from Cloud Manager

Pipeline Started!!!

Reply

If you see this notification for your pipeline, feel free to experiment with the webhook (see the [Next Steps](#) section for some ideas).

If not, you should go back to Lesson 5 and make sure the `.env` file is set up correctly. If you need to change anything in this file, you need to stop the webhook process (using Ctrl-C) and restart it.

Another thing to look at is the Debug Tracing tab in the Adobe I/O console. This will list the requests sent to your webhook. To see this information, click the View button for your Event Registration in the console and select the Debug Tracing tab. This will show you the requests made to your webhook by Adobe I/O.

Event Details Debug Tracing

Debug tracing All statuses ▾ Refresh list

⚠ 7e2408a7-8728-4e29-b0f7-52d77237e15b-probe-1548170634197 (404 error)	2019-01-22 10:23:54 ↵
⚠ 3c0b8f9a-757a-4e8c-8b1c-f2838c18065f (404 error)	2019-01-22 10:23:53 ↵
cbc840e8-d687-4316-85b4-8d14481c3fe9	2019-01-18 13:46:33 ↵
Oab3e2c6-733f-45cc-bb2b-71789722350e	2019-01-18 13:31:56 ↵
7e2408a7-8728-4e29-b0f7-52d77237e15b-probe-1547836265779	2019-01-18 13:31:06 ↵
⚠ 7e2408a7-8728-4e29-b0f7-52d77237e15b-probe-1547836084257 (404 error)	2019-01-18 13:28:05 ↵
7e2408a7-8728-4e29-b0f7-52d77237e15b-probe-1547836034595	2019-01-18 13:27:15 ↵

Close

Clicking on one of these entries will show you the headers and body for both the request and response.

cbc840e8-d687-4316-85b4-8d14481c3fe9 2019-01-18 13:46:33 ▾

Request Response (200) Request completed in: 590ms

HEADERS	PAYOUT
Request URL: https://4830004e.ngrok.io/webhook Request method: POST Content-Type: application/json; charset=utf-8 x-adobe-event-id: cbc840e8-d687-4316-85b4-8d14481c3fe9 x-adobe-provider: cloudmanager x-adobe-signature: h7o/UogbQr5tbB/T77cKj7M2AZGUzJuWZARL6+Hnea8= x-adobe-event-code: pipeline_execution_start user-agent: Adobe/1.0	PAYOUT <pre>{"event_id": "cbc840e8-d687-4316-85b4-8d14481c3fe9", "event": {"@id": "urn:oid:cloudmanager:2e4a9943-40c5-43f3-9738-446alc7af626", "@type": "https://ns.adobe.com/experience/cloudmanager/event/started", "activitystreams:published": "2019-01-18T18:46:25.564Z", "activitystreams:to": [{"@type": "xdmImsOrg", "xdmImsOrg:id": "C80E31365C1030E80A495CFC@AdobeOrg"}, "activitystreams:actor": {"@type": "xdmImsUser", "xdmImsUser:id": "48603B3041E986A499201549@AdobeID"}, "activitystreams:object": {"@id": "https://cloudmanager.adobe.io/api/program/1/pipeline/8702/execution/1970", "@type": "https://ns.adobe.com/experience/cloudmanager/pipeline-execution"}, "xdmEventEnvelope:objectType": "https://ns.adobe.com/experience/cloudmanager/pipeline-execution"}}</pre>

If you're still stuck, check with a Lab TA.

Next Steps

In this lab, we have really just scratched the surface of what can be done with the Cloud Manager API and webhooks. If you've gotten this far and want to experiment some more, here are some ideas of things to try:

Remember that anytime you change the webhook code or the `.env` file, you need to stop and restart the webhook process, but you shouldn't restart ngrok.

- Change the notification sent to Microsoft Teams when a pipeline is started. Hint: open up `webhook.js` and find where this message is defined.
- Send a message to Microsoft Teams when the pipeline ends. Hint: each time the webhook is invoked, it is passed a JSON object where the `@type` property indicates the event type. The list of event types can be found in the [API Reference](#).
- Change the color or title of the notification sent to Microsoft Teams. Hint: the current color is `0072C6`; look for this in `webhook.js`.
- Instead of (or in addition to) Microsoft Teams, send a notification to a different service or even [IFTTT](#). Hint: most services (including IFTTT) support the same kind of incoming webhooks as Microsoft Teams, just with a different payload.
- As seen in Lesson 1, CI/CD pipelines in Cloud Manager can be triggered on changes to the git repository.
 - Reconfigure your pipeline (but only your pipeline) to use this trigger and try pushing another change to the project code from Visual Studio Code and see the pipeline re-execute. Hint: You can only edit a pipeline's configuration when it is idle.
 - Add the trigger for the execution to your Microsoft Teams notification. Hint: how an execution was triggered is contained in the API response used to get the execution details and will be either `MANUAL` or `ON_COMMIT`.

Additional Resources

- To learn more about Cloud Manager, visit https://www.adobe.com/go/aem_cloud_mgr_userguide_en
- To learn more about the Cloud Manager API, visit <https://www.adobe.io/apis/experiencecloud/cloud-manager/docs.html>
- To see the reference documentation for the Cloud Manager API, visit <https://www.adobe.io/apis/experiencecloud/cloud-manager/api-reference.html>
- To obtain the source code used in the pipeline lessons, visit <https://github.com/adobe/summit2019-l722-lab-project>
- To obtain the source code used in the webhook lessons, visit <https://github.com/adobe/summit2019-l722-lab-webhook>
- To learn more about the `MessageCard` data used for Microsoft Teams notifications, visit <https://docs.microsoft.com/en-us/outlook/actionable-messages/message-card-reference>.