Processing Large Amounts of Data in Salesforce with Bulk API 2.0

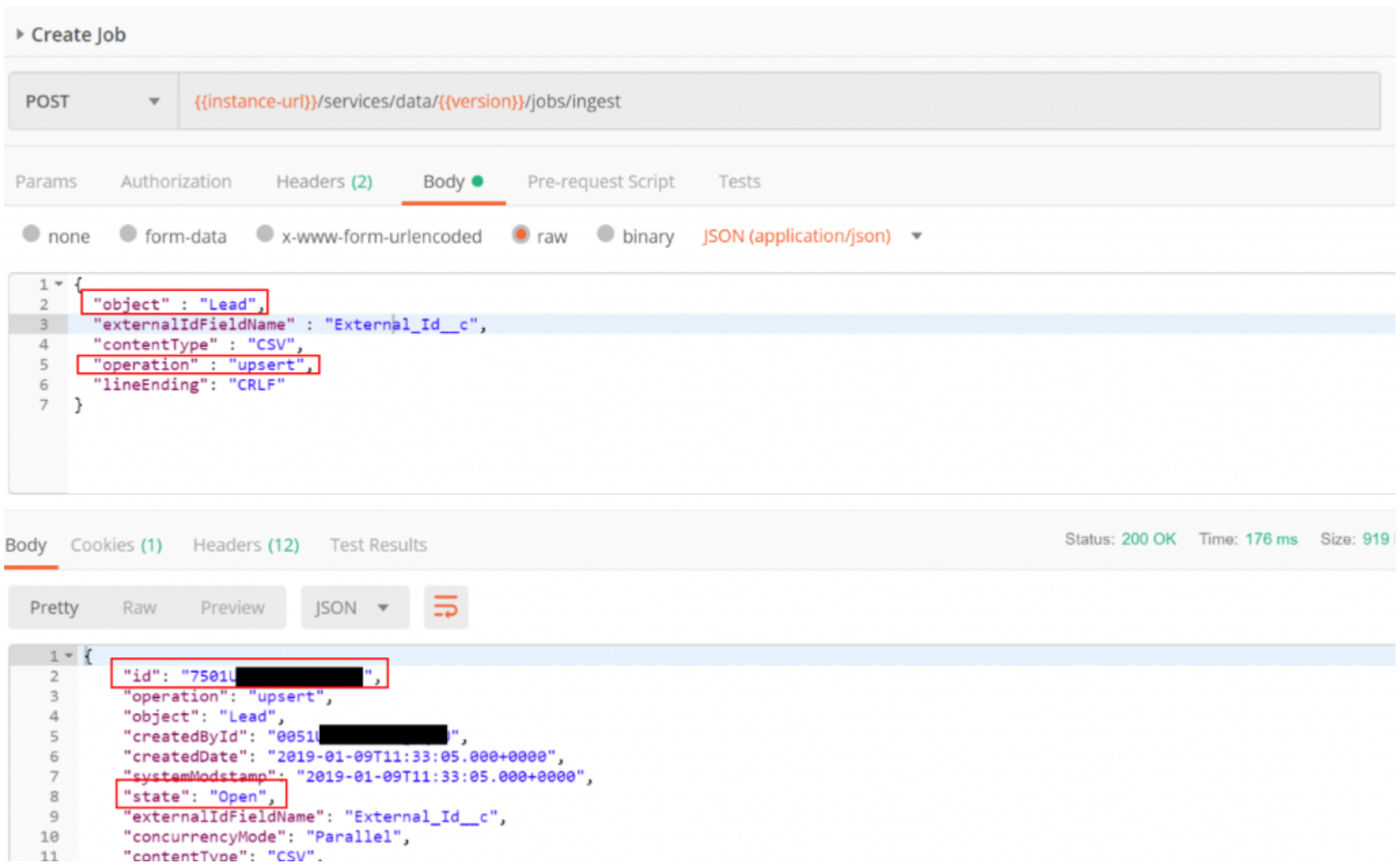
Running into limits with your Salesforce REST API usage, or simply proactively looking to process a large amount of data in Salesforce? Due to the limits in place on how many calls you can make to the REST API on a per 24-hour period that may limit how much data you can process an alternative approach is needed. Thankfully, with the Bulk API, you can process up to 100 million records per 24-hour period, which should (hopefully) be more than enough for you.

If you need help authenticating with the Salesforce API and getting up and running, or using Postman to manage requests and collections, I've written about those both previously. Also, if you would like a way to generate lots of sample data to test with, I've written about the Faker JS library which could help with that too. (Note: Faker is available on other platforms also: .NET, Ruby, Python, and Java).

As a final note, you can download the Postman collection I've been using for my Salesforce series on Github if you want to follow along that way.

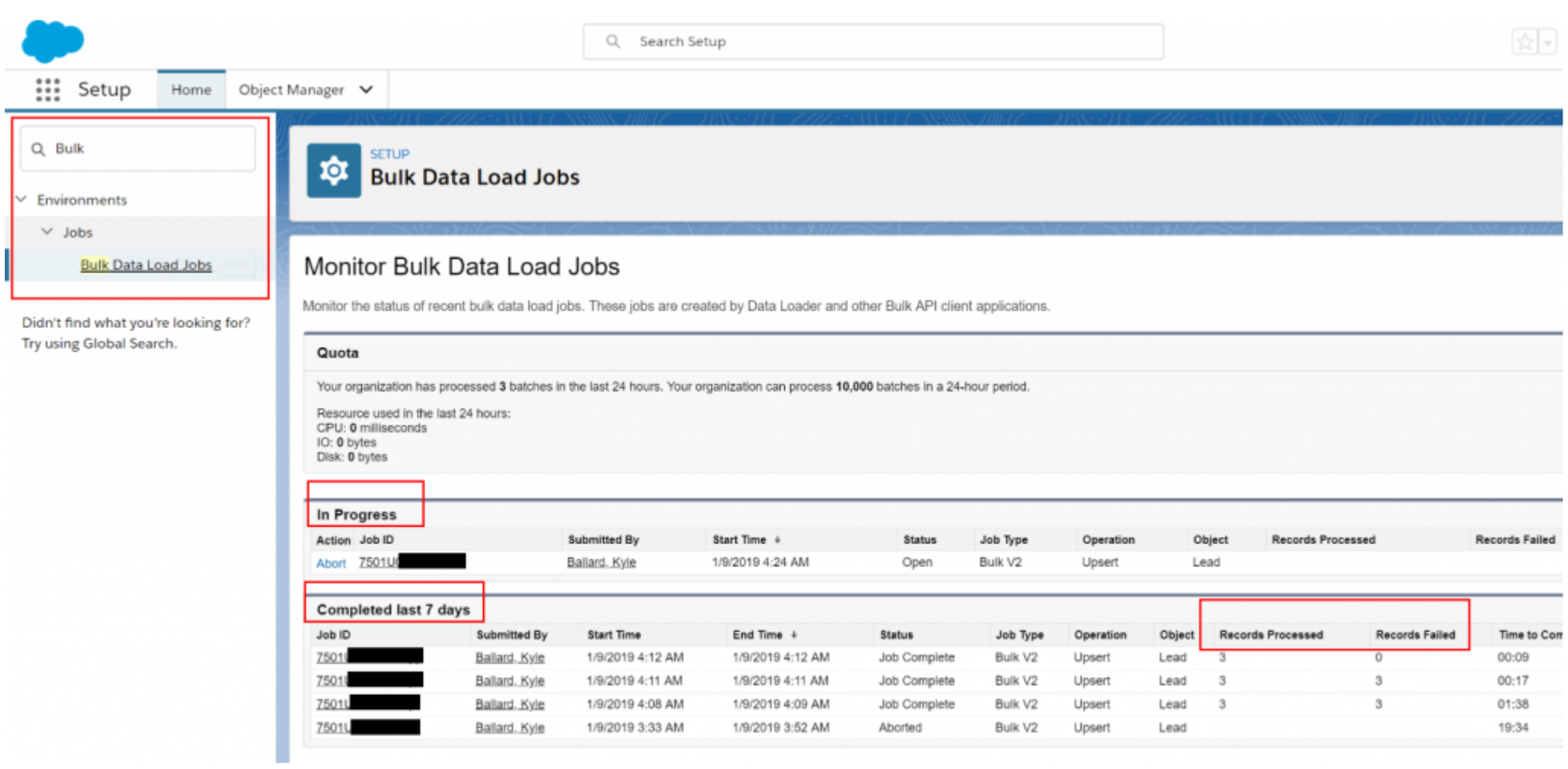
Create the Job

The first step in creating a Bulk API request is to create a job. As part of the job request, you specify the object type, operation (insert, delete, upsert), and a few additional fields. In this example, we're using the upsert operation paired with the External_Id__c field. If you don't have have this external id field on your Lead object, you can see my REST API post where I talk about working with external keys which is very important when connecting external business systems. More information about the request body is available here.



Once the response is returned to us, we can see we're given a job id, as well as some additional information about the job, including the job's state, which is currently set to Open. Open state means the job is ready to begin accepting data. We also have a Postman "Test" attached to this request, which will save the *{{job-id}}* parameter which is returned and can then be used in future requests.

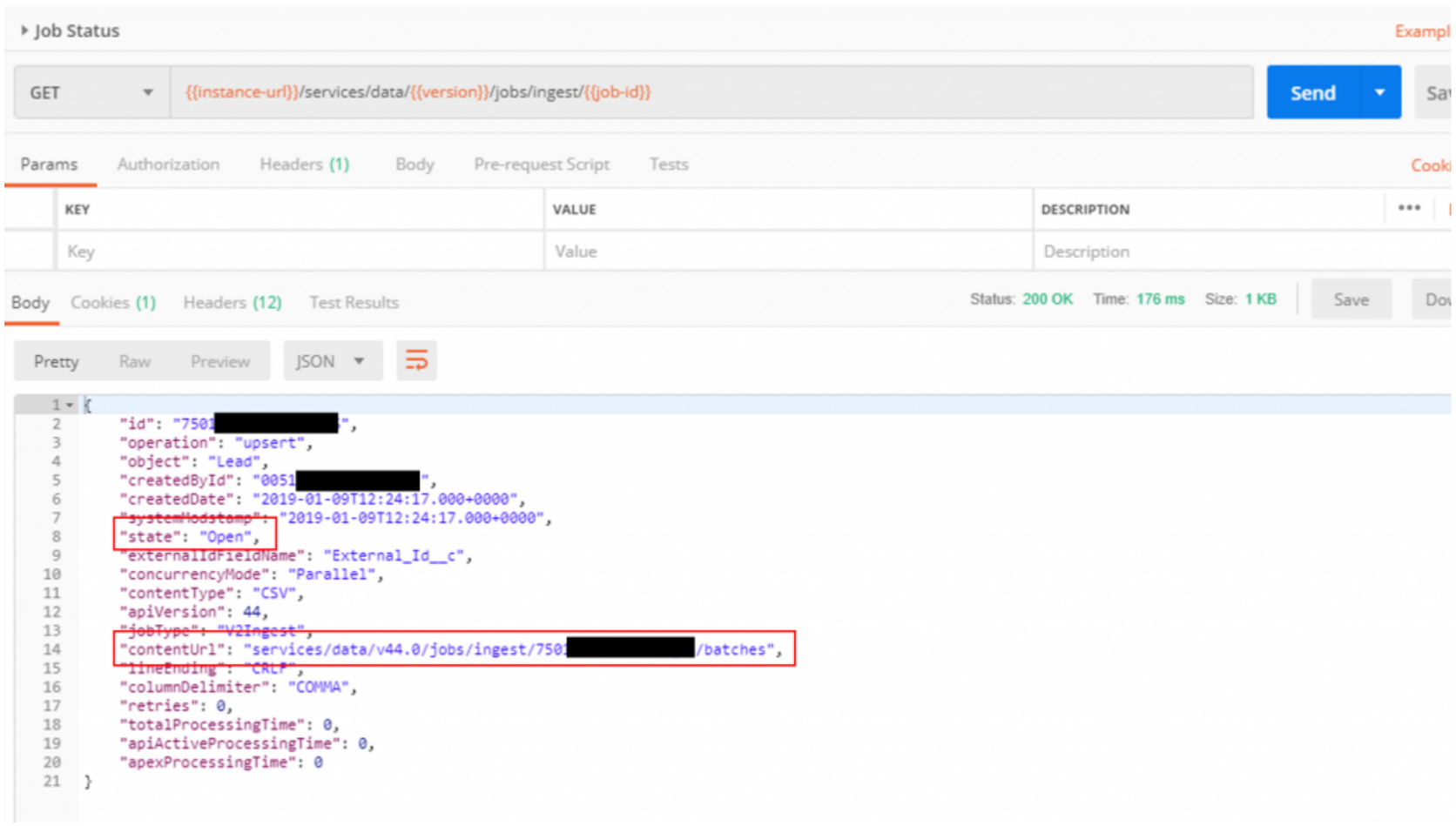
As a bonus, you can also view your bulk data jobs in the Salesforce web administration panel. Just head to Setup > Environments > Jobs > Bulk Data Load Jobs' and you can see your currently open jobs as well as any jobs you completed in the last 7 days.



Checking the Job Status

A common request when working with bulk jobs, is to check the status of the job. We'll do this now, and see the job is still in the 'Open' state, ready to accept data, but in the future when we use this, we may see our job in any of the following states: Open, UploadComplete, InProgress, JobCompleted, Aborted, or Failed.

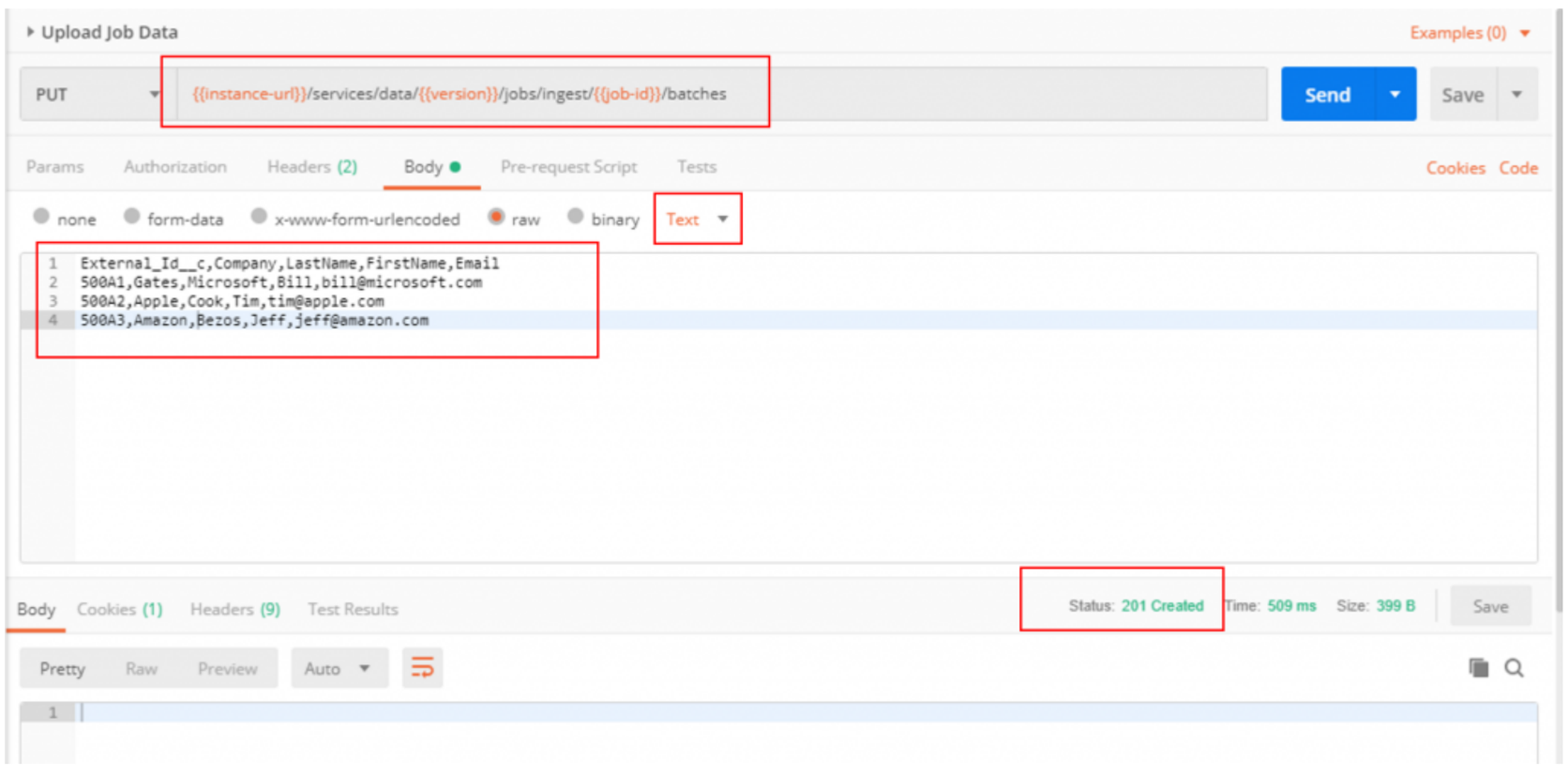
We're also given the contentUrl parameter here, which is where we want to post our actual CSV data.



Uploading the Data

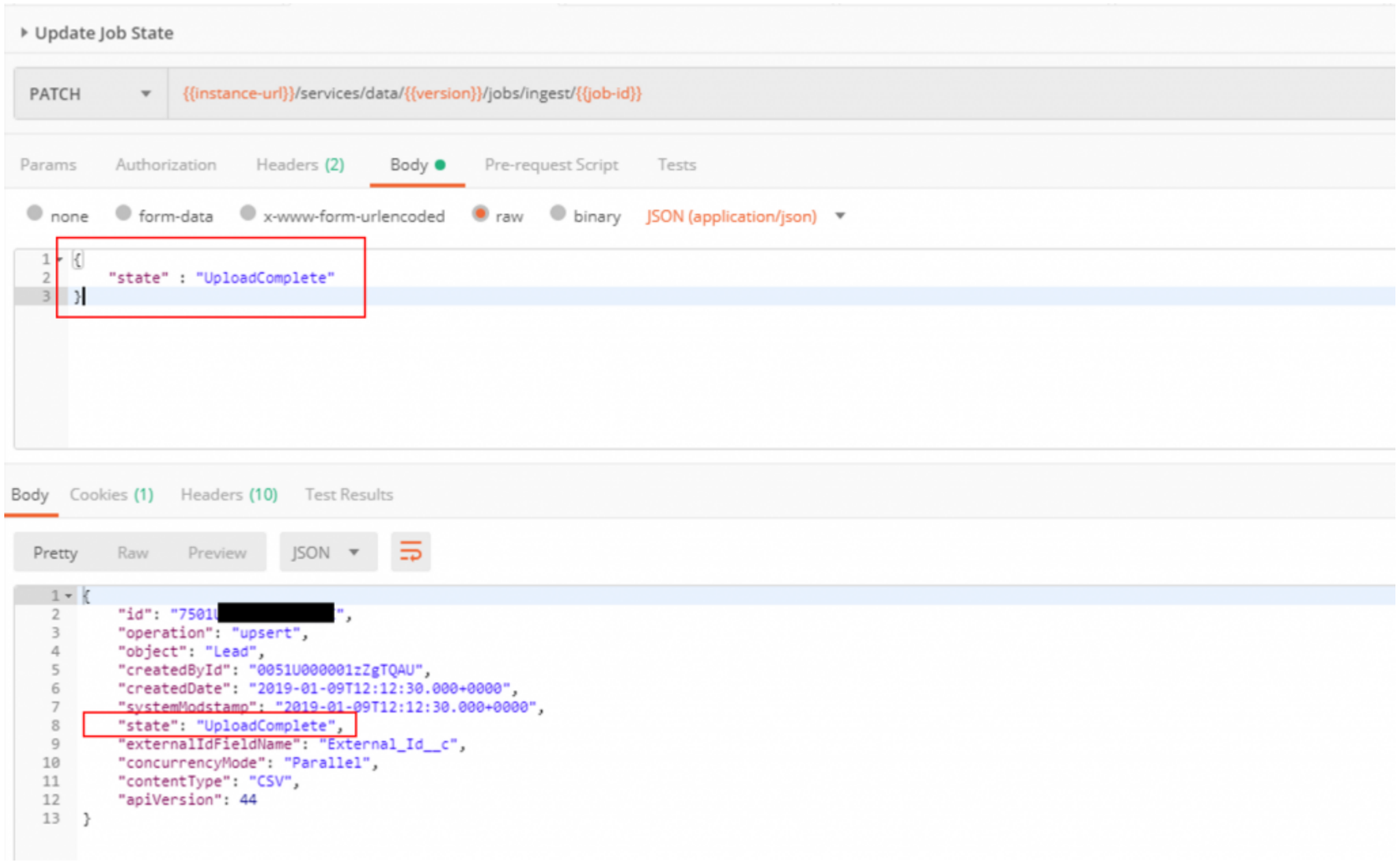
Once we are armed with an 'Open' job, ready to accept data, we're ready to start sending data to the job. We do this by sending a PUT request to the *contentUrl* from our job. In the screenshot below you can see we've set our Content-Type to text/csv, and are sending the CSV data in the request body. Here you can match fields for your job's object type (Lead in this case). If you were sending something like Contact records, you could also specify a field for Account.External_Id__c (assuming you had external id setup on Account) to properly link the contact records to the corresponding Account object.

There are also limits to be aware of when sending your CSV data to the job. The current limit is 150 megabytes of base64 encoded content. If your content is not already base64 encoded, consider limiting your size to 100 megabytes since Salesforce converts your request to base64 upon receipt and this can result in up to a 50% increase in size.



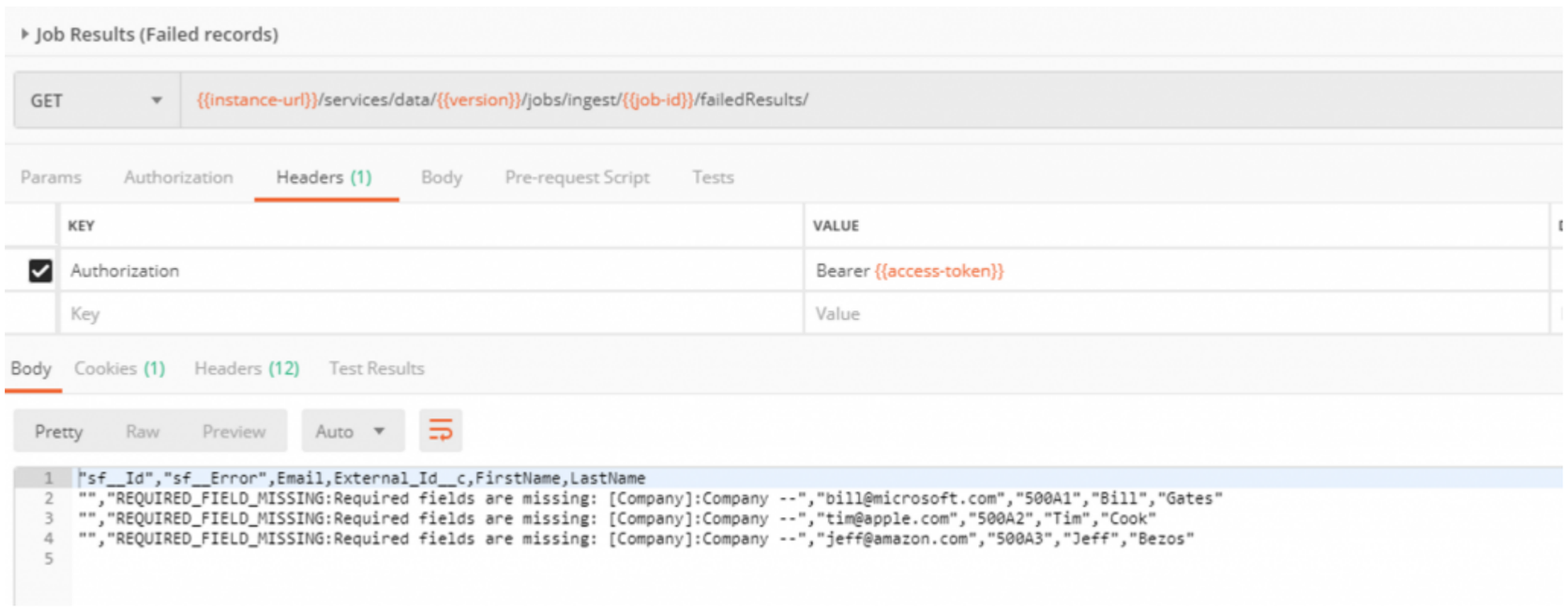
Marking the Job Complete

We briefly mentioned job state earlier when talking about job status. Once you have finished sending the batches to your job, you will need to mark your job completed by setting the status to *UploadComplete*. If you for some reason didn't want to finish processing the job and wanted to discard it, you could also set the job state here to *Aborted*. Once the job is marked as *UploadComplete*, Salesforce will shortly thereafter mark the status as *InProgress*, and shortly after that, as *JobCompleted* or *Failed*.



Obtaining the Job Results

Salesforce makes two endpoints available to obtain the results of your bulk job's successful and failed records. Once your job's status is JobComplete, you can make a request to */jobs/ingest/{{job-id}}/failedResults* to obtain the records that failed to process, including an example of why those records were unsuccessful. You'll always get back a sf__Id and sf__Error in your response so you can handle these in your application appropriately. Similarly, you can also send a request to */jobs/ingest/{{job-id}}/successfulResults* to obtain the records which were successfully processed. Inside the success records, you'll also receive a sf__Id and also a sf__Created property which indicates if the record was created (or modified).



Summary

In this post we discussed the Salesforce recommended way to work with larger datasets. The Salesforce REST API is great for handling transactional records, or even working with up to 25 records at a time with the composite and batch REST endpoints, but for larger recordsets, up to 100mb in size, the preferred way is using the Bulk API.