Lab Introduction
In this lab you will make Apcera REST API calls against your cluster using a web application called API Playground. You provide the server API endpoint, HTTP method type, and any parameters for the API method you want to call. The API response is displayed.

Lab Exercise

This lab comprises the following exercises:

1. Obtaining your API access token.
2. Making a basic API calls from the command-line using cURL
3. Stopping a job using the API from the command-line.

Lab Prerequisites
This lab assumes that you have set up your lab environment. If you have not done so, please follow the lab setup guide now.

Related Documentation
- **Apcera REST API Reference and API Models**
- **Apcera REST API Recipes**

Lab Instruction
For each exercise, complete the steps.

Exercise A: Obtain API token and make an API call from the command line
In this exercise you make a API call to /v1/jobs/ using cURL, a command-line utility for making HTTP requests. (See the lab setup guide for install instructions).

Recall that each request to a API method requires an HTTP “Authorization” header that contains your token There are several ways to do this, the specific approach you take depends on the application needs. For purposes of demonstration, you will simply copy the API token from the ~/.apc file that was obtained when you logged into the training cluster using the `apc login` command.

Task 1: Copy token from your ~/.apc file into environment variable
In this task, you’ll copy your access token from your local .apc file and create a new environment variable that contains the token value. You’ll also create another environment variable that holds the API Server’s endpoint (api.<cluster>.<domain>)

1. Open your **~/.apc** in a text editor.
2. Copy the entire value for your cluster’s token, including the "Bearer” string.

```
{
```

```
  "target": "http://yourcluster.example.com",
  "tokens": {
    "http://yourcluster.example.com": "Bearer eyJ0eXAiOi..."
  },
...
}
```

3. Create an environment variable named `APC_TOKEN` that contains the bearer token, being sure to put the variable's value in quotes:

   ```
   export APC_TOKEN="Bearer eyJ0eXAiOiIiLCJhbGciOiIi..."
   ```

4. Lastly, create an environment variable named `CLUSTER` that contains the API Server's endpoint. For example, if you are targeting the Kiso training cluster:

   ```
   export CLUSTER=api.kiso.io
   ```

Task 2: Make an API call

Next, you'll make a call to the `GET /v1/info` API using cURL.

1. From a terminal window, enter the the following command and press enter:

   ```
   curl -H "Authorization: $APC_TOKEN" -X GET $CLUSTER/v1/info
   ```

   You should get the following JSON-encoded response:

   ```
   {"name":"kiso","url":"http://api.kiso.io/","cluster_domain":"kiso.io"
   }
   ```

2. If you installed `jq` (as documented in the lab setup guide) you can pipe (`|`) the JSON response through `jq` to format the output for easier reading:

   ```
   curl -H "Authorization: $APC_TOKEN" -X GET $CLUSTER/v1/info | jq
   {
     "name": "kiso",
     "url": "http://api.kiso.io/",
     "cluster_domain": "kiso.io"
   }
   ```

Exercise B: Updating a job with the API

A common API task is updating a job object. The basic process to update a job involves two API calls: one to get a JSON description of the job you want to update (typically by calling `GET`

`/v1/jobs`). You then modify the job's JSON description as desired, and call `PUT /v1/jobs/{uuid}` passing the updated JSON description in the HTTP body, where `{uuid}` is the job's UUID.

In this exercise you'll create a new capsule using APC or the web console. You'll then use an API call to stop the capsule by setting its `state` property to `"stopped"`.

Task 1: Create the capsule using APC or web console

- Using APC, run the following command:

  ```
  apc capsule create testcapsule --image linux
  ```

- Or using the web console, select **New > Capsule** from the left navigation and select an available Ubuntu distribution. Give the capsule a name and click **Submit** to create the capsule.

Note the job's fully-qualified name, or FQN (`/sandbox/user::testcapsule`, for example).

Task 2: Get capsule job's JSON description

To get the capsule's JSON description you'll call GET /v1/jobs, passing the capsule's FQN as query parameter and save the response to **capsule.json** in the current working directory:

```
curl -H "Authorization: $APC_TOKEN" -X GET -G $CLUSTER/v1/jobs \
--data-urlencode "fqn=job::/sandbox/username::testcapsule" > capsule.json
```

Task 3: Update the JSON description

Next you'll update the capsule's JSON description in the capsule.json file.

1. Open **capsule.json** in a text editor.
2. Locate the `"state"` field and change its value to `"stopped"`. Also note the value of the job's `uuid` field as you'll need it in a following step:|

```
[
  {
    "uuid": "6bc6d841-0e8d-44fc-9fe7-a3640e3f74a4",
    ...
    "state": "stopped",
    ...
  }
]
```

3. Remove the square brackets (`[]`) from the top and bottom of the document and save the changes to the file. (The reason is that `GET /v1/jobs/` returns an array of job objects, rather than a single object).

```
{
  "uuid": "6bc6d841-0e8d-44fc-9fe7-a3640e3f74a4",
  ...
}
```

Task 4: Call API to update the job description

The last step to update the capsule is to call `PUT /v1/jobs/{uuid}`, where `{uuid}` is the capsule's UUID obtained from the previous call to `GET /v1/jobs`.

1. Run the following cURL command to update the job description on the cluster with one in capsule.json. Replace **`{UUID}`** with the value of the `uuid` field in your capsule.json file:

   ```
   curl -H "Authorization: $APC_TOKEN" -X PUT \
   $CLUSTER/v1/jobs/{UUID} \
   --data "@capsule.json"
   ```

To verify if the capsule has stopped you can do one of the following:

- From the command-line run:

  ```
  apc capsule show testcapsule
  ```

  The value of the **State** field in the output should be **stopped.**

- Using the web console, check the capsule's state.