Adobe I/O

Building Composable AI with Adobe Sensei functions and Adobe I/O Runtime

Adobe Summit 2018

Dawn Burrows, Sr Manager, Adobe I/O

Holly Schinsky, Sr Computer Scientist, Adobe I/O

Dragos Dascalita Haut, Principal Engineer, Adobe I/O

@adobeio

www.adobe.io

What You Will Learn

In this lab you will learn how to build a serverless app that takes advantage of Adobe's cloud offerings, Adobe I/O developer tools and APIs and Adobe Sensei functions. Adobe Sensei functions enable the enterprise and partners to consume, compose and extend Adobe Sensei.

Technology Used

- Apache OpenWhisk
- Adobe I/O developer tools and APIs

Project Exercises/Solutions

The folder for the exercises and solutions have been pre-loaded on your workstation in your user directory at `~/adobe-sensei-actions-lab`

> *If at any point you are stuck in an exercise, simply replace your version with the* `composition.js` *from the corresponding* ***solutions*** *folder.*

Requirements & Dependencies

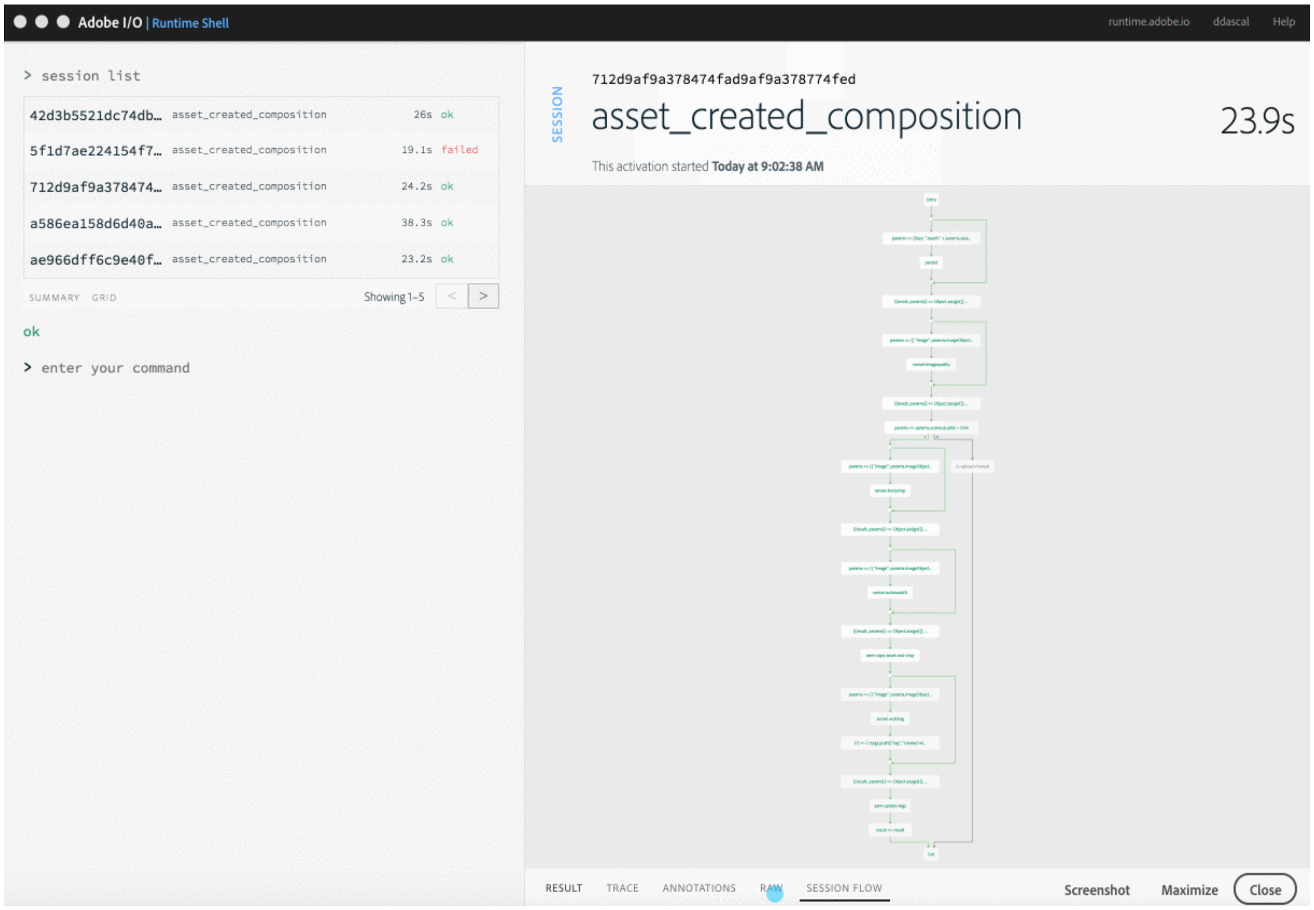
- Visual Studio Code
- Browser
- Internet Connection
- Adobe Creative Cloud userid/pw
- Adobe Experience Manager instance
- Adobe Experience Manager userid/pw
- Images (stock photos provided)

Issues/Feedback

- Please file an issue here if you run into any problems or simply have feedback.
- You can also use the *Comments* section at the bottom of each module to ask a question or report a problem.
- Or find me on Twitter:

  Follow
  @adobeio

Demo & Overviews



```
> session list
42d3b5521dc74db…   asset_created_composition   26s    ok
5f1d7ae224154f7…   asset_created_composition   19.1s  failed
712d9af9a378474…   asset_created_composition   24.2s  ok
a586ea158d6d40a…   asset_created_composition   38.3s  ok
ae966dff6c9e40f…   asset_created_composition   23.2s  ok
SUMMARY   GRID                          Showing 1–5  <  >
ok
> enter your command
```

712d9af9a378474fad9af9a378774fed

asset_created_composition 23.9s

This activation started **Today at 9:02:38 AM**

Serverless Concepts

- `action` – (aka: function) a piece of code that performs one specific task that can be deployed to the Adobe I/O Runtime as source code. An action performs work when invoked via REST API. Actions can also automatically respond to events from the Adobe Cloud Platforms via a webhook and trigger.
- `trigger` – a statement that you want to react to a certain event and fired when one is received.
- `rule` – a rule associates a trigger with an action. Every time a trigger fires, the rule invokes the associated action.
- `composition` – informally named apps, run in the cloud using automatically managed compute and memory resources and have control over action invocation and data flow.

Adobe I/O Runtime Shell

Overview

The Adobe I/O Runtime Shell is a graphical interface to help you visualize and debug your serverless functions and compositions with great ease. It has been pre-installed on your machine and you will use it throughout the workshop. Before using it however, you must authorize your namespace to give yourself your own sandbox to play in for the duration of this lab.

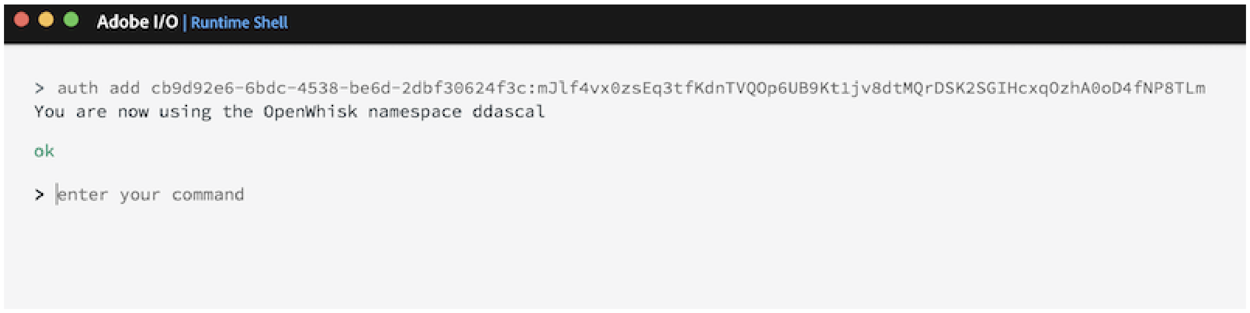
Setup

1. Locate the **Adobe I/O icon** in your application toolbar to open the **Adobe I/O Runtime Shell**.

2. Authorize your designated Adobe I/O Runtime namespace based on the auth key provided for your userid by entering the following command:

   ```
   auth add your_namespace_key_goes_here
   ```

   Verify you receive a response for your userid similar to below before continuing:

   ```
   > auth add cb9d92e6-6bdc-4538-be6d-2dbf30624f3c:mJlf4vx0zsEq3tfKdnTVQOp6UB9Kt1jv8dtMQrDSK2SGIHcxqOzhA0oD4fNP8TLm
   You are now using the OpenWhisk namespace ddascal

   ok

   > enter your command
   ```

Learning Exercises

1. Using the **Adobe I/O Runtime Shell**, create a new `app` (aka: `composition`) based on the built-in `hello.js` demo using the following command:

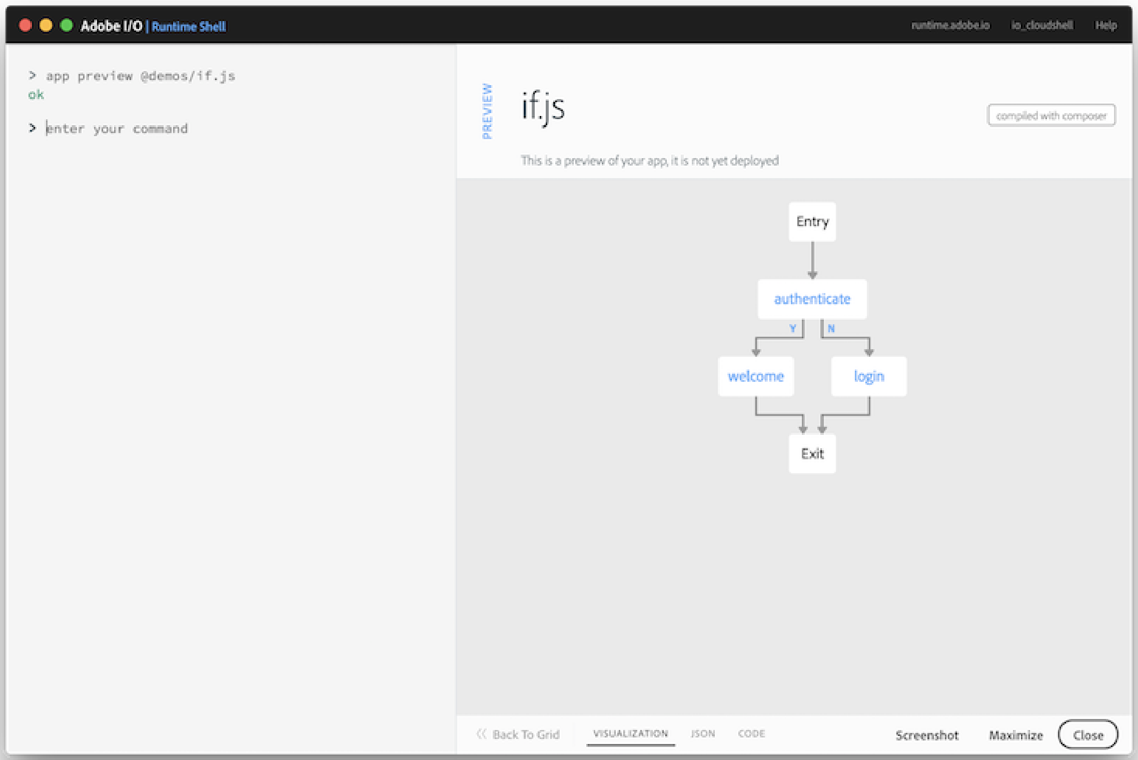
   ```
   app create hello-app @demos/hello.js
   ```

2. Next, invoke your `hello-app` with a `name` parameter:

   ```
   app invoke -p name sensei
   ```

3. Next, preview the built-in `if.js` demo to see an example of a composition with a flow structure:

   ```
   app preview @demos/if.js
   ```



```
> app preview @demos/if.js
ok
> enter your command
```

4. Take a moment to click on the **CODE** tab and notice how it uses the `authenticate` *action* as the condition, and takes the `welcome` or `login` action path depending on the result returned.

   ```
   composer.if(
       /* cond */
       'authenticate',
       /* then */
       'welcome',
       /* else */
       'login')
   ```

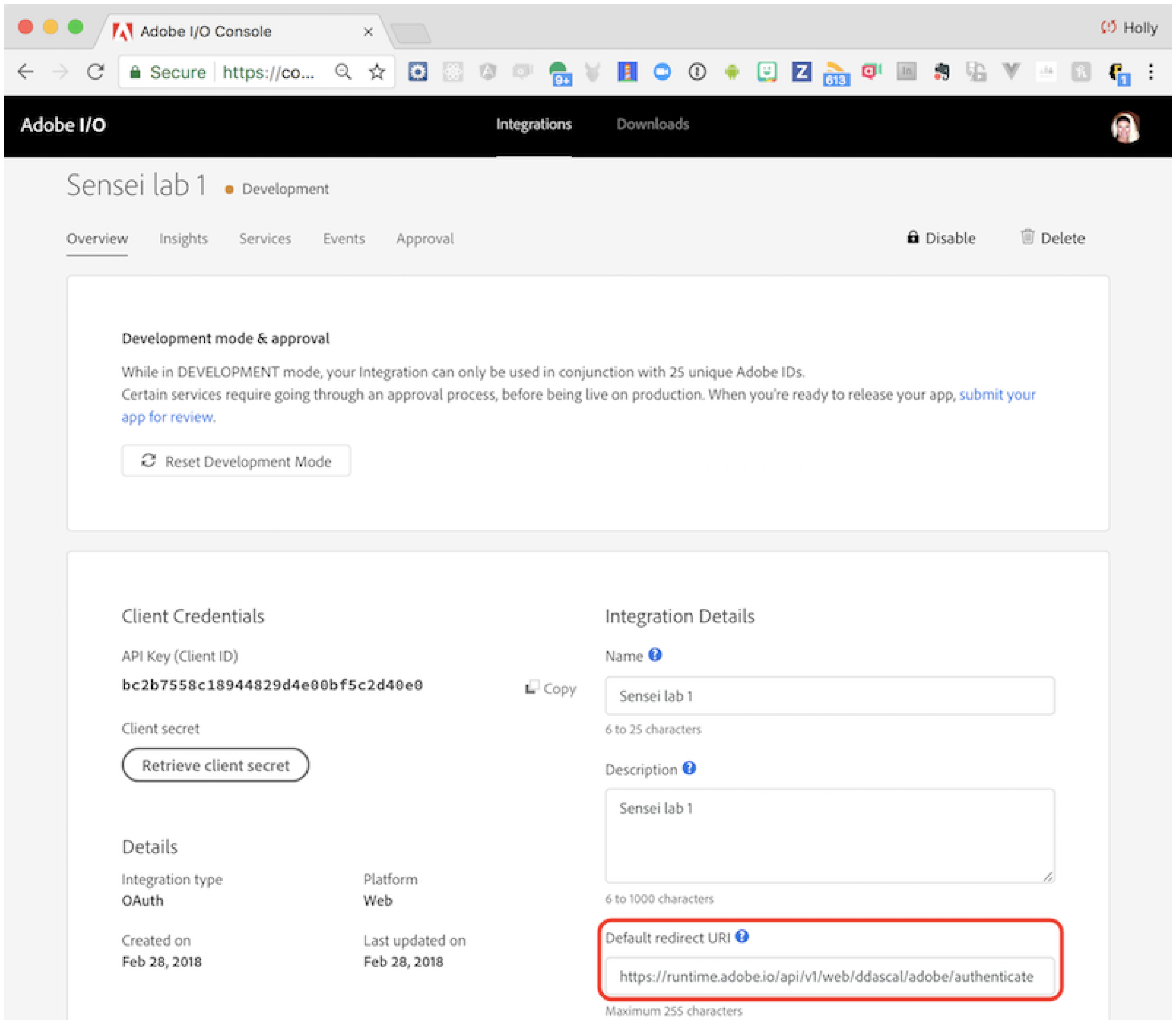
Adobe Console Integration

Overview

The Adobe I/O Console provides the definitive destination for any developer looking to engage with and integrate Adobe cloud, mobile, and web technologies.

Setup

1. Open your browser to the Adobe I/O Console and login with your pre-defined CC id and pw.
2. Ensure the **Integrations** tab is selected, then locate the integration by the name of `Sensei lab X` (where X is your assigned lab #) and click on it.
3. Once it's open, locate the **Default redirect URI** for your assigned user number and copy it. For instance, the one assigned to user **Sensei Lab 1** is outlined in red below:



Sensei lab 1 • Development

Overview Insights Services Events Approval Disable Delete

Development mode & approval

While in DEVELOPMENT mode, your Integration can only be used in conjunction with 25 unique Adobe IDs.
Certain services require going through an approval process, before being live on production. When you're ready to release your app, submit your app for review.

Reset Development Mode

Client Credentials

API Key (Client ID)
bc2b7558c18944829d4e00bf5c2d40e0 Copy

Client secret
Retrieve client secret

Details

Integration type
OAuth

Platform
Web

Created on
Feb 28, 2018

Last updated on
Feb 28, 2018

Integration Details

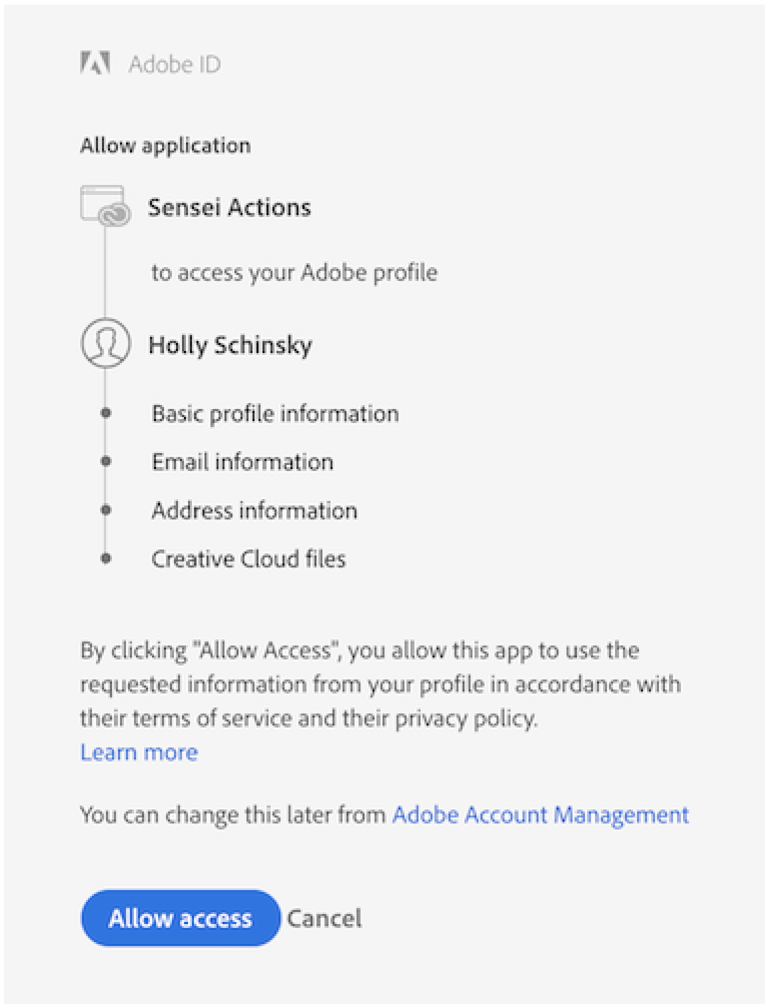
Name
Sensei lab 1
6 to 25 characters

Description
Sensei lab 1
6 to 1000 characters

Default redirect URI
https://runtime.adobe.io/api/v1/web/ddascal/adobe/authenticate
Maximum 255 characters

> ***NOTE:*** *This URL will be different for each user so make sure you locate the one for your specific user number.*

4. Open the URL copied from the **Default redirect URI** in the previous step to authorize your app to access your Creative Cloud files. When it is run the first time, you will receive a pop-up like below, where you should click on the **Allow Access** button. You will then be re-routed to the Adobe Creative Cloud site where you can login with the Creative Cloud userid/pw provided to you for the lab.



Adobe ID

Allow application

Sensei Actions

to access your Adobe profile

Holly Schinsky

- Basic profile information
- Email information
- Address information
- Creative Cloud files

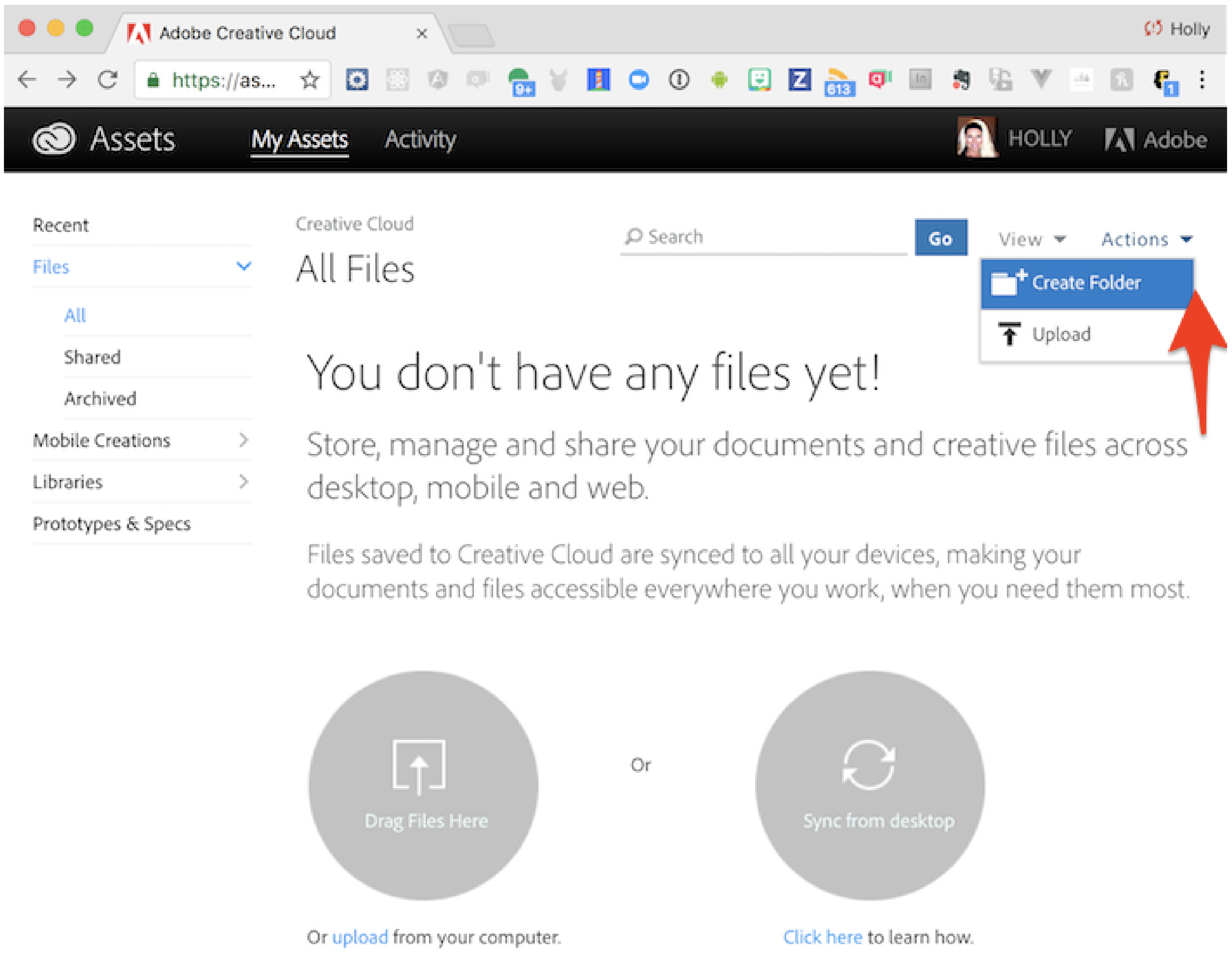
By clicking "Allow Access", you allow this app to use the requested information from your profile in accordance with their terms of service and their privacy policy.
Learn more

You can change this later from Adobe Account Management

> ***NOTE:*** *Don't worry if you did not receive the pop-up, it may have been run already in a previous lab.*

5. Once logged in, look for the **Creative Cloud Files** or **Files** link under **Assets** and click on it.

6. **IMPORTANT:** Before moving on, you must create your own personal folder to be used throughout this workshop. On the top right, click on the **Actions** arrow and choose **Create Folder** from the options. Name it `sensei-lab-x` where `x` is your assigned lab user number.



All Files

You don't have any files yet!

Store, manage and share your documents and creative files across desktop, mobile and web.

Files saved to Creative Cloud are synced to all your devices, making your documents and files accessible everywhere you work, when you need them most.

Or upload from your computer.

Click here to learn how.

Event Handler Webhook Definition

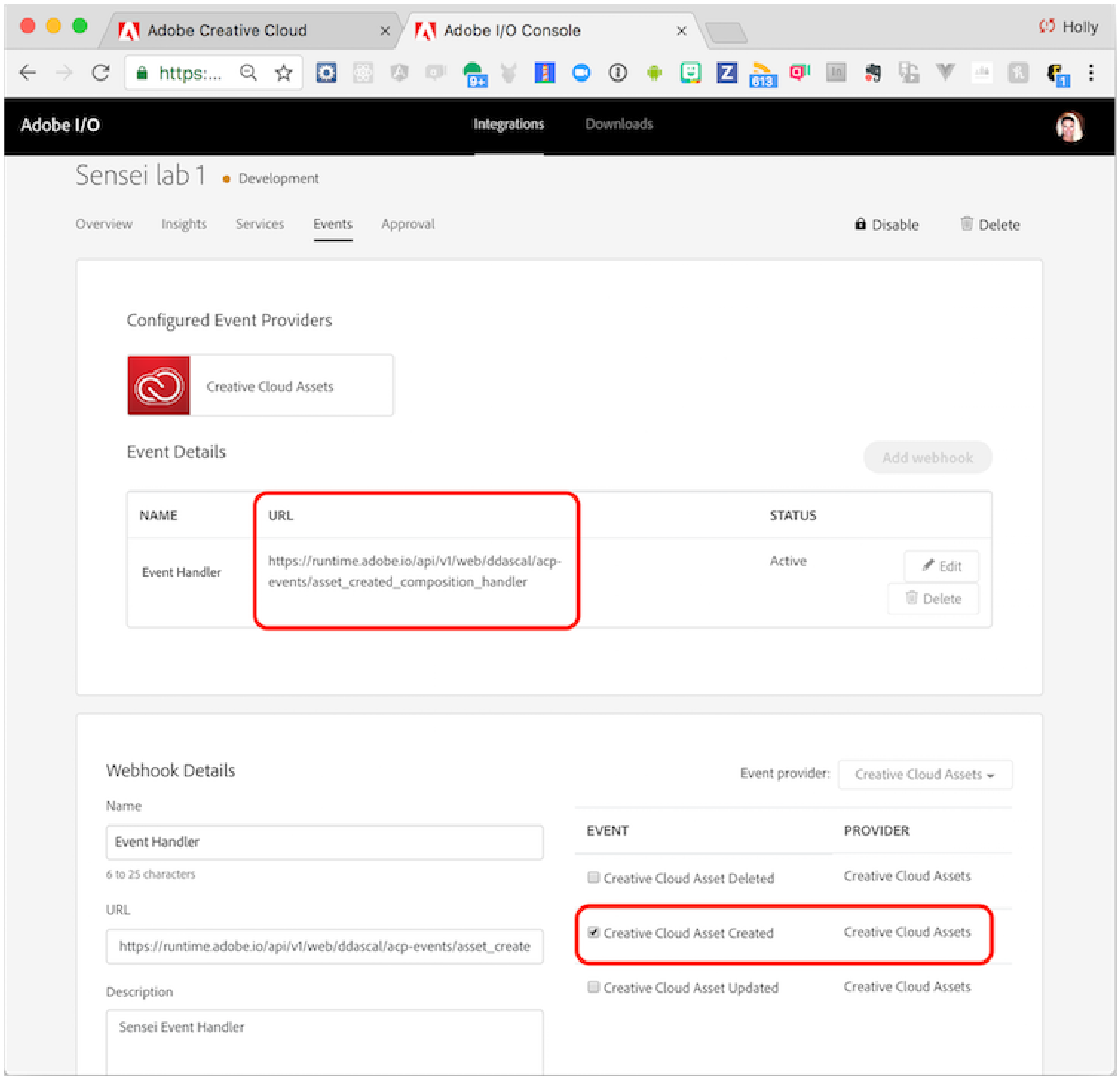
Overview

Understand how your Adobe I/O integration is configured with a webhook action to respond to **Creative Cloud Asset Created** events.

Steps

1. Go back to the Adobe I/O Console, ensure the **Integrations** tab is selected and click on your pre-defined integration again (ie: `Sensei lab X` - where X is your assigned lab #).

2. Select the **Events** tab and notice an item listed with the name **Event Handler**. Click on the **Edit** button to the right of it item to expand the details of the configured webhook, like shown below:



Adobe I/O

Sensei lab 1 ● Development

Configured Event Providers: Creative Cloud Assets

Event Details

NAME	URL	STATUS
Event Handler	https://runtime.adobe.io/api/v1/web/ddascal/acp-events/asset_created_composition_handler	Active

Webhook Details

Name: Event Handler

URL: https://runtime.adobe.io/api/v1/web/ddascal/acp-events/asset_create

Description: Sensei Event Handler

Event provider: Creative Cloud Assets

EVENT	PROVIDER
☐ Creative Cloud Asset Deleted	Creative Cloud Assets
☑ Creative Cloud Asset Created	Creative Cloud Assets
☐ Creative Cloud Asset Updated	Creative Cloud Assets

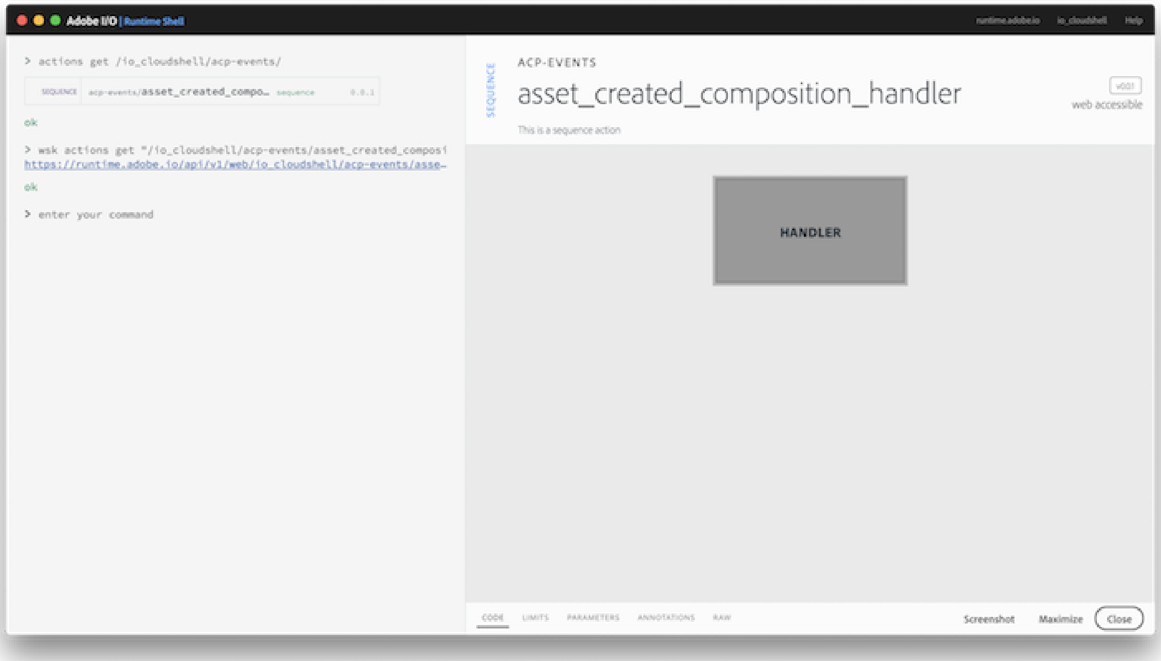
> *Notice the items outlined in red above. These two fields define a URL endpoint event handler to be invoked any time a Creative Cloud Asset Created event occurs.*

3. The URL defined for the webhook event handler points to a pre-defined `action` you could now view in the Adobe I/O Runtime Shell. Switch back to the Adobe I/O Runtime Shell and use the `action get` command followed by your specific `namespace/package/handler` from the URL directly following the `https://runtime.adobe.io/api/v1/web` portion of the URL:

   For instance, if the URL was `https://runtime.adobe.io/api/v1/web/sensei-lab-1/acp-events/asset_created_composition_handler` you would run the following command:

```
action get acp-events/asset_created_composition_handler
```

In the result you should see a SEQUENCE action with the name of `asset_created_composition_handler` in package `acp-events` .

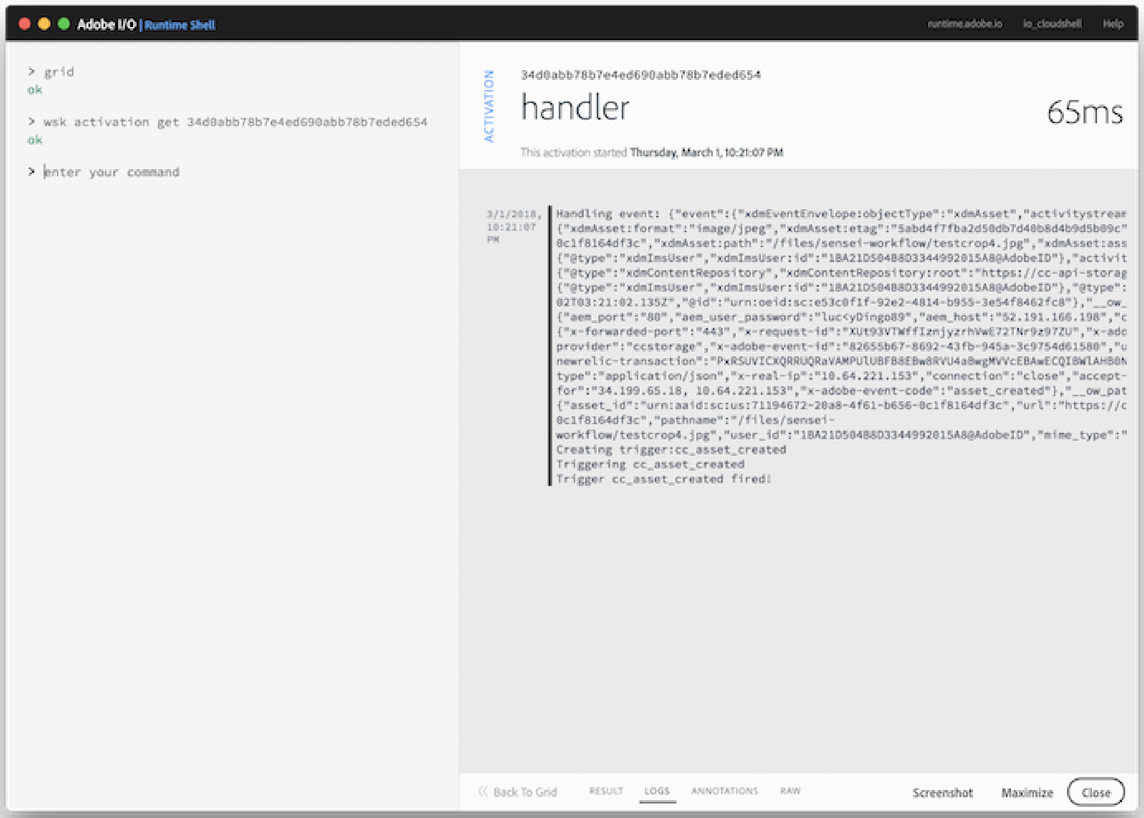


```
> actions get /io_cloudshell/acp-events/
SEQUENCE   acp-events/asset_created_compo…   sequence   0.0.1
ok

> wsk actions get "/io_cloudshell/acp-events/asset_created_composi
https://runtime.adobe.io/api/v1/web/io_cloudshell/acp-events/asse…
ok

> enter your command
```

> ***NOTE:*** *You can also list actions specific to a namespace/package and use a command like:* `actions get /sensei-lab-x/acp-events/` *with your assigned namespace/lab number.*

4. Click on the **HANDLER** action defined within the `asset_created_composition_handler` and take a look at the code definition briefly to see more details on how this handler works.

   An example of the log output from an invocation of this handler is shown below for reference:



```
> grid
ok

> wsk activation get 34d0abb78b7e4ed690abb78b7eded654
ok

> enter your command
```

34d0abb78b7e4ed690abb78b7eded654

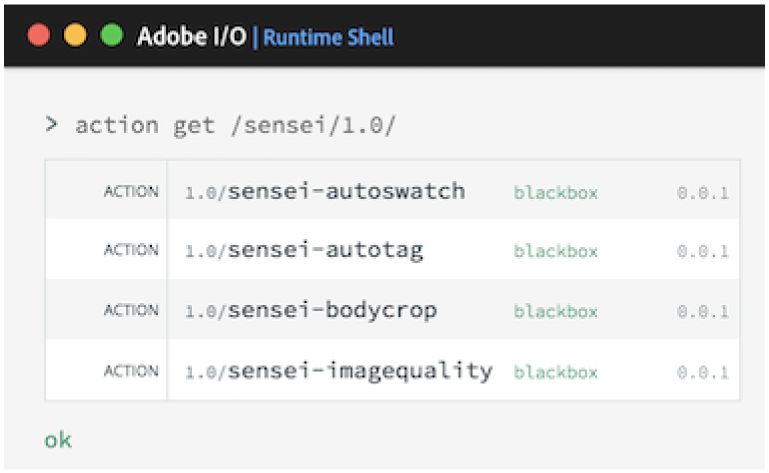
handler

65ms

This activation started **Thursday, March 1, 10:21:07 PM**

```
3/1/2018,   Handling event: {"event":{"xdmEventEnvelope:objectType":"xdmAsset","activitystrear
10:21:07    {"xdmAsset:format":"image/jpeg","xdmAsset:etag":"5abd4f7fba2d50db7d40b8d4b9d5b09c"
PM          0c1f8164df3c","xdmAsset:path":"/files/sensei-workflow/testcrop4.jpg","xdmAsset:ass
            {"@type":"xdmImsUser","xdmImsUser:id":"1BA21D504B8D3344992015A8@AdobeID"},"activit
            {"@type":"xdmContentRepository","xdmContentRepository:root":"https://cc-api-storag
            {"@type":"xdmImsUser","xdmImsUser:id":"1BA21D504B8D3344992015A8@AdobeID"},"@type":
            02T03:21:02.135Z","@id":"urn:oeid:sc:e53c0f1f-92e2-4814-b955-3e54f8462fc8"},"__ow_
            {"aem_port":"80","aem_user_password":"luc<yDingo89","aem_host":"52.191.166.198","c
            {"x-forwarded-port":"443","x-request-id":"XUt93VTWffIznjyzrhVwE72TNr9z97ZU","x-adc
            provider":"ccstorage","x-adobe-event-id":"82655b67-8692-43fb-945a-3c9754d61580","u
            newrelic-transaction":"PxRSUVICXQRRUQRaVAMPUlUBFB8EBw8RVU4aBwgMVVcEBAwECQI8WlAHB0N
            type":"application/json","x-real-ip":"10.64.221.153","connection":"close","accept-
            for":"34.199.65.18, 10.64.221.153","x-adobe-event-code":"asset_created"},"__ow_pat
            {"asset_id":"urn:aaid:sc:us:71194672-20a8-4f61-b656-0c1f8164df3c","url":"https://c
            0c1f8164df3c","pathname":"/files/sensei-
            workflow/testcrop4.jpg","user_id":"1BA21D504B8D3344992015A8@AdobeID","mime_type":"
            Creating trigger:cc_asset_created
            Triggering cc_asset_created
            Trigger cc_asset_created fired!
```

5. Before leaving the Adobe I/O Runtime Shell, execute the `action get /sensei/1.0/` command to view the names of the Adobe Sensei functions currently defined in your namespace. You should see a list like the following. These actions will be used throughout the rest of the exercises to add specific Adobe Sensei functions.



```
> action get /sensei/1.0/

    ACTION   1.0/sensei-autoswatch     blackbox   0.0.1
    ACTION   1.0/sensei-autotag        blackbox   0.0.1
    ACTION   1.0/sensei-bodycrop       blackbox   0.0.1
    ACTION   1.0/sensei-imagequality   blackbox   0.0.1

ok
```

Exercise 1: Adobe Sensei function: image quality

Overview

In this exercise you will use the **Adobe Sensei image quality** function to retrieve aesthetic scores for an image, including an overall quality score.

Steps

1. Open the **Visual Studio Code** application on your worksation located under the **Applications** folder (or in the dock toolbar).

2. Go to **File -> Open** and locate the folder for the exercises and solutions pre-loaded on your workstation in your user directory at `~/adobe-sensei-actions-lab`.

3. Begin by opening the `exercises/exercise-1/composition.js` file to learn about some specific concepts you'll need to understand for the remainder of the lab.

   `composer.sequence(task_1, task_2, ...)`

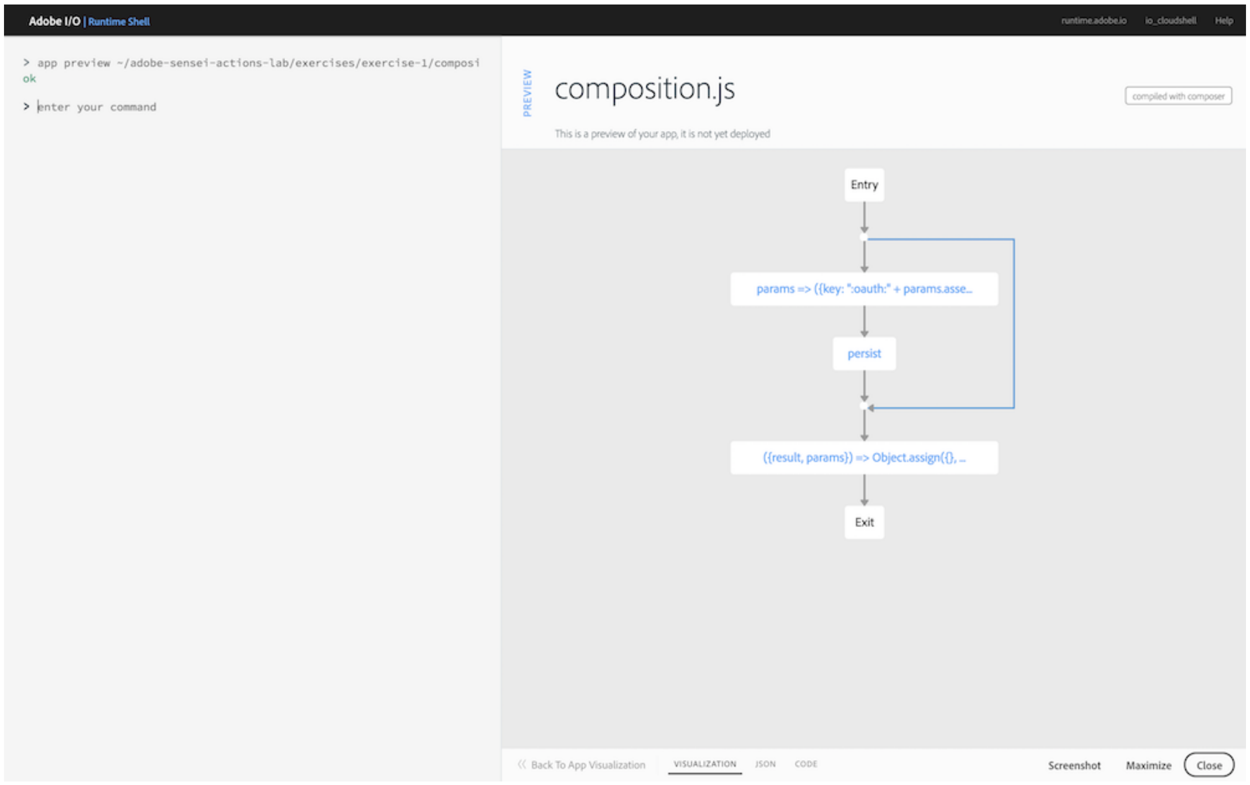
   > *Runs a sequence of tasks where the output parameter from the 1st task in the sequence is the input parameter for the next task.*

   `composer.retain(task)`

   > *The `retain` call is a parameter retention function that produces an output with two fields: `params` and `result` where `params` is the input parameter of the composition and `result` is the output of `task`.*

4. Before editing any code, go back into the **Adobe I/O Runtime Shell** application and preview the base composition flow with the following command:

   ```
   app preview ~/adobe-sensei-actions-lab/exercises/exercise-1/composition.js
   ```


Adobe I/O | Runtime Shell
runtime.adobe.io
io_cloudshell
Help
PREVIEW
composition.js
compiled with composer
This is a preview of your app, it is not yet deployed
Entry
params => ({key: ":oauth:" + params.asse...
persist
({result, params}) => Object.assign({}, ...
Exit
Back To App Visualization
VISUALIZATION
JSON
CODE
Screenshot
Maximize
Close

```
> app preview ~/adobe-sensei-actions-lab/exercises/exercise-1/composi
ok

> enter your command
```

5. Now switch back into your editor where the `exercises/exercise-1/composition.js` file is open, locate the `TODO` comment block and add the following:

```
/* TODO: Invoke the /sensei/1.0/sensei-imagequality action
 * passing the imageObject as parameter */
,composer.retain(
  composer.sequence(
    params => ({
      "image": params.imageObject
    }),
  '/sensei/1.0/sensei-imagequality'
  )
),
/* grab image quality results */
({result, params}) => Object.assign({}, result, params)
```

This code will:

   1. Invoke the `/sensei/1.0/sensei-imagequality` action with a parameter named `image` set to the `imageObject` parameter (previously defined in the base composition code and extracted from the asset created event).

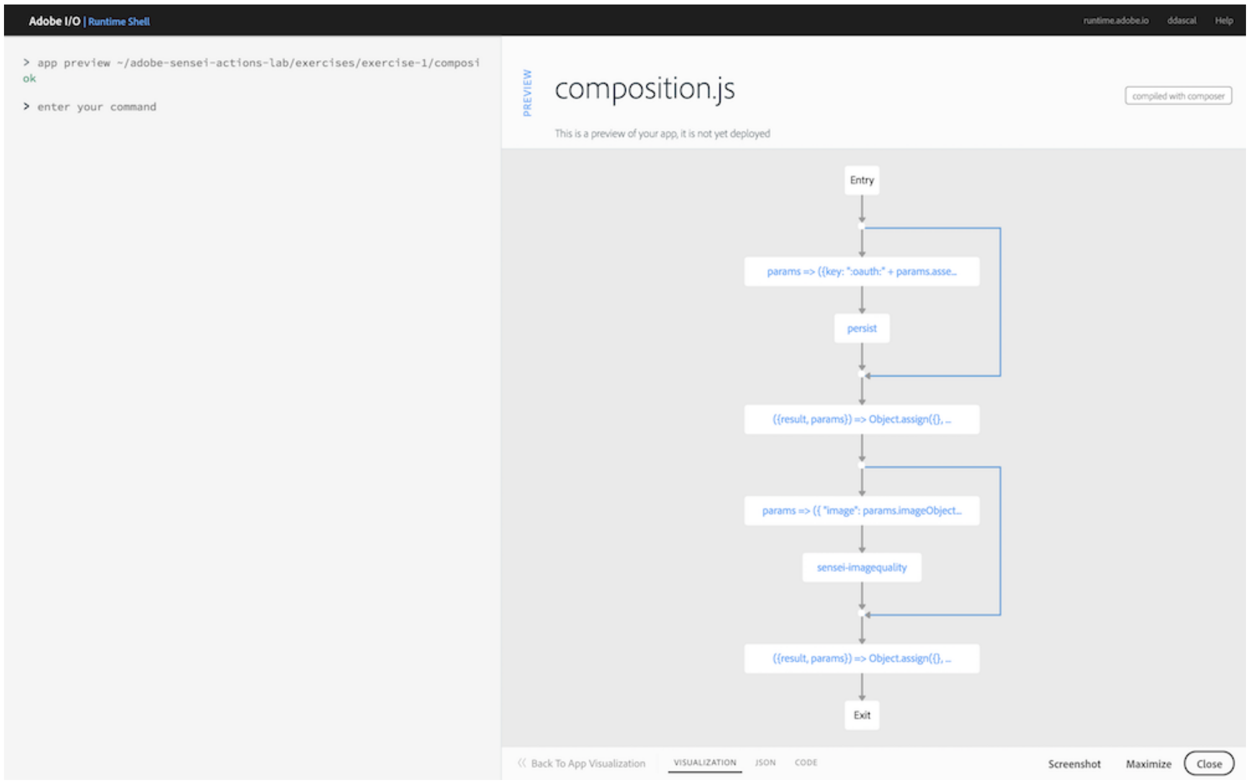
   2. Retain the result received from running the action for further use

Try it!

1. Switch back into the **Adobe I/O Runtime Shell** and preview your updated composition flow with your changes. Make sure it contains the `sensei-imagequality` action in the

visualization as shown below before moving on:

```
app preview ~/adobe-sensei-actions-lab/exercises/exercise-1/composition.js
```



```
> app preview ~/adobe-sensei-actions-lab/exercises/exercise-1/composi
ok

> enter your command
```

Adobe I/O | Runtime Shell
runtime.adobe.io
ddascal
Help
PREVIEW
composition.js
compiled with composer
This is a preview of your app, it is not yet deployed
Entry
params => ({key: ":oauth:" + params.asse...
persist
({result, params}) => Object.assign({}, ...
params => ({ "image": params.imageObject...
sensei-imagequality
({result, params}) => Object.assign({}, ...
Exit
Back To App Visualization
VISUALIZATION
JSON
CODE
Screenshot
Maximize
Close

> ***NOTE:*** *Your visualization will automatically update to preview your latest version if you had it running while making your code edits.*

2. Deploy the `asset_created_composition` with your updated version:

```
app update asset_created_composition ~/adobe-sensei-actions-lab/exercises/exercise-1/c
omposition.js
```

3. Now go back to the browser where your Creative Cloud instance is open and navigate into the new folder you created previously (ie: **sensei-lab-1**).

4. Drag and drop (or upload) an image from your workstation into your Creative Cloud folder to trigger an `asset_created` event.

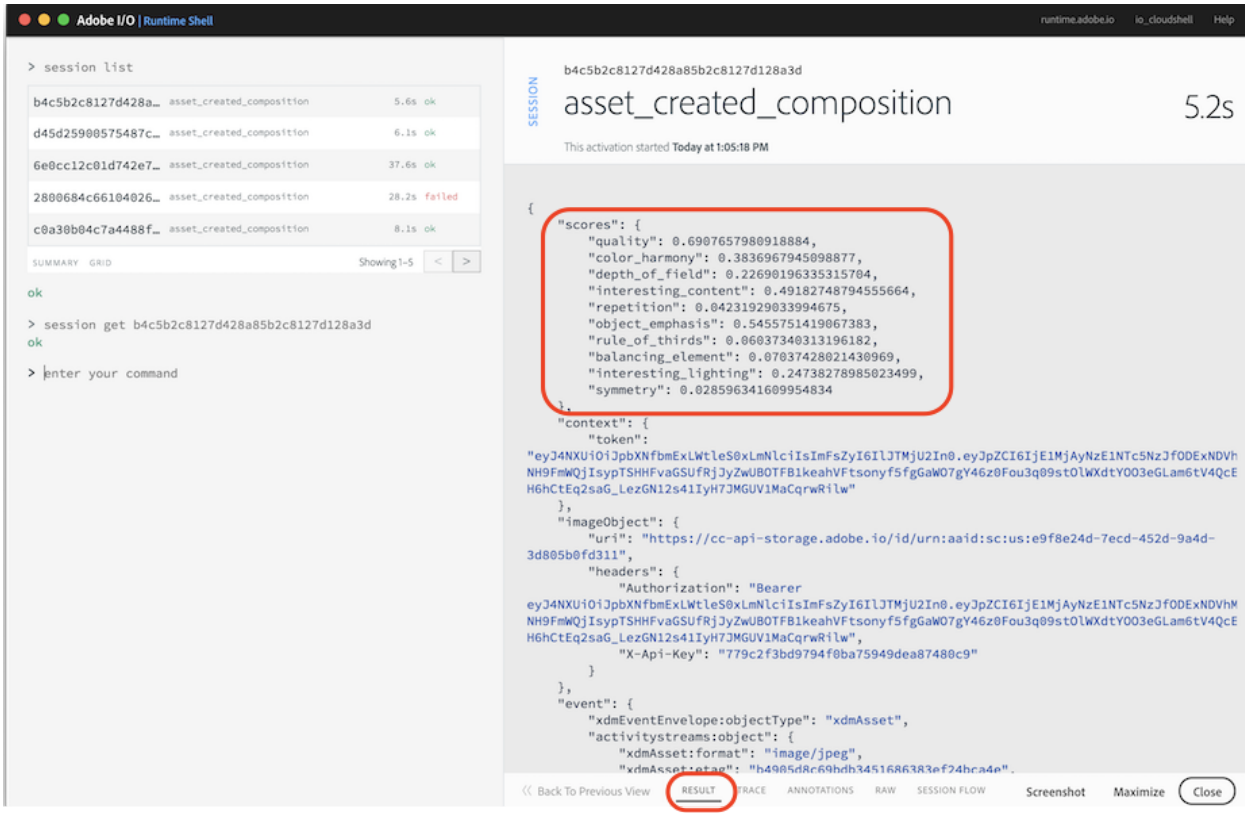
> ***NOTE:*** *There are images provided for testing within your `~/adobe-sensei-actions-lab/stock-photos` folder. For instance, drag and drop `image1.png` and look at the resulting quality scores.*

5. Switch back to the **Adobe I/O Runtime Shell** application and run the `session list` command to list all of the current sessions:

```
session list
```

6. Locate the most recent `asset_created_composition` running and click on the session id to view the session details.

   The `RESULT` tab displays the results in JSON format. If the Sensei image quality action ran successfully, you should see an element in the JSON named `scores`, with values assigned for attributes further explained below. The `quality` value is an overall score based on the rest of the individual attributes, and ranges between 0-1 where a higher quality image results in a higher score.



```
> session list
b4c5b2c8127d428a… asset_created_composition   5.6s ok
d45d25900575487c… asset_created_composition   6.1s ok
6e0cc12c01d742e7… asset_created_composition  37.6s ok
2800684c66104026… asset_created_composition  28.2s failed
c0a30b04c7a4488f… asset_created_composition   8.1s ok
ok

> session get b4c5b2c8127d428a85b2c8127d128a3d
ok

> enter your command
```

b4c5b2c8127d428a85b2c8127d128a3d

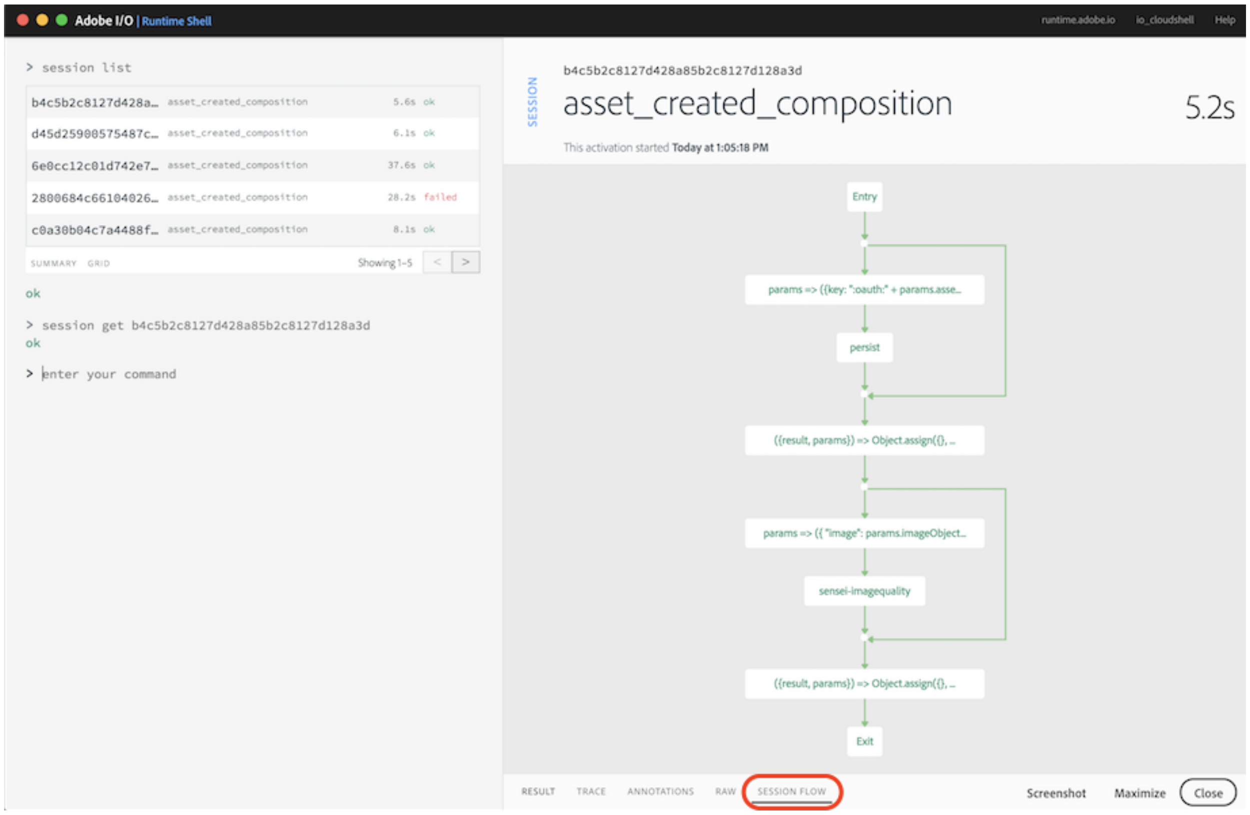
asset_created_composition 5.2s

This activation started **Today at 1:05:18 PM**

```
{
    "scores": {
        "quality": 0.6907657980918884,
        "color_harmony": 0.3836967945098877,
        "depth_of_field": 0.22690196335315704,
        "interesting_content": 0.49182748794555664,
        "repetition": 0.04231929033994675,
        "object_emphasis": 0.5455751419067383,
        "rule_of_thirds": 0.06037340313196182,
        "balancing_element": 0.07037428021430969,
        "interesting_lighting": 0.24738278985023499,
        "symmetry": 0.028596341609954834
    },
    "context": {
        "token":
"eyJ4NXUiOiJpbXNfbmExLWtleS0xLmNlciIsImFsZyI6IlJTMjU2In0.eyJpZCI6IjE1MjAyNzE1NTc5NzJfODExNDVh
NH9FmWQjIsypTSHHFvaGSUfRjJyZwUBOTFB1keahVFtsonyf5fgGaWO7gY46z0Fou3q09stOlWXdtYOO3eGLam6tV4QcE
H6hCtEq2saG_LezGN12s41IyH7JMGUV1MaCqrwRilw"
    },
    "imageObject": {
        "uri": "https://cc-api-storage.adobe.io/id/urn:aaid:sc:us:e9f8e24d-7ecd-452d-9a4d-
3d805b0fd311",
        "headers": {
            "Authorization": "Bearer
eyJ4NXUiOiJpbXNfbmExLWtleS0xLmNlciIsImFsZyI6IlJTMjU2In0.eyJpZCI6IjE1MjAyNzE1NTc5NzJfODExNDVhM
NH9FmWQjIsypTSHHFvaGSUfRjJyZwUBOTFB1keahVFtsonyf5fgGaWO7gY46z0Fou3q09stOlWXdtYOO3eGLam6tV4QcE
H6hCtEq2saG_LezGN12s41IyH7JMGUV1MaCqrwRilw",
            "X-Api-Key": "779c2f3bd9794f0ba75949dea87480c9"
        }
    },
    "event": {
        "xdmEventEnvelope:objectType": "xdmAsset",
        "activitystreams:object": {
            "xdmAsset:format": "image/jpeg",
            "xdmAsset:etag": "b4905d8c69bdb3451686383ef24bca4e",
```

- **Quality** – The overall score
- **Balancing Element** – How balanced the overall composition is
- **Color Harmony** – How balanced the colors are
- **Interesting Content** – Whether or not the image contains interesting content
- **Interesting Lighting** – Colorful-ness in the lighting – night scene, sunset, interesting lighting effects…
- **Repetition** – Whether there's repetition (pattern, composition…) in the photo
- **Symmetry** – Based on symmetry in composition and object
- **Depth of Field** – https://en.wikipedia.org/wiki/Depth_of_field
- **Object Emphasis** – Detects if there's a main subject
- **Rule of Thirds** – https://en.wikipedia.org/wiki/Rule_of_thirds
- **Vivid Color** – Saturation of color and visual reception

2. Next, click on the **SESSION FLOW** tab. If your code ran successfully you should see something similar to below, where all actions completed successfully and are denoted in green:



```
> session list
b4c5b2c8127d428a…  asset_created_composition   5.6s  ok
d45d25900575487c…  asset_created_composition   6.1s  ok
6e0cc12c01d742e7…  asset_created_composition  37.6s  ok
2800684c66104026…  asset_created_composition  28.2s  failed
c0a30b04c7a4488f…  asset_created_composition   8.1s  ok
ok

> session get b4c5b2c8127d428a85b2c8127d128a3d
ok

> enter your command
```

b4c5b2c8127d428a85b2c8127d128a3d

asset_created_composition 5.2s

This activation started **Today at 1:05:18 PM**

Adobe I/O | Runtime Shell
runtime.adobe.io
io_cloudshell
Help
SUMMARY
GRID
Showing 1–5
SESSION
Entry
params => ({key: ":oauth:" + params.asse…
persist
({result, params}) => Object.assign({}, …
params => ({ "image": params.imageObject…
sensei-imagequality
({result, params}) => Object.assign({}, …
Exit
RESULT
TRACE
ANNOTATIONS
RAW
SESSION FLOW
Screenshot
Maximize
Close

Exercise 2: Manual image processing

Overview

In this exercise you will add code to check the overall quality score received from the previous **Adobe Sensei image quality** function. When the quality score does not meet an acceptable range, you will call an action to upload it back to Creative Cloud for manual processing.

Steps

1. In Visual Studio Code, open the `exercises/exercise-2/composition.js` file.

2. Locate the `TODO` comment block and add the following snippet:

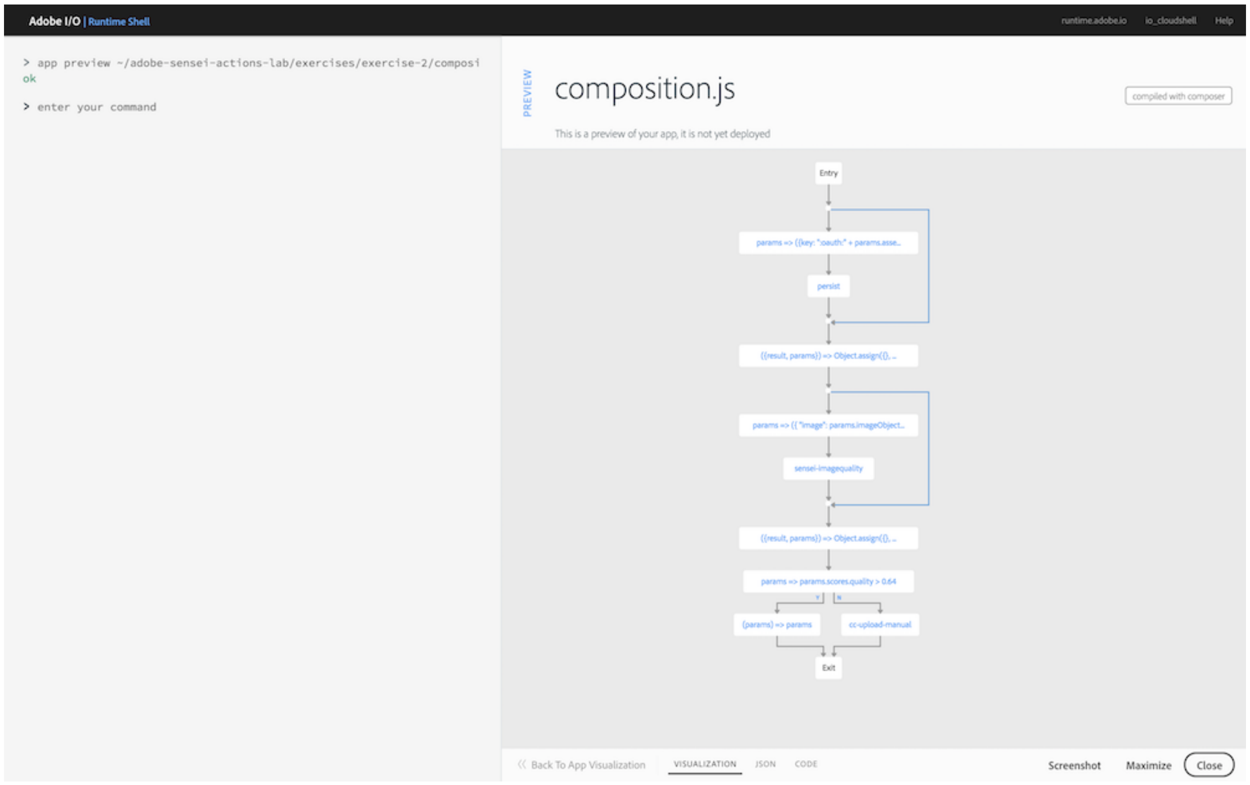
   ```
   /* TODO: ... */
   composer.if(
     params => params.scores.quality > 0.64,
     /* if quality is met, process the image and upload to Adobe Experience Manager */
     (params) => params,
     /* if quality is NOT met, copy asset to manual process folder in CC */
    '/adobe/acp-assets-0.5.0/cc-upload-manual')
   ```

   > ***HINT:** Look back to the results of the previous exercise to remind you of the `quality` score returned for an image.*

Try it!

1. First preview your composition again to ensure your new changes are shown before moving on:

   ```
   app preview ~/adobe-sensei-actions-lab/exercises/exercise-2/composition.js
   ```



```
> app preview ~/adobe-sensei-actions-lab/exercises/exercise-2/composi
ok

> enter your command
```

Adobe I/O | Runtime Shell
runtime.adobe.io
io_cloudshell
Help
PREVIEW
composition.js
compiled with composer
This is a preview of your app, it is not yet deployed
Entry
params => ({key: ":oauth:" + params.asse…
persist
({result, params}) => Object.assign({}, …
params => ({ "image": params.imageObject…
sensei-imagequality
({result, params}) => Object.assign({}, …
params => params.scores.quality > 0.64
Y
N
(params) => params
cc-upload-manual
Exit
Back To App Visualization
VISUALIZATION
JSON
CODE
Screenshot
Maximize
Close

2. Next, update the currently deployed version of the `asset_created_composition` with your edited version:

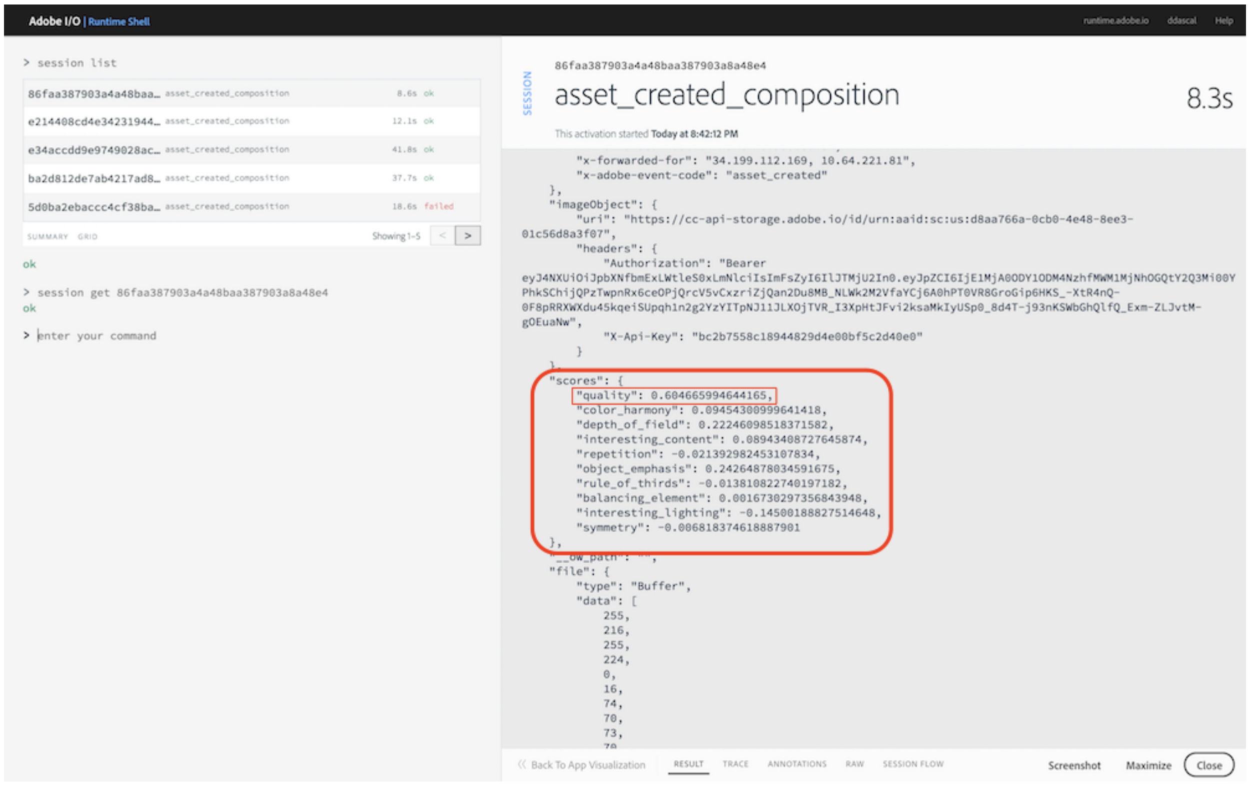
```
app update asset_created_composition ~/adobe-sensei-actions-lab/exercises/exercise-2/c
omposition.js
```

3. Open the browser to your Creative Cloud instance and trigger the `asset_created` event by uploading a low quality image into it. One has been included for you within the projects `~/adobe-sensei-actions-lab/stock-photos` folder by the name of `quality-fail.png` .

4. Switch back to the **Adobe I/O Runtime Shell** to find your session by running the following command:

```
session list
```

5. Locate the most recent `asset_created_composition` running and click on the session id to view the result. When it was run successfully the **RESULTS** tab JSON will show the scores indicating the quality value is indeed less than **0.64**:



```
> session list
86faa387903a4a48baa… asset_created_composition    8.6s ok
e214408cd4e34231944… asset_created_composition   12.1s ok
e34accdd9e9749028ac… asset_created_composition   41.8s ok
ba2d812de7ab4217ad8… asset_created_composition   37.7s ok
5d0ba2ebaccc4cf38ba… asset_created_composition   18.6s failed
SUMMARY  GRID                              Showing 1-5  <  >
ok

> session get 86faa387903a4a48baa387903a8a48e4
ok

> enter your command
```

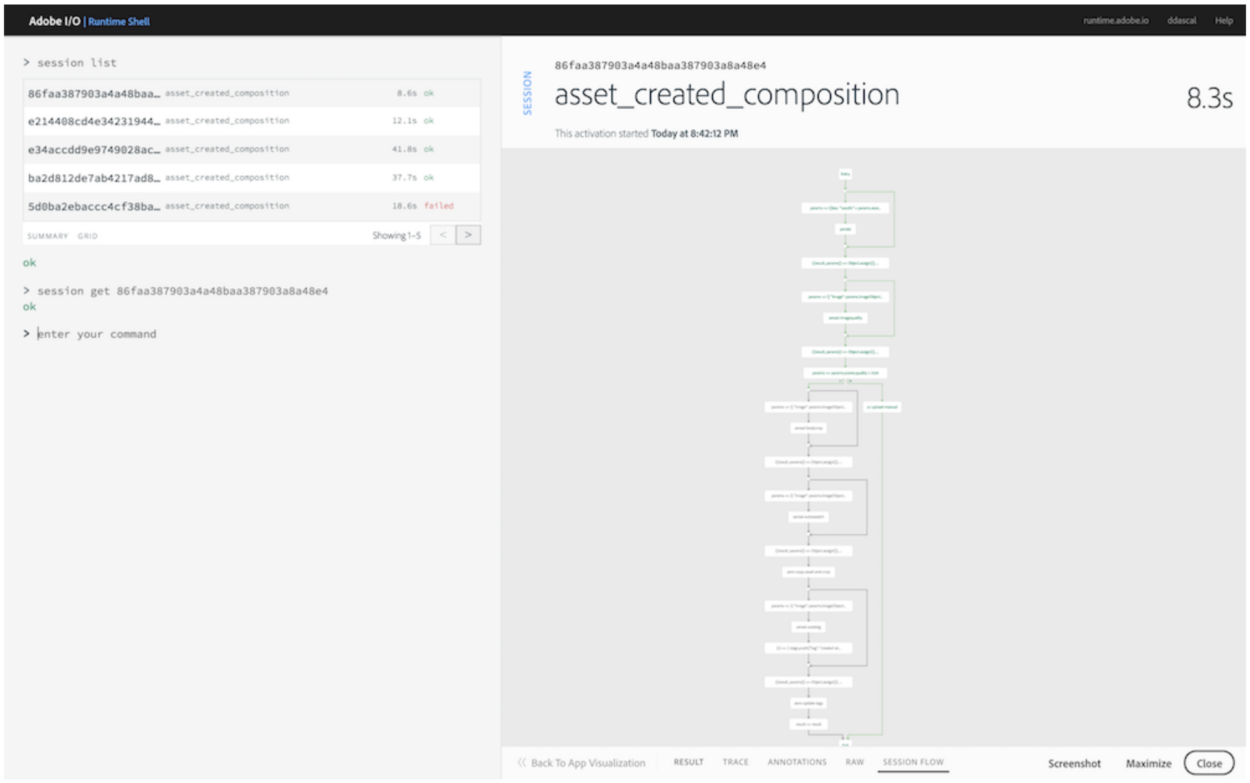
86faa387903a4a48baa387903a8a48e4

asset_created_composition 8.3s

This activation started **Today at 8:42:12 PM**

```
        "x-forwarded-for": "34.199.112.169, 10.64.221.81",
        "x-adobe-event-code": "asset_created"
    },
    "imageObject": {
        "uri": "https://cc-api-storage.adobe.io/id/urn:aaid:sc:us:d8aa766a-0cb0-4e48-8ee3-
01c56d8a3f07",
        "headers": {
            "Authorization": "Bearer
eyJ4NXUiOiJpbXNfbmExLWtleS0xLmNlciIsImFsZyI6IlJTMjU2In0.eyJpZCI6IjE1MjA0ODY1ODM4NzhfMWM1MjNhOGQtY2Q3Mi00Y
PhkSChijQPzTwpnRx6ceOPjQrcV5vCxzriZjQan2Du8MB_NLWk2M2VfaYCj6A0hPT0VR8GroGip6HKS_-XtR4nQ-
0F8pRRXWXdu45kqeiSUpqh1n2g2YzYITpNJ11JLXOjTVR_I3XpHtJFvi2ksaMkIyUSp0_8d4T-j93nKSWbGhQlfQ_Exm-ZLJvtM-
gOEuaNw",
            "X-Api-Key": "bc2b7558c18944829d4e00bf5c2d40e0"
        }
    },
    "scores": {
        "quality": 0.604665994644165,
        "color_harmony": 0.09454300999641418,
        "depth_of_field": 0.22246098518371582,
        "interesting_content": 0.08943408727645874,
        "repetition": -0.021392982453107834,
        "object_emphasis": 0.24264878034591675,
        "rule_of_thirds": -0.013810822740197182,
        "balancing_element": 0.0016730297356843948,
        "interesting_lighting": -0.14500188827514648,
        "symmetry": -0.006818374618887901
    },
    "__ow_path": "",
    "file": {
        "type": "Buffer",
        "data": [
            255,
            216,
            255,
            224,
            0,
            16,
            74,
            70,
            73,
```

Also note how the **SESSION FLOW** tab at the bottom shows the path ended with the `cc-upload-manual` action:



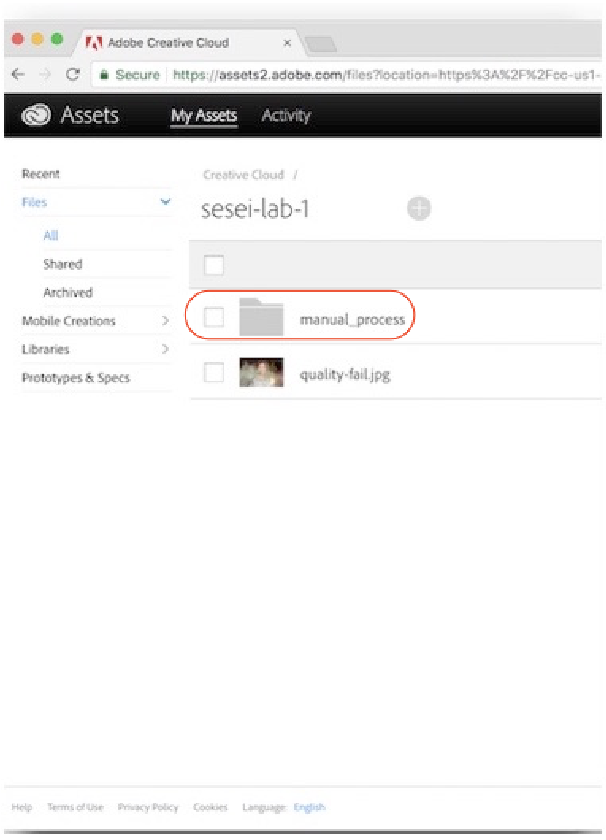
```
> session list
86faa387903a4a48baa… asset_created_composition    8.6s ok
e214408cd4e34231944… asset_created_composition   12.1s ok
e34accdd9e9749028ac… asset_created_composition   41.8s ok
ba2d812de7ab4217ad8… asset_created_composition   37.7s ok
5d0ba2ebaccc4cf38ba… asset_created_composition   18.6s failed
SUMMARY  GRID                              Showing 1-5  <  >
ok

> session get 86faa387903a4a48baa387903a8a48e4
ok

> enter your command
```

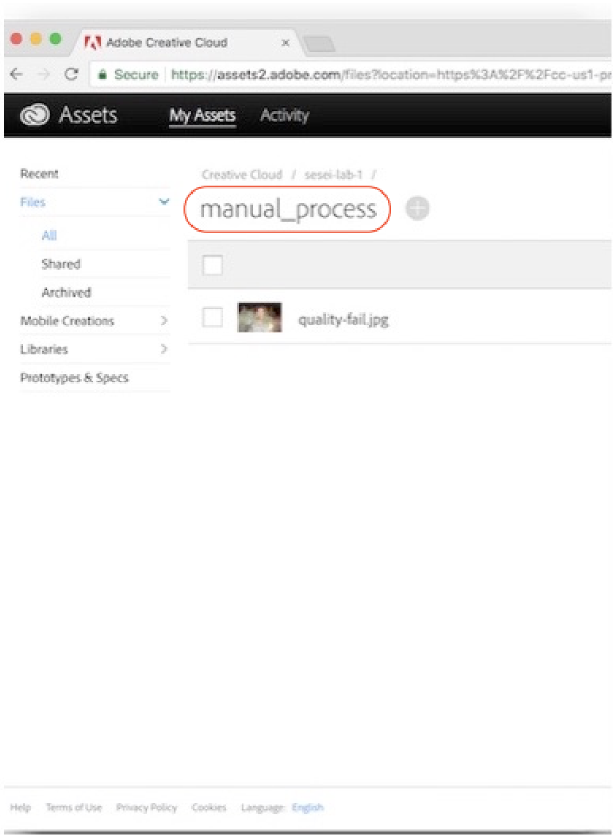
86faa387903a4a48baa387903a8a48e4

asset_created_composition 8.3s

This activation started **Today at 8:42:12 PM**

6. Now go back into your Creative Cloud instance and you should see that a new folder was created called `manual` and contains the `quality-fail.png` file.


Adobe Creative Cloud
Secure
https://assets2.adobe.com/files?location=https%3A%2F%2Fcc-us1-
Assets
My Assets
Activity
Recent
Files
All
Shared
Archived
Mobile Creations
Libraries
Prototypes & Specs
Creative Cloud /
sesei-lab-1
manual_process
quality-fail.jpg
Help
Terms of Use
Privacy Policy
Cookies
Language: English


Adobe Creative Cloud
Secure
https://assets2.adobe.com/files?location=https%3A%2F%2Fcc-us1-p
Assets
My Assets
Activity
Recent
Files
All
Shared
Archived
Mobile Creations
Libraries
Prototypes & Specs
Creative Cloud / sesei-lab-1 /
manual_process
quality-fail.jpg
Help
Terms of Use
Privacy Policy
Cookies
Language: English

Exercise 3: Adobe Sensei function: body crop

Overview

In this exercise you will use the **Adobe Sensei body crop** function to identify reference points in the provided image to use for cropping out the body automatically.

Steps

1. In Visual Studio Code, open `exercises/exercise-3/composition.js`.

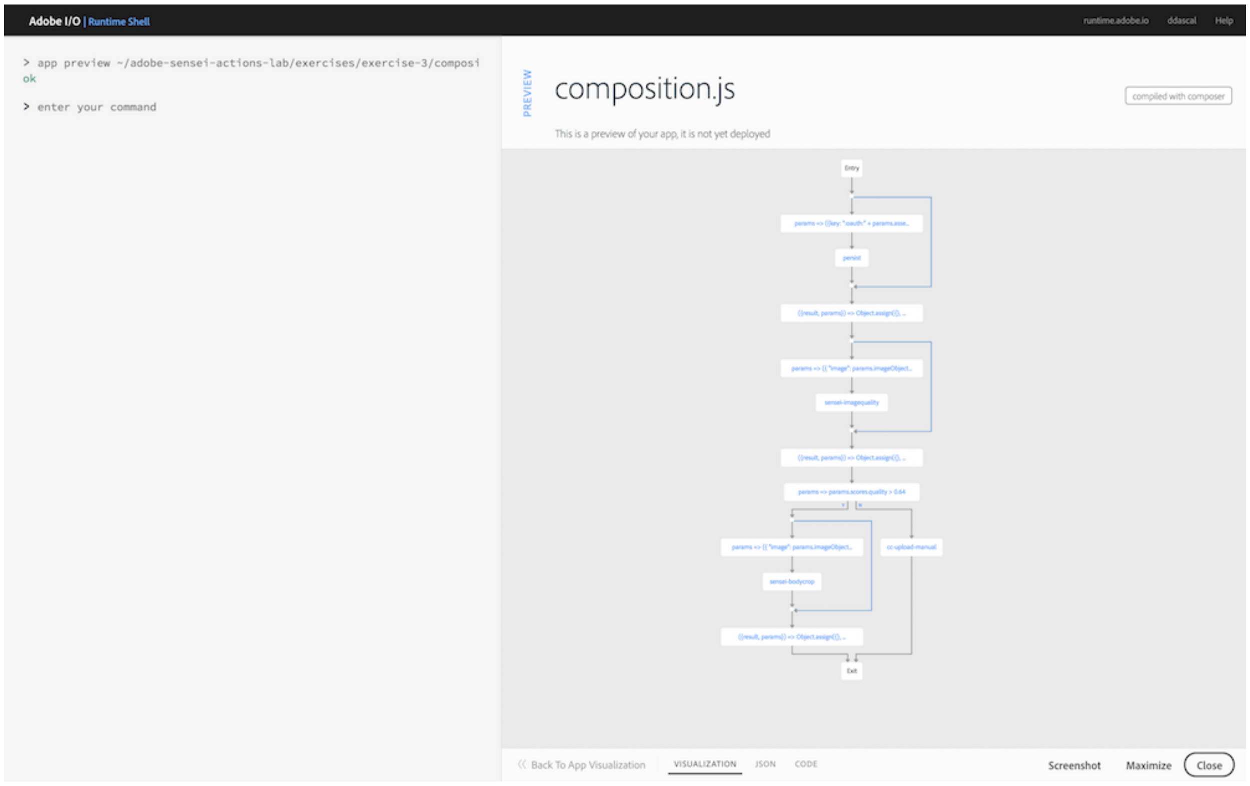
2. Just after the `TODO` block, add the code snippet below to invoke the `/sensei/1.0/sensei-bodycrop` action. The code will pass in the `image` object and save the result in a `crops` object. The result will contain the constraints recommended for cropping out the body of the image.

```
/**
 *  TODO: Use the action '/sensei/1.0/sensei-bodycrop' to crop the body.
 */
composer.retain(
    composer.sequence(
        params => ({
            "image": params.imageObject
        }),
    '/sensei/1.0/sensei-bodycrop'
    )
),
/* grab bodycrop results */
({result, params}) => Object.assign({},
    { crops: result },
    params
)
```

Try it!

1. First, preview your composition to ensure it contains the `sensei-bodycrop` action:

```
app preview ~/adobe-sensei-actions-lab/exercises/exercise-3/composition.js
```



```
> app preview ~/adobe-sensei-actions-lab/exercises/exercise-3/composi
ok
> enter your command
```

Adobe I/O | Runtime Shell
runtime.adobe.io
ddascal
Help
PREVIEW
composition.js
compiled with composer
This is a preview of your app, it is not yet deployed
Entry
params => ({key: "oauth:" + params.asse...
persist
({result, params}) => Object.assign({}, ...
params => ({ "image": params.imageObject...
sensei-imagequality
({result, params}) => Object.assign({}, ...
params => params.scores.quality > 0.64
Y
N
params => ({ "image": params.imageObject...
sensei-bodycrop
({result, params}) => Object.assign({}, ...
cc-upload-manual
Exit
Back To App Visualization
VISUALIZATION
JSON
CODE
Screenshot
Maximize
Close

2. Next update the current `asset_created_composition` app with your new version:

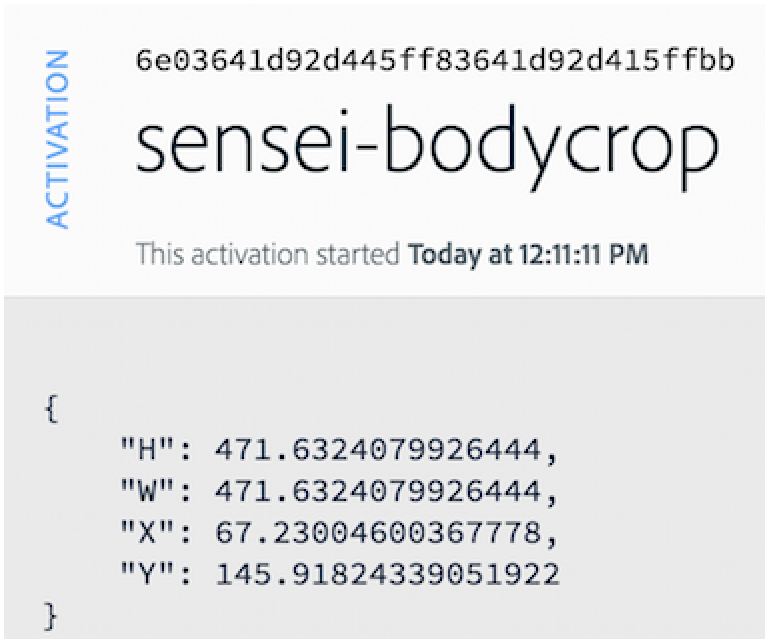
```
app update asset_created_composition ~/adobe-sensei-actions-lab/exercises/exercise-3/c
omposition.js
```

3. Now open the browser to your Creative Cloud folder and upload a new image to trigger an `asset_created` event. For instance, try `image2.png` from the `~/adobe-sensei-actions-lab/stock-photos` folder.

4. Switch back to the **Adobe I/O Runtime Shell** and type:

```
session list
```

5. Locate the most recent `asset_created_composition` running and click on the session id to view the result. The response should contain the coordinates to crop, like shown below:



6e03641d92d445ff83641d92d415ffbb

sensei-bodycrop

This activation started **Today at 12:11:11 PM**

```
{
    "H": 471.6324079926444,
    "W": 471.6324079926444,
    "X": 67.23004600367778,
    "Y": 145.91824339051922
}
```

Exercise 4: Adobe Sensei function: auto swatch + Adobe Experience Manager Copy/Crop

Overview

In this exercise you will add the **Adobe Sensei auto swatch** function to automatically extract color swatches from your image, then call an Adobe Experience Manager action that will crop and create swatches from your image, then copy them into Adobe Experience Manager.

Steps

1. In Visual Studio Code, open the `exercises/exercise-4/composition.js` file.

2. Just after the `TODO` block, add the following code to invoke the `/sensei/1.0/sensei-autoswatch` action to crop the body of the image. The parameters you're passing are:

    - `image` – the image object to use for swatches
    - `results` – max number of swatches to return
    - `size` – size of the swatch (or 0)

    ```
    /**
     * TODO: Invoke '/sensei/1.0/sensei-autoswatch'
     */
    composer.retain(
      composer.sequence(
        params => ({
          "image": params.imageObject,
          "results": 2,
          "size": 0
        }),
      '/sensei/1.0/sensei-autoswatch'
      )
    ),
    /* grab autoswatch results */
    ({result, params}) => Object.assign({}, result, params),
    ```

3. Now that the image quality has been checked and the crop and swatch results returned, the image is ready to be copied into Adobe Experience Manager for final distribution.

    After the `sensei-autoswatch` results are returned, code the following action to perform the crops on the image and copy it into Adobe Experience Manager:

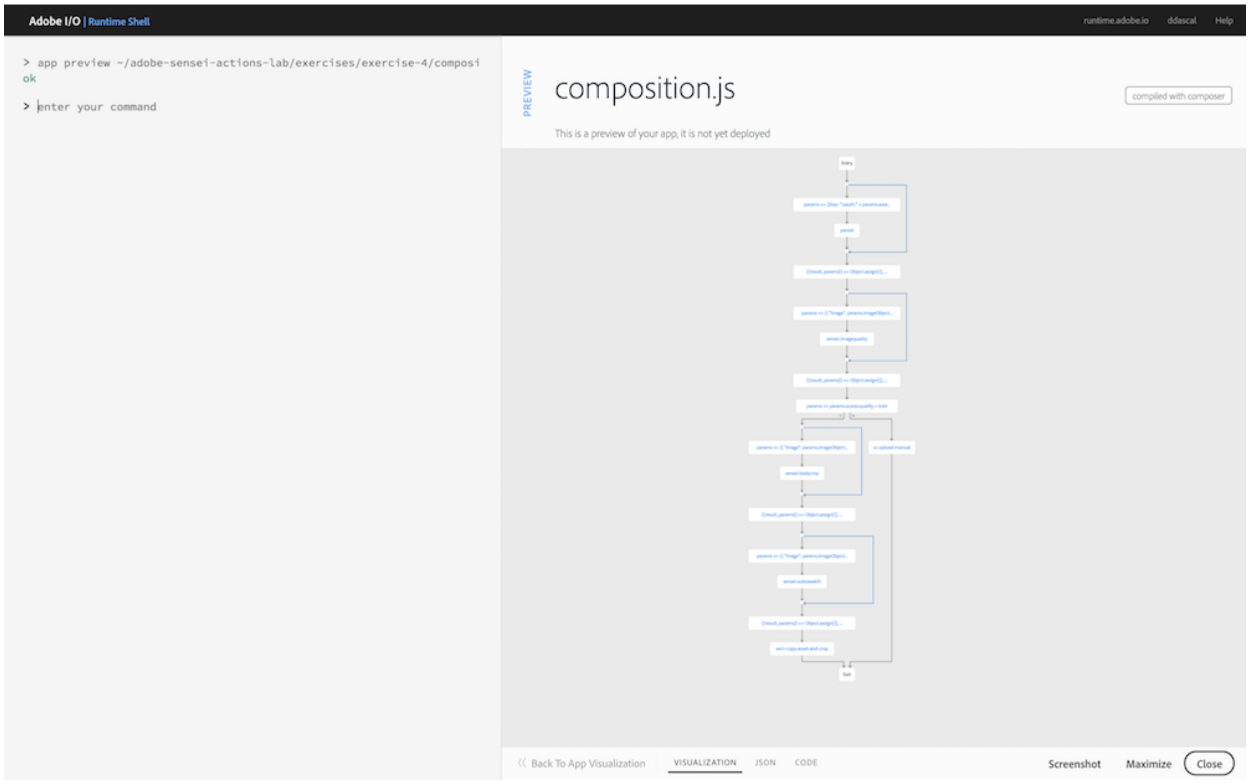
```
    /**
     *  TODO: Copy asset to Adobe Experience Manager
     *  invoking '/adobe/acp-assets-0.5.0/aem-copy-asset-and-crop' action
     */
    '/adobe/acp-assets-0.5.0/aem-copy-asset-and-crop'
```

> ***NOTE:*** *This action will also perform the actual cropping and swatch creation from the image prior to copying to Adobe Experience Manager.*

Try it!

1. First, preview your composition again to ensure your new changes are shown:

```
app preview ~/adobe-sensei-actions-lab/exercises/exercise-4/composition.js
```



```
> app preview ~/adobe-sensei-actions-lab/exercises/exercise-4/composi
ok

> enter your command
```

2. Next, update the current `asset_created_composition` with your new version:

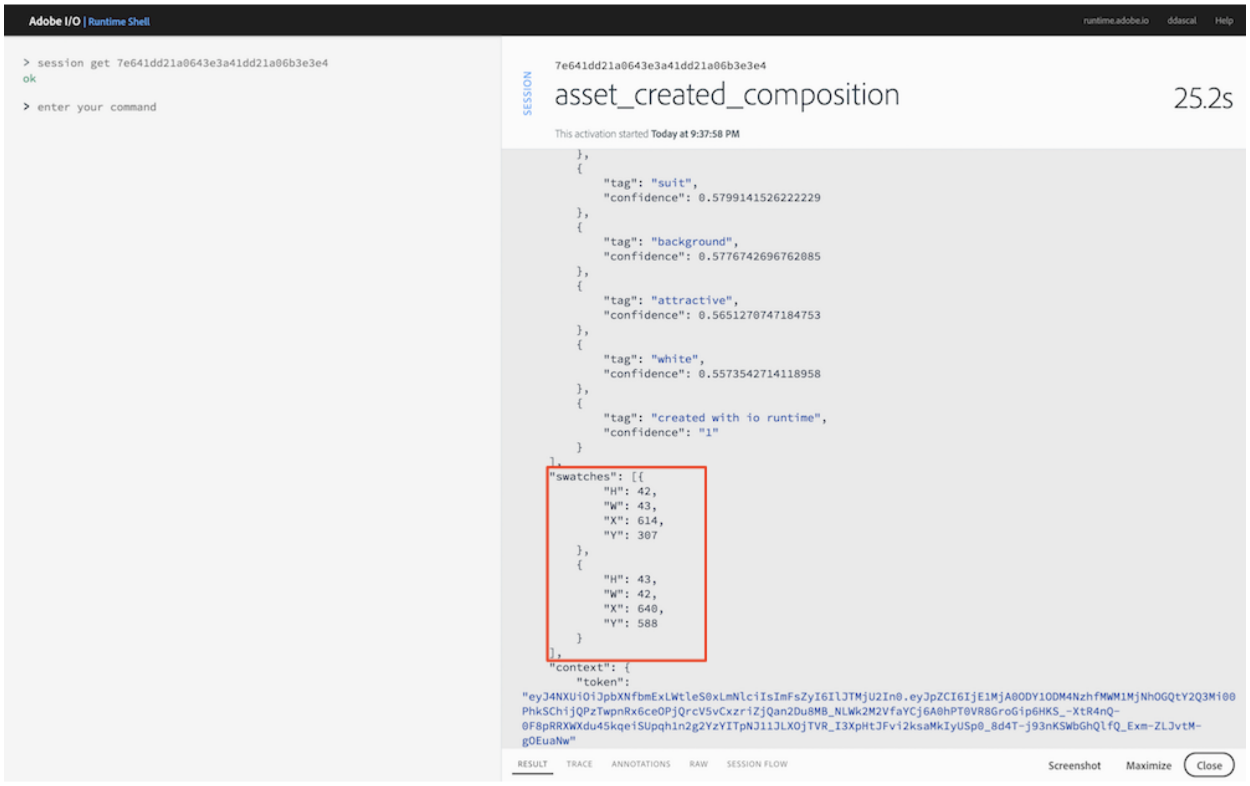
```
app update asset_created_composition ~/adobe-sensei-actions-lab/exercises/exercise-4/c
omposition.js
```

3. Now open the browser to your Creative Cloud and upload an image to trigger an `asset_created` event.

4. Switch back to the **Adobe I/O Runtime Shell** and type:

```
session list
```

5. Locate the most recent `asset_created_composition` running and click on the session id to view the result. If you scroll down in the results JSON you should see the swatch results that were generated like shown in the session below:



```
> session get 7e641dd21a0643e3a41dd21a06b3e3e4
ok

> enter your command
```

7e641dd21a0643e3a41dd21a06b3e3e4

asset_created_composition 25.2s

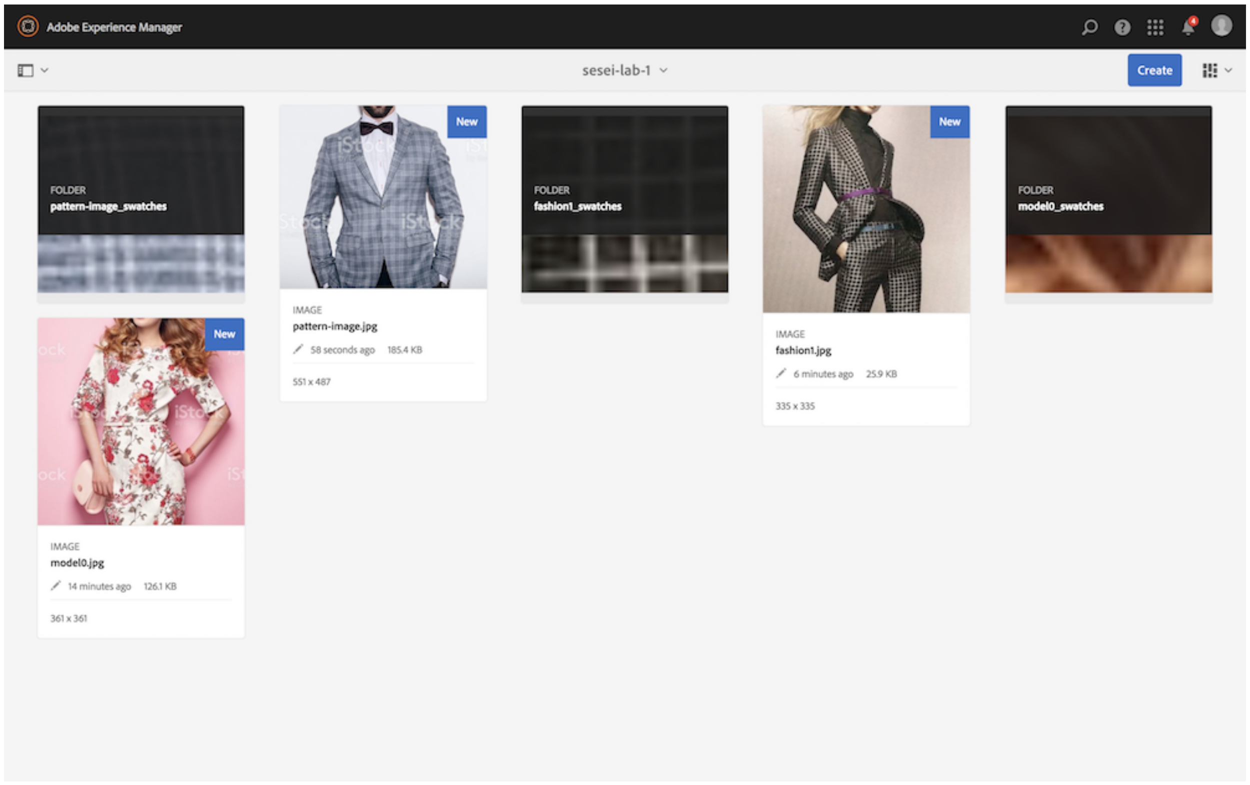
This activation started Today at 9:37:58 PM

```
        },
        {
            "tag": "suit",
            "confidence": 0.5799141526222229
        },
        {
            "tag": "background",
            "confidence": 0.5776742696762085
        },
        {
            "tag": "attractive",
            "confidence": 0.5651270747184753
        },
        {
            "tag": "white",
            "confidence": 0.5573542714118958
        },
        {
            "tag": "created with io runtime",
            "confidence": "1"
        }
    ],
    "swatches": [{
            "H": 42,
            "W": 43,
            "X": 614,
            "Y": 307
        },
        {
            "H": 43,
            "W": 42,
            "X": 640,
            "Y": 588
        }
    ],
    "context": {
        "token":
"eyJ4NXUiOiJpbXNfbmExLWtleS0xLmNlciIsImFsZyI6IlJTMjU2In0.eyJpZCI6IjE1MjA0ODY1ODM4NzhfMWM1MjNhOGQtY2Q3Mi00
PhkSChijQPzTwpnRx6ceOPjQrcV5vCxzriZjQan2Du8MB_NLWk2M2VfaYCj6A0hPT0VR8GroGip6HKS_-XtR4nQ-
0F8pRRXWXdu45kqeiSUpqh1n2g2YzYITpNJ11JLXOjTVR_I3XpHtJFvi2ksaMkIyUSp0_8d4T-j93nKSWbGhQlfQ_Exm-ZLJvtM-
gOEuaNw"
```

6. If the app ran successfully then you will see the asset copied into an AEM instance (with the same folder name as you defined in the Creative Cloud).

> ***NOTE:*** *Use the Adobe Experience Manager host and credentials provided to login and check Adobe Experience Manager for the file.*

Notice the images now have the face cropped out and a new folder has been created to hold the swatches for the image sized based on the sensei-swatch action results:


Adobe Experience Manager

sesei-lab-1

Create

FOLDER
pattern-image_swatches

New

IMAGE
pattern-image.jpg
58 seconds ago 185.4 KB
551 x 487

FOLDER
fashion1_swatches

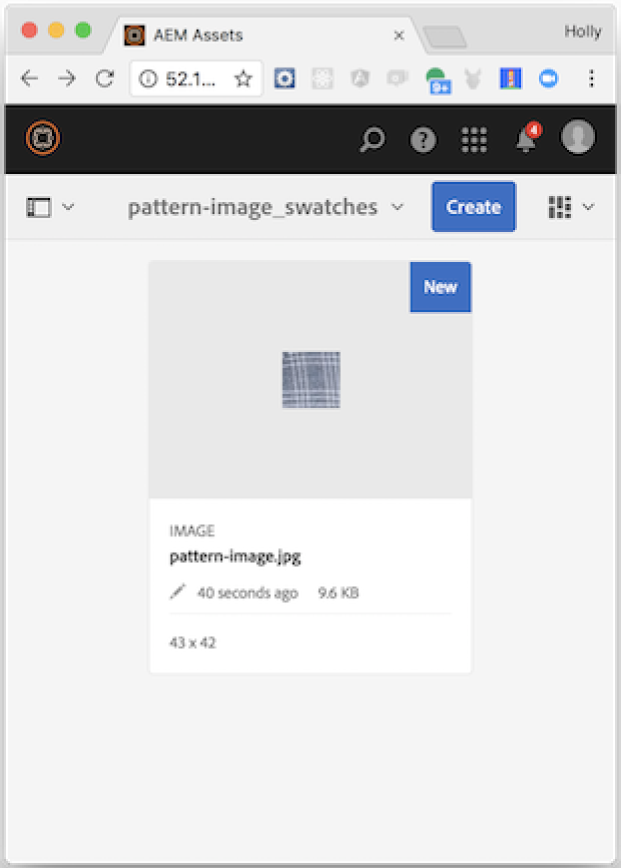
New

IMAGE
fashion1.jpg
6 minutes ago 25.9 KB
335 x 335

FOLDER
model0_swatches

New

IMAGE
model0.jpg
14 minutes ago 126.1 KB
361 x 361


AEM Assets

Holly

52.1...

pattern-image_swatches

Create

New

IMAGE
pattern-image.jpg
40 seconds ago 9.6 KB
43 x 42

Exercise 5: Adobe Sensei function: auto tag + Adobe Experience Manager Update Tags

Overview

In this exercise you will add the **Adobe Sensei auto tag** function to auto tag an image followed by the Adobe Experience Manager Update Tags action to update the tags for the image in Adobe Experience Manager.

Steps

1. In Visual Studio Code, open the `exercises/exercise-5/composition.js` file.
2. Just after the `TODO` block, begin adding code to call the `/sensei/1.0/sensei-autotag` action to predict tags for the image. The parameters you'll need to pass are:
    - `image` – the image object to use for swatches
    - `confidence` – level of confidence from 0–1 where 1 is the highest confidence
    - `results` – number of tags to return

    ```
    /**
     * TODO: Autotag the image invoking '/sensei/1.0/sensei-autotag' action.
     */
    composer.retain(
      composer.sequence(
        params => ({
          "image": params.imageObject,
          "confidence": 0.5,
          "results": 10
        }),
      '/sensei/1.0/sensei-autotag',
      (r) => { r.tags.push({"tag": "created with io runtime", "confidence":"1"}); retu
    rn r; }
      )
    ),
    /* grab autotag results */
    ({result, params}) => Object.assign({}, result, params),
    ```

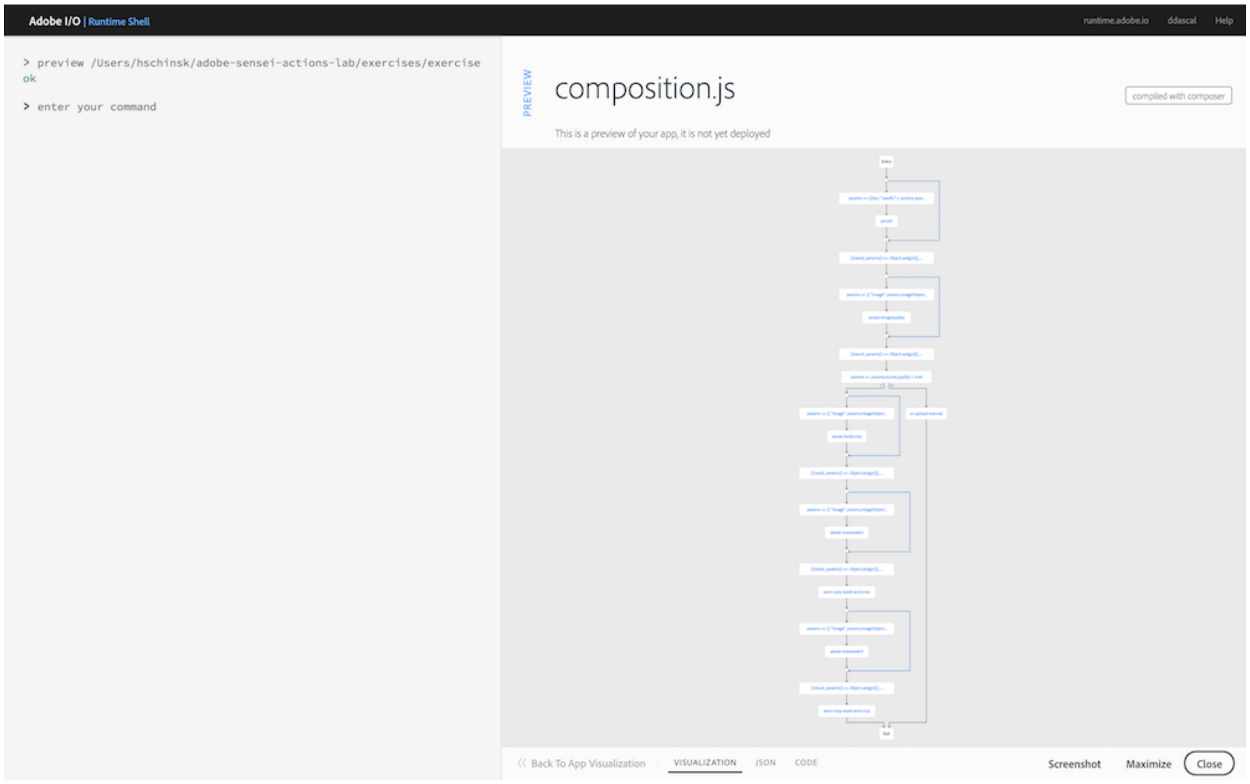
3. After the tags have been assigned, the image needs to be updated in Adobe Experience Manager to include them. After the autotag result has been returned, add the following code to call an action to update the tags for the image in Adobe Experience Manager:

```
        /**
          * TODO: Update the tags in Adobe Experience Manager Assets
          *  by invoking '/adobe/acp-assets-0.5.0/aem-update-tags'
          */
          '/adobe/acp-assets-0.5.0/aem-update-tags',
          /**
          * Return the result as-is
          */
          result => result
        ),
```

Try it!

1. First, preview your composition again to ensure your new changes are shown:

```
app preview ~/adobe-sensei-actions-lab/exercises/exercise-5/composition.js
```



```
> preview /Users/hschinsk/adobe-sensei-actions-lab/exercises/exercise
ok

> enter your command
```

2. Next, update the current `asset_created_composition` with your new version:

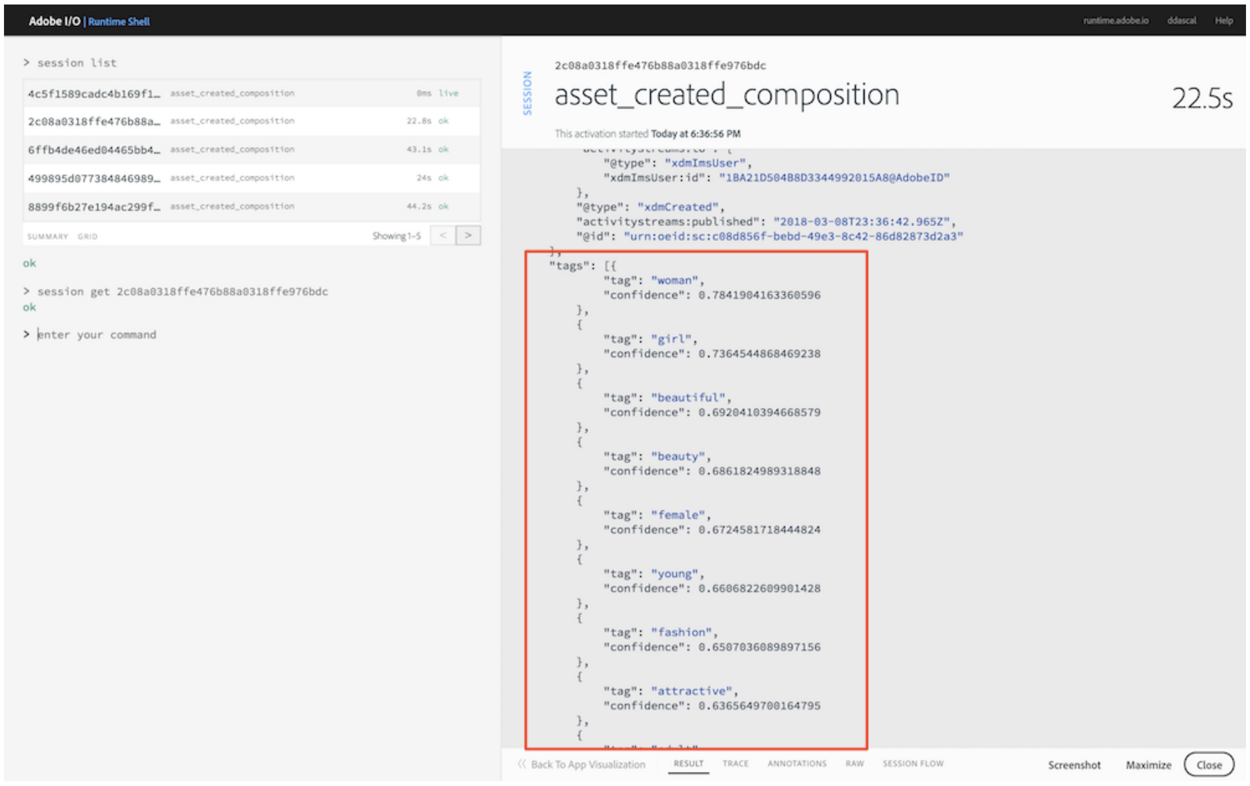
```
app update asset_created_composition ~/adobe-sensei-actions-lab/exercises/exercise-5/c
omposition.js
```

3. Now open the browser to your Creative Cloud and upload an image to trigger an `asset_created` event.

4. Switch back to the **Adobe I/O Runtime Shell** and type:

```
session list
```

5. Locate the most recent `asset_created_composition` running and click on the session id to view the result. If you scroll down in the results JSON you should see the tags that were generated like shown in the session below:



```
> session list
4c5f1589cadc4b169f1…  asset_created_composition   0ms   live
2c08a0318ffe476b88a…  asset_created_composition   22.8s ok
6ffb4de46ed04465bb4…  asset_created_composition   43.1s ok
499895d077384846989…  asset_created_composition   24s   ok
8899f6b27e194ac299f…  asset_created_composition   44.2s ok
ok

> session get 2c08a0318ffe476b88a0318ffe976bdc
ok

> enter your command
```

2c08a0318ffe476b88a0318ffe976bdc

asset_created_composition 22.5s

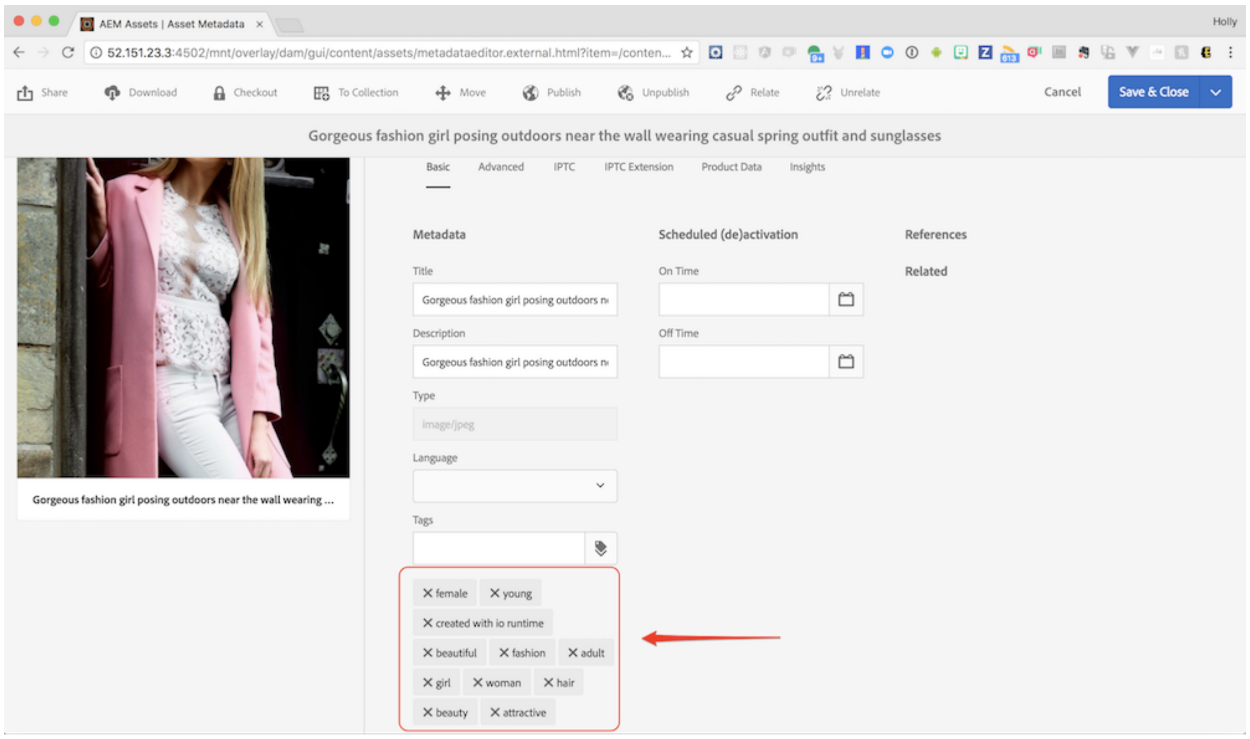
This activation started **Today at 6:36:56 PM**

```
            "@type": "xdmImsUser",
            "xdmImsUser:id": "1BA21D504B8D3344992015A8@AdobeID"
        },
        "@type": "xdmCreated",
        "activitystreams:published": "2018-03-08T23:36:42.965Z",
        "@id": "urn:oeid:sc:c08d856f-bebd-49e3-8c42-86d82873d2a3"
    },
    "tags": [{
            "tag": "woman",
            "confidence": 0.7841904163360596
        },
        {
            "tag": "girl",
            "confidence": 0.7364544868469238
        },
        {
            "tag": "beautiful",
            "confidence": 0.6920410394668579
        },
        {
            "tag": "beauty",
            "confidence": 0.6861824989318848
        },
        {
            "tag": "female",
            "confidence": 0.6724581718444824
        },
        {
            "tag": "young",
            "confidence": 0.6606822609901428
        },
        {
            "tag": "fashion",
            "confidence": 0.6507036089897156
        },
        {
            "tag": "attractive",
            "confidence": 0.6365649700164795
        },
        {
```

6. If the app ran successfully then you will see the asset copied into an Adobe Experience Manager instance within the same folder name as you defined in your Creative Cloud instance.

> ***NOTE:*** *Use the Adobe Experience Manager host and credentials provided to login and check Adobe Experience Manager for the file.*

Once you locate the file in Adobe Experience Manager, either hover over it and click the circled **i** button (or click into it and click on **Properties**) to view the details and check for the existence of tags. One of them should say `created with io runtime` as shown below:



Gorgeous fashion girl posing outdoors near the wall wearing casual spring outfit and sunglasses

Gorgeous fashion girl posing outdoors near the wall wearing ...

Metadata

Title: Gorgeous fashion girl posing outdoors n

Description: Gorgeous fashion girl posing outdoors n

Type: image/jpeg

Language:

Tags: female, young, created with io runtime, beautiful, fashion, adult, girl, woman, hair, beauty, attractive

Scheduled (de)activation

On Time:

Off Time:

References

Related