Lab Guide

ServiceNow custom REST APIs:

Build Custom Services the right way with Scripted REST APIs

This

Page

Intentionally

Left

Blank

Lab Goal

Lab Setup

Before we get started building custom services with Scripted REST APIs, we need to get our lab instance setup. In this lab you will be modifying an existing scoped application. Start out by importing the Polls Application from Source Control. Follow the directions below to fork this application to your GitHub account and begin working.

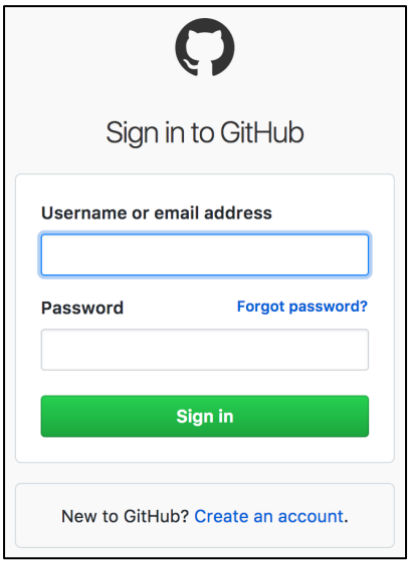
Prerequisites

In order to complete this lab, you must:

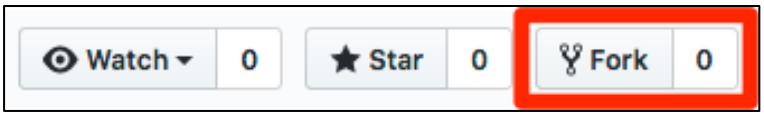
- Create a GitHub account, if you do not already have one.
- Install Postman from https://getpostman.com if you do not already have it.

Fork the Lab GitHub Repository

1. Log in to your GitHub account at https://github.com/login.



2. Navigate to: https://github.com/balazsburgermeister/ScriptedRESTAPI
3. Click **Fork.**

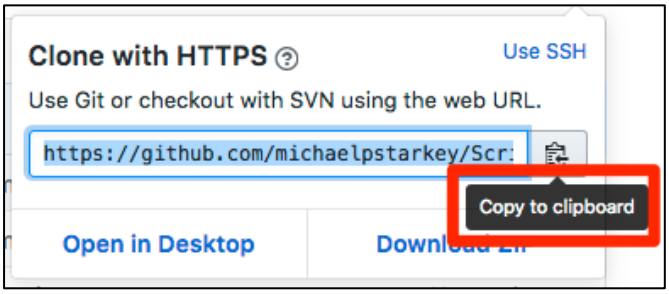


4. Note in the upper left that the repository has been copied to your account. You now have a copy of the lab material for reference after the conference!

5. Locate and click on the **Clone or download** button and then click the clipboard to the right.

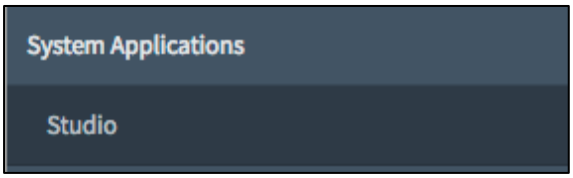
   This action copies the URL in the clipboard.

   IMPORTANT: Be sure to copy the **HTTPS** repo URL in GitHub.



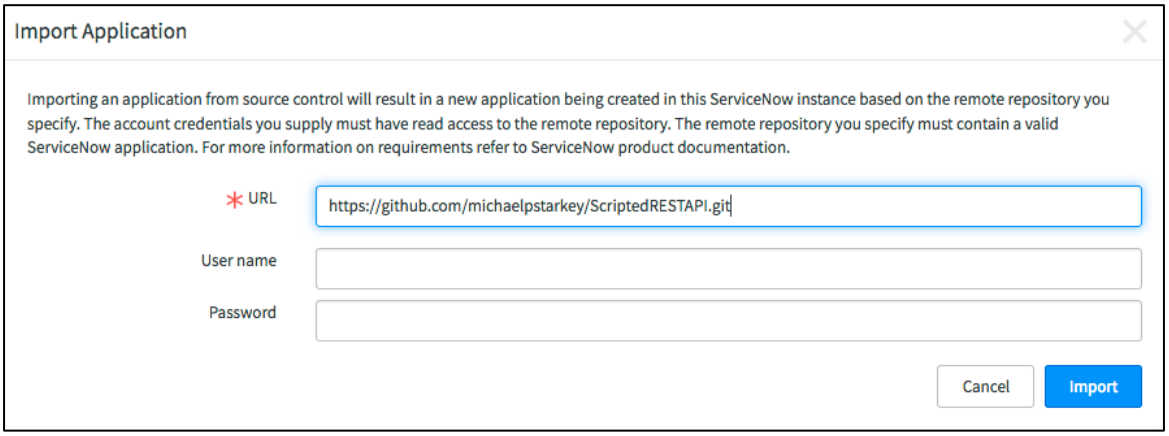
Import the Polls Application from Source Control

6. Log in to your instance with the credentials provided on the cover sheet of this document.
7. Navigate to **System Applications > Studio**.



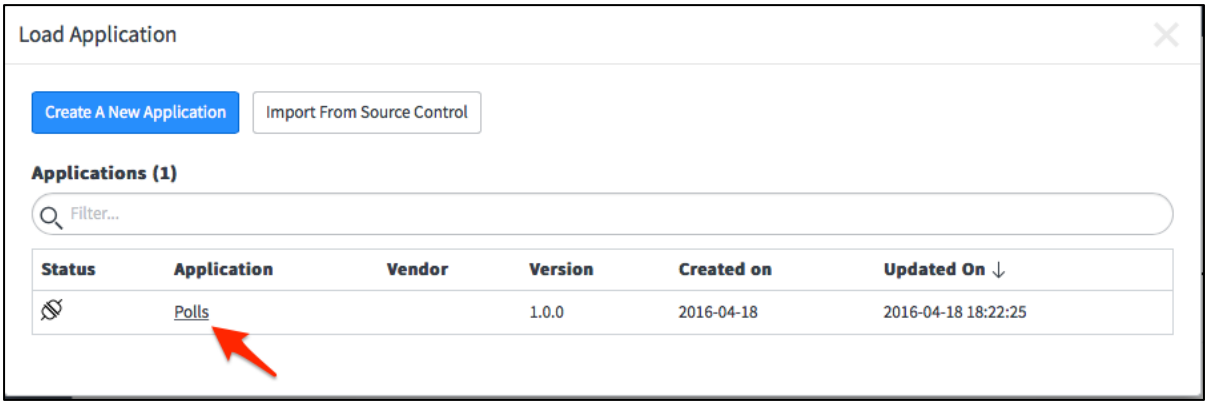
8. Click **Import From Source Control.**

9. In the Import Application window, paste the URL copied in step **5** and provide your GitHub credentials. Click **Import**.



10. When the import completes, click **Select Application**.

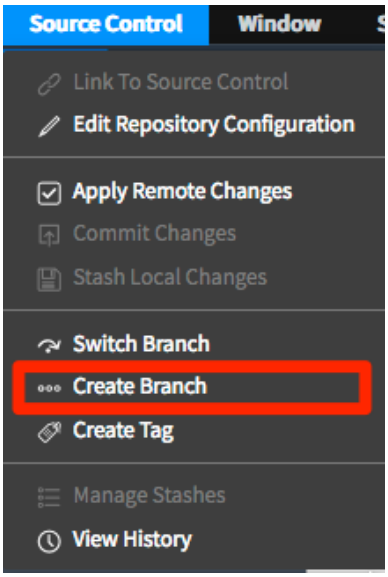
11. Click on the **Polls** application you just imported.



You've now successfully imported your forked version of the application for use in this workshop.

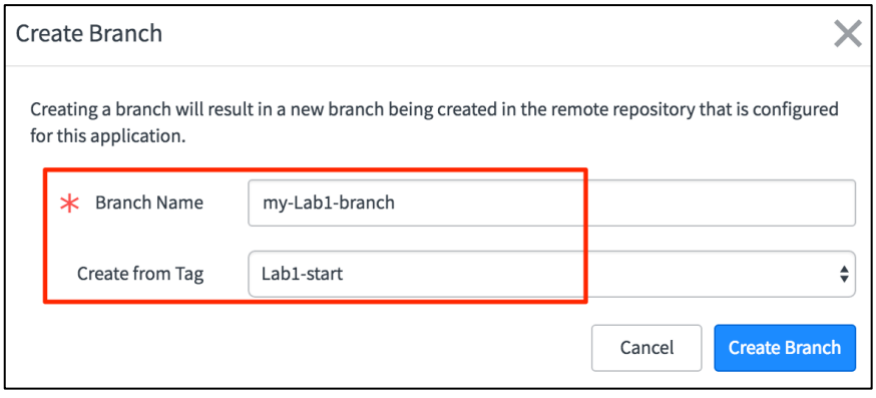
Get ready for Lab 1 – Create a new branch from Lab1-start tag in Studio

12. In **Studio**, navigate to **Source Control > Create Branch**.



13. In the pop-up window, enter a branch name, then select **Lab1-start** from the **Create from Tag menu**, and click **Create Branch**.

    Branch: **my-Lab1-branch**
    Create from Tag: **Lab1-start**



14. When the create is complete, click **Close Dialog** in the Create Branch pop-up.

15. Verify Studio is on branch **my-Lab1-branch** from the bottom right corner of the screen.

Lab setup is complete. You are now ready to start Lab 1.

Lab Goal

Lab 1
Build "Hello, world!"

The purpose of this lab is to familiarize yourself with ServiceNow Scripted REST APIs. In this first lab you'll build a Scripted REST API that returns "Hello, world!" in response to a GET request. After building the API you'll use the ServiceNow REST API Explorer and API testing tool Postman to make requests to the REST API.

Prerequisite

- Knowledge of REST APIs
- Knowledge of HTTP clients
- Postman API testing tool. To get Postman go to: https://www.getpostman.com/

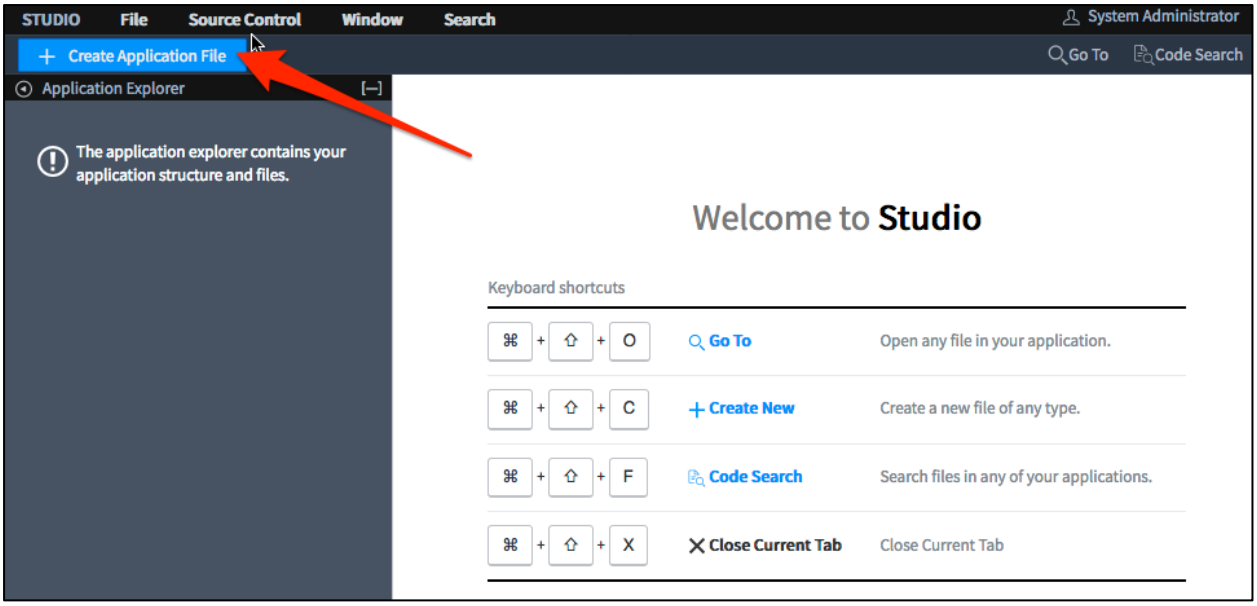
Create Lab 1 starting branch

1. If you completed the lab setup, proceed to the next step.

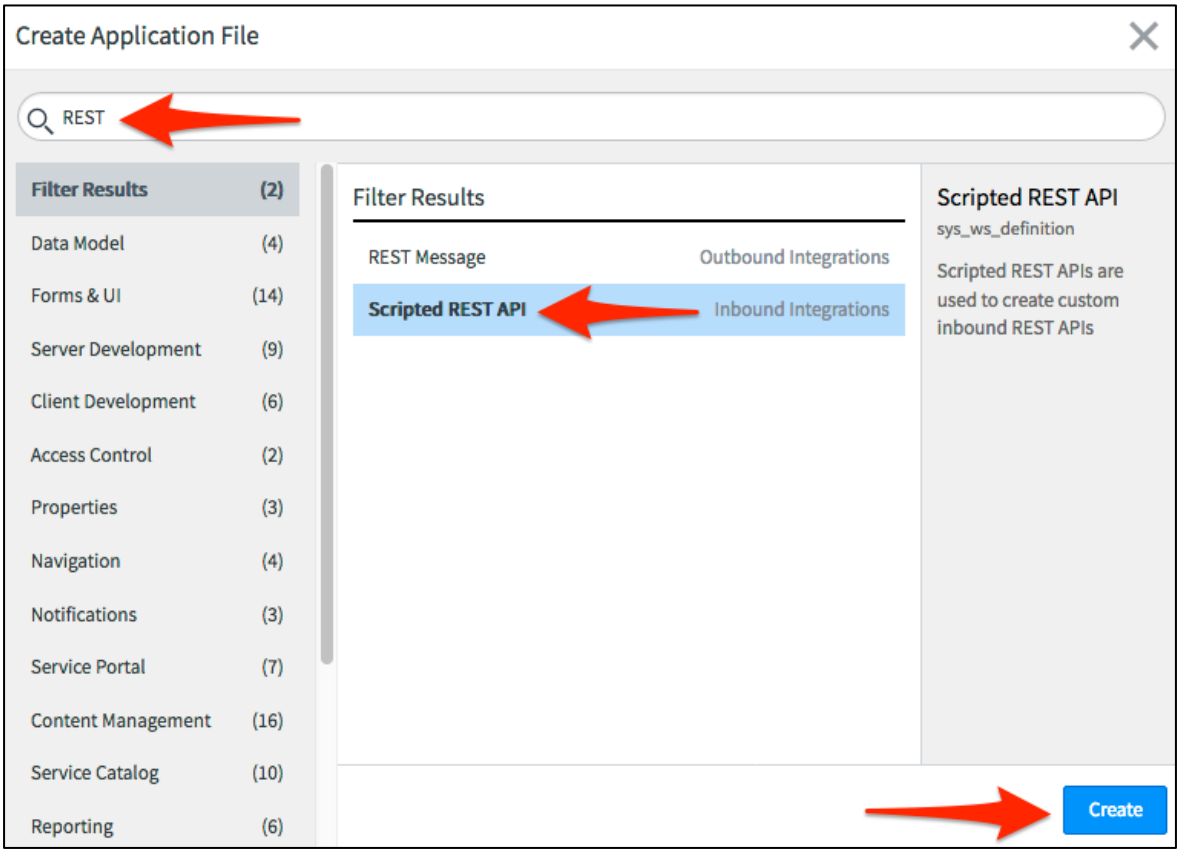
   If you haven't yet completed lab setup, follow the steps in lab setup to create the **my-Lab1-branch** from the **Lab1-start** git tag.

Create the Hello World Scripted REST API

2. In Studio, click **Create Application File**.

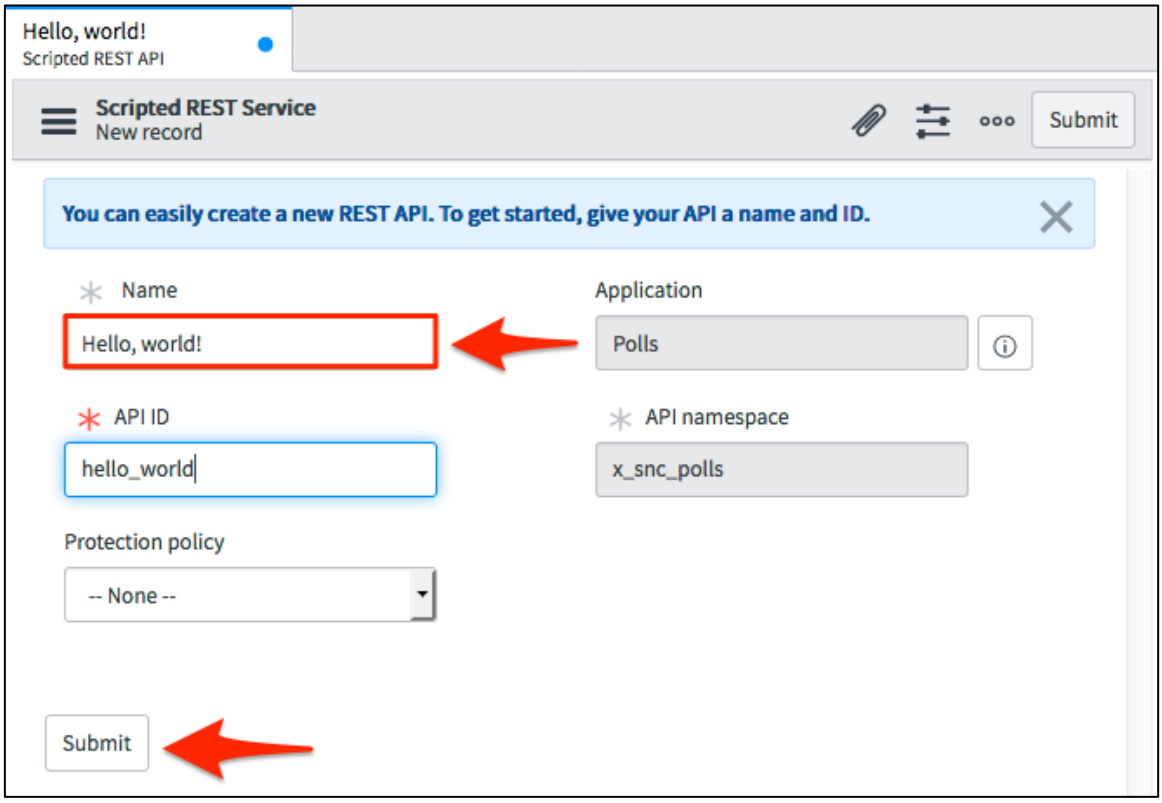


3. In the **Create Application File** window, type **REST** in the filter then select **Scripted REST API** and click **Create**.



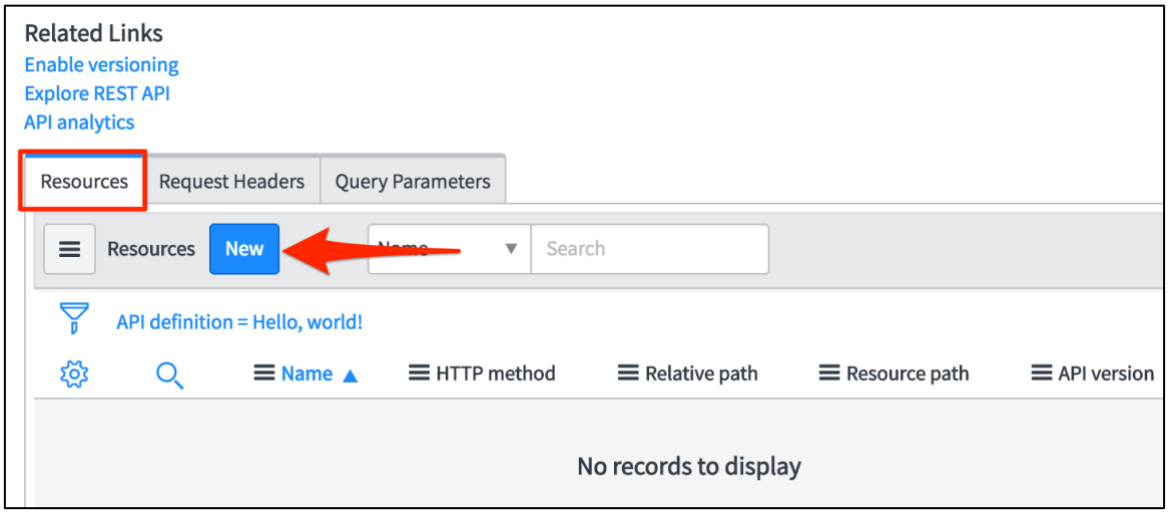
4. Give the **API** a name. Note the **API ID** populates automatically from the API Name, but can be changed.

   Name: **Hello, world!**



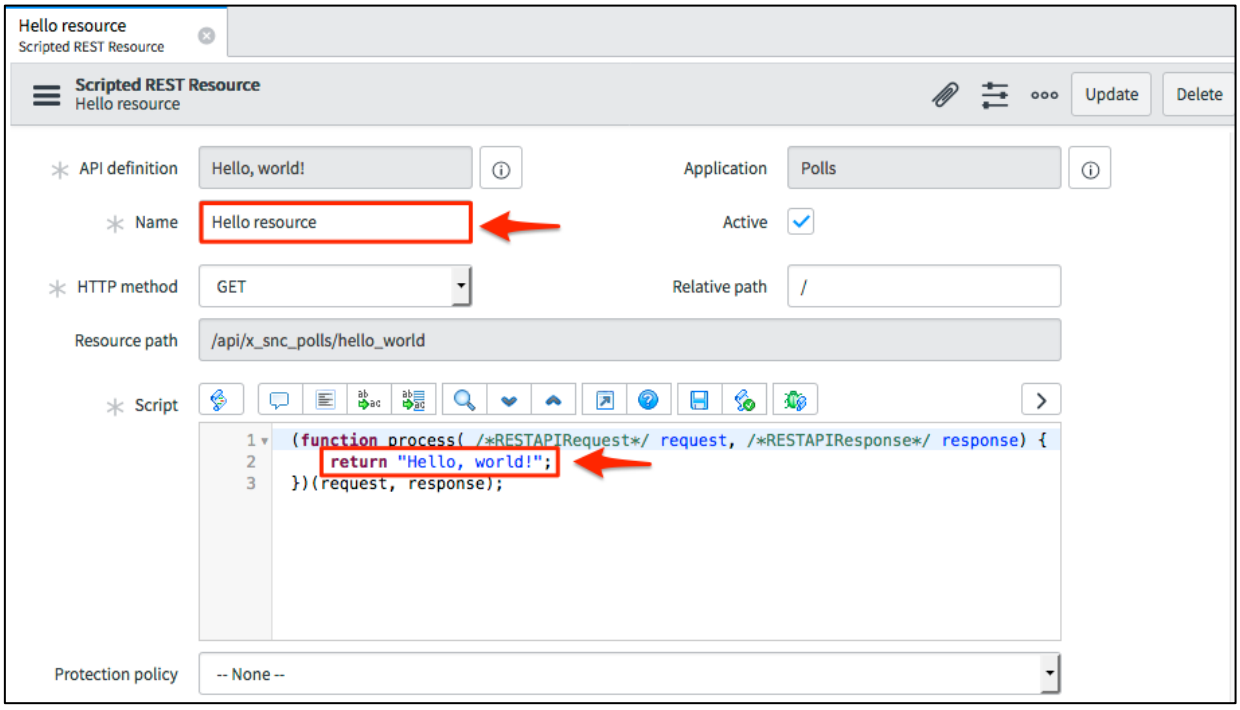
   Click **Submit.**

5. Add a **Resource** to the API by finding the **Resources** related list. Click **New**.



6. Specify the following properties for the new resource and complete the script.

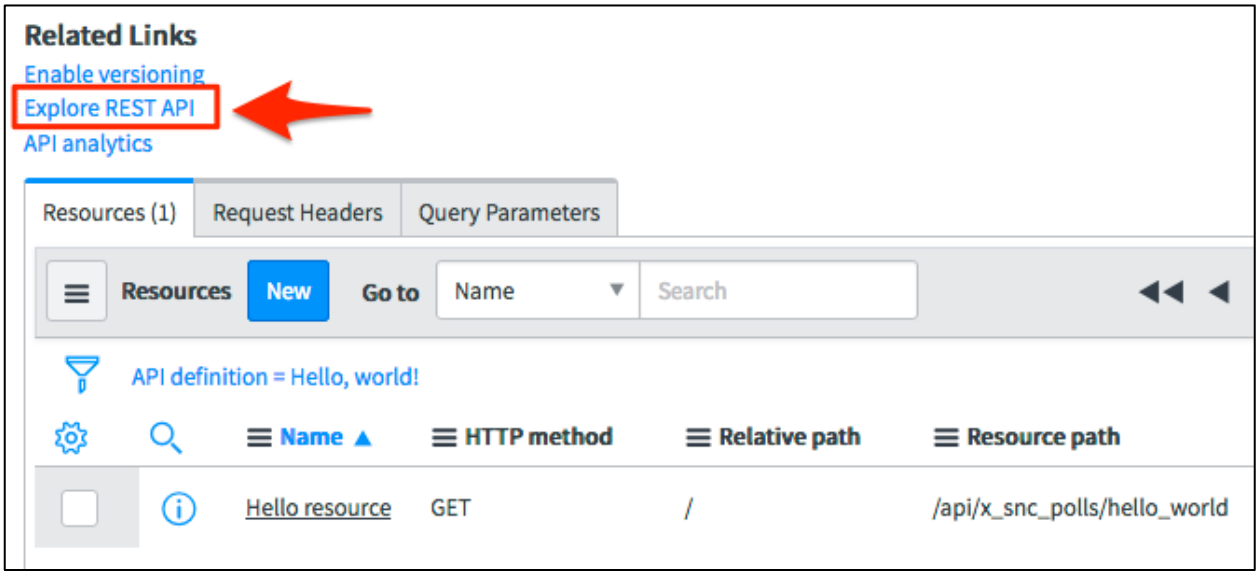
   Name: **Hello resource**
   Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab1



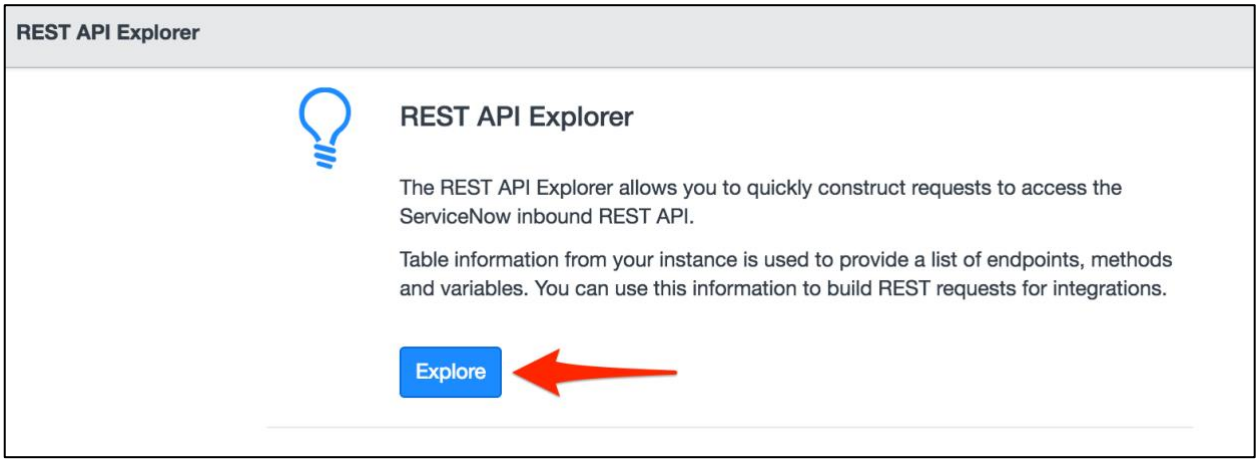
   Click **Submit**.

Test with REST API Explorer

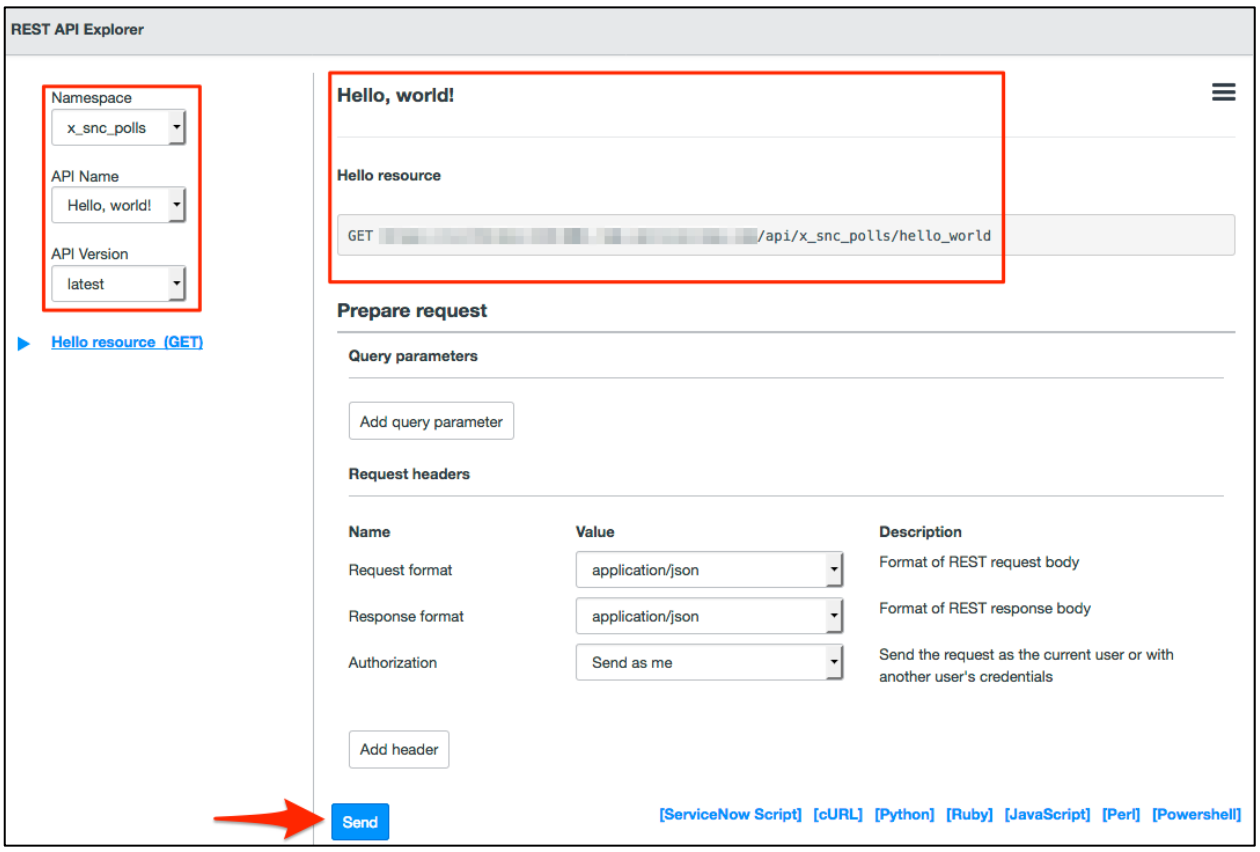
7. Click **Explore REST API**.



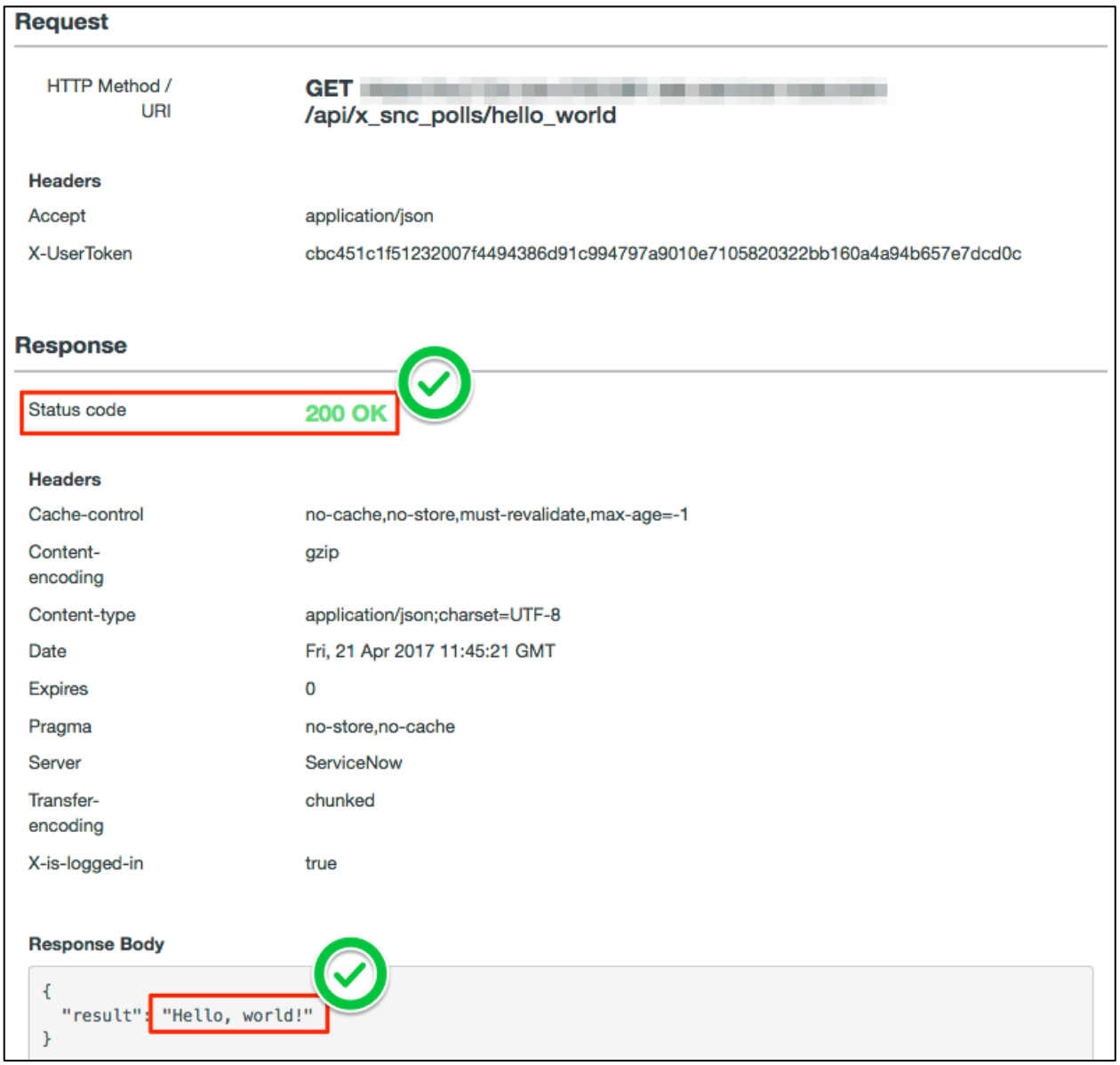
8. The **REST API Explorer** opens in a new browser window. Click **Explore**.



9. The “**Hello, world!”** Scripted REST API is pre-selected in the Explorer menus. Click **Send**.



10. Verify response status code is **200 OK** and response body is **“Hello, world!”**.



Get Caught Up

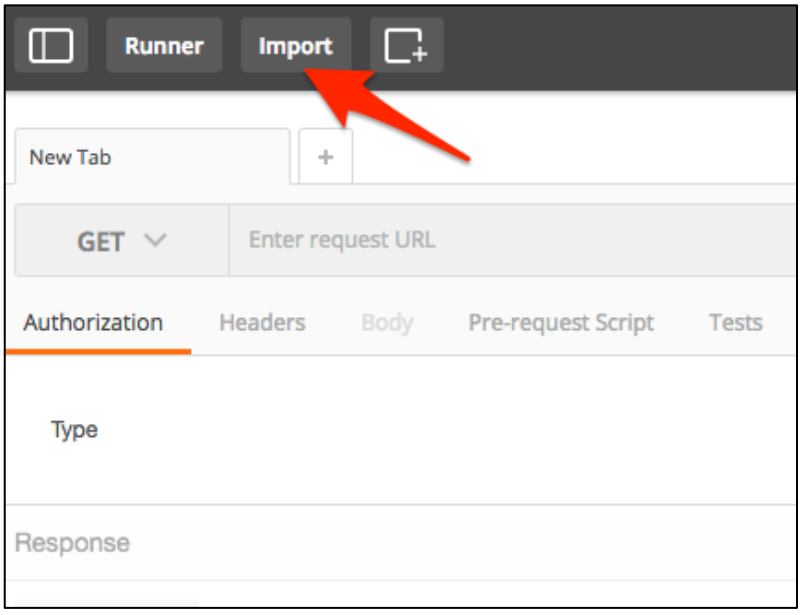
If you were unable to successfully complete the lab this far, you can “fast forward” using the following steps. Otherwise proceed to the next section **Test with Postman**.

11. Similar to creating the Lab1 starting branch, the completed lab can also be checked out from a tag (**Lab1-complete**) in Source control.
12. In **Studio**, navigate to **Source Control > Create Branch**.
13. In the pop-up window, enter a branch name, then select **Lab1-complete** from the **Create from Tag menu**, and click **Create Branch**.

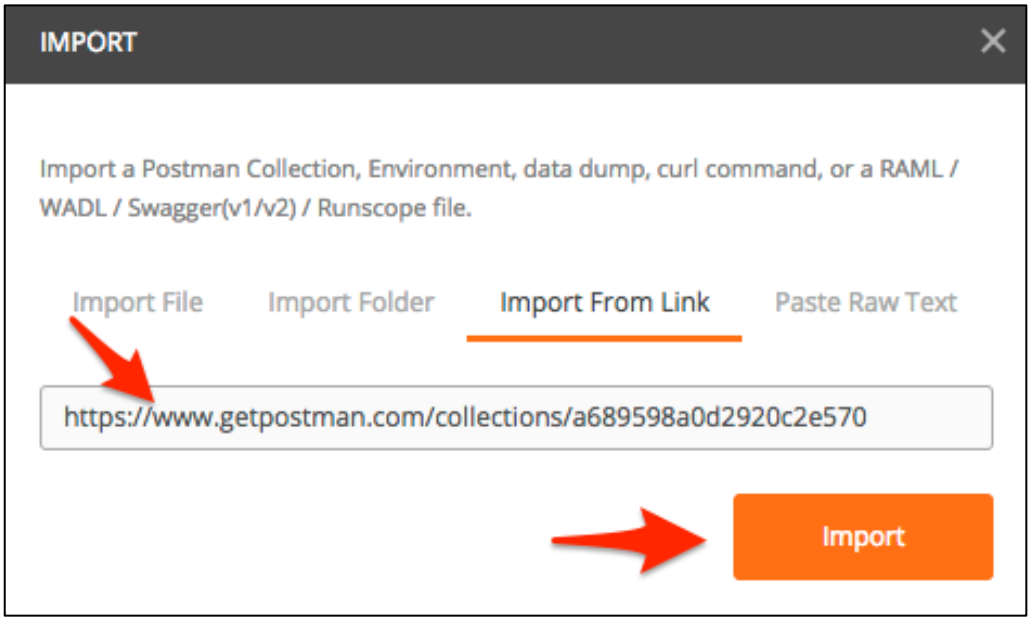
    Branch: **my-Lab1-branch-complete**
    Create from Tag: **Lab1-complete**
14. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
15. Verify Studio is on branch **my-Lab1-branch-complete**.
16. You are now ready to continue with the next section of **Lab 1**.

Test with Postman

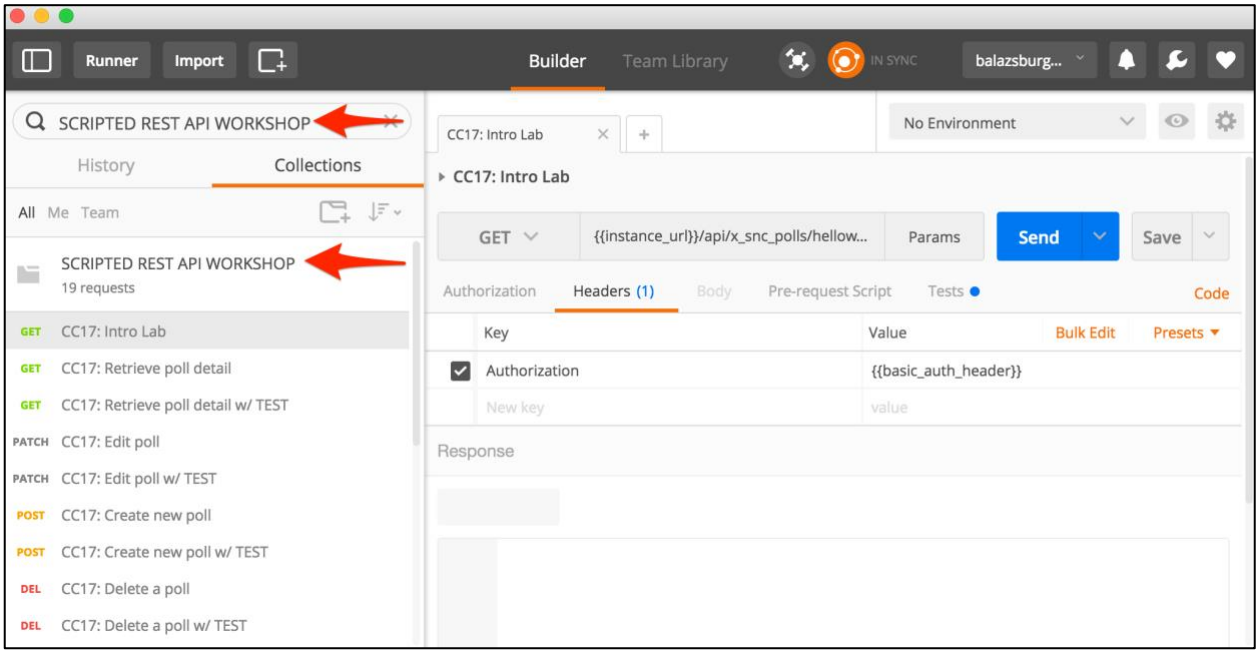
17. Open the **Postman** application on your laptop.
18. Import the Postman collection we will be using for this workshop from:
    Postman Collection Link: https://www.getpostman.com/collections/a689598a0d2920c2e570
19. In Postman, click **Import**.



20. Paste the link to our Postman collection in the **Import from Link** and click **Import.**

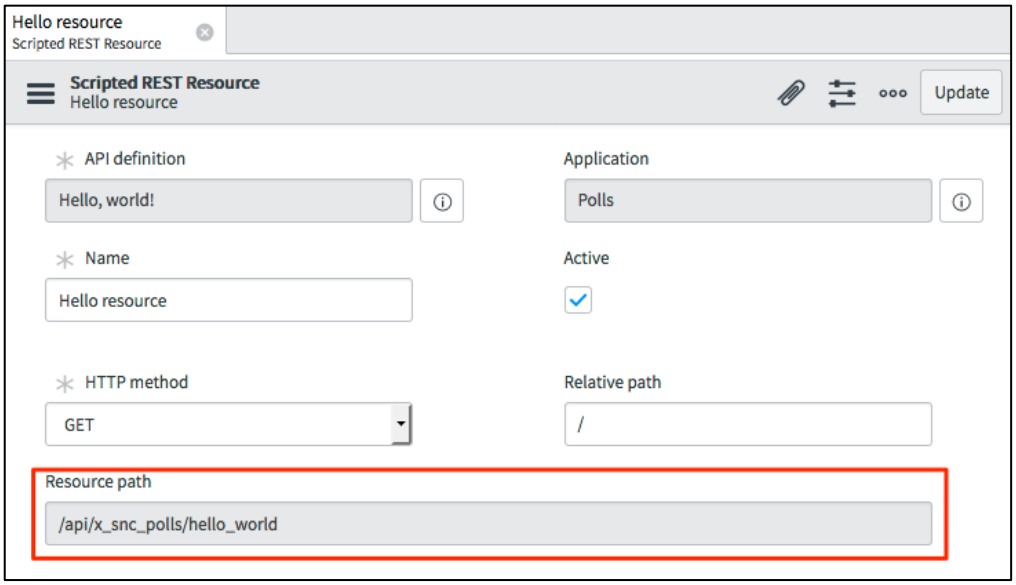


21. Verify you have the "Scripted REST API Workshop" collection loaded by searching for it in the navigator on the left hand side.



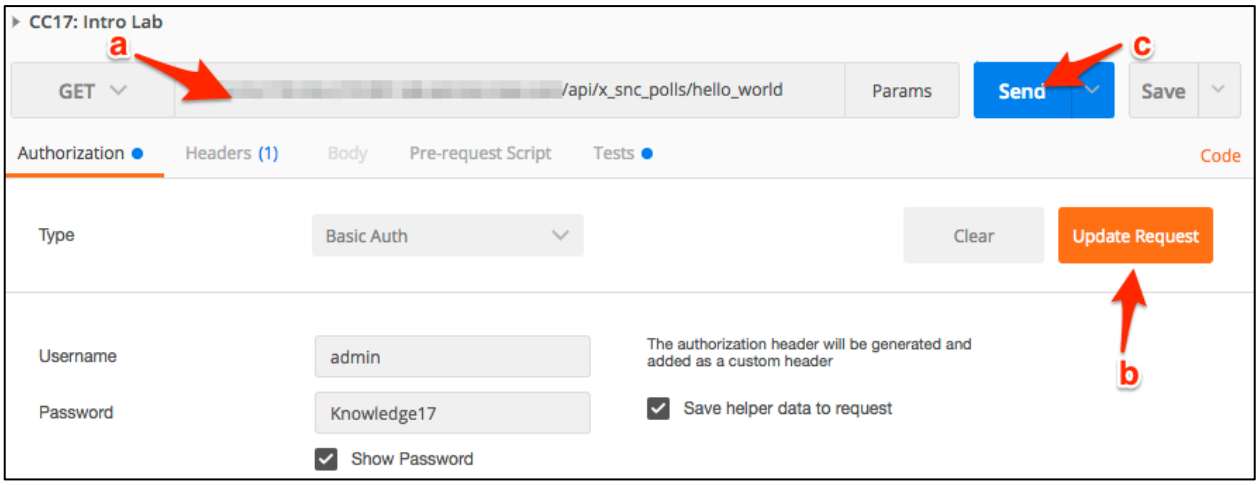
22. In the **Scripted REST API Workshop** select the **CC17: Intro Lab**.

    a. Replace {{instance_url}} with your lab instance URL (for example, https://my_instance.lab.service-now.com), and replace the resource URI with the resource from your Hello World Scripted REST API. Copy/paste the resource path from the **Resource path** field.

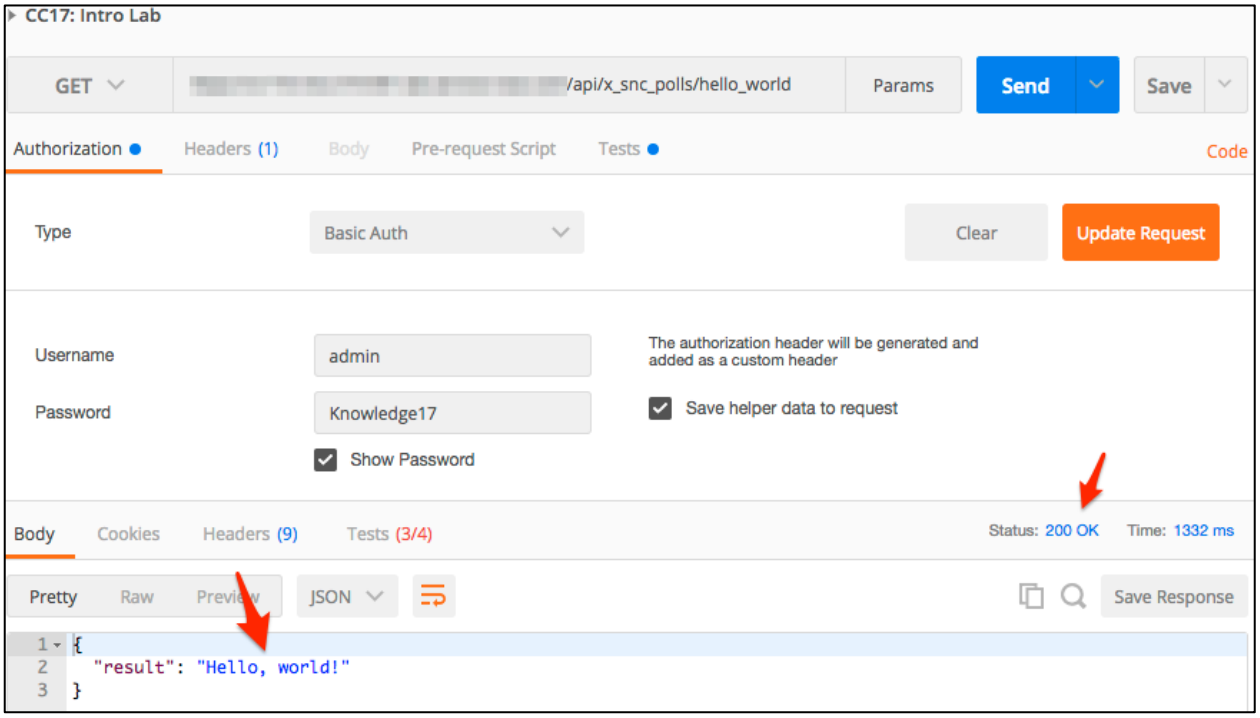


    b. Click **Update Request**.

    c. Click **Send** to send the HTTP request.



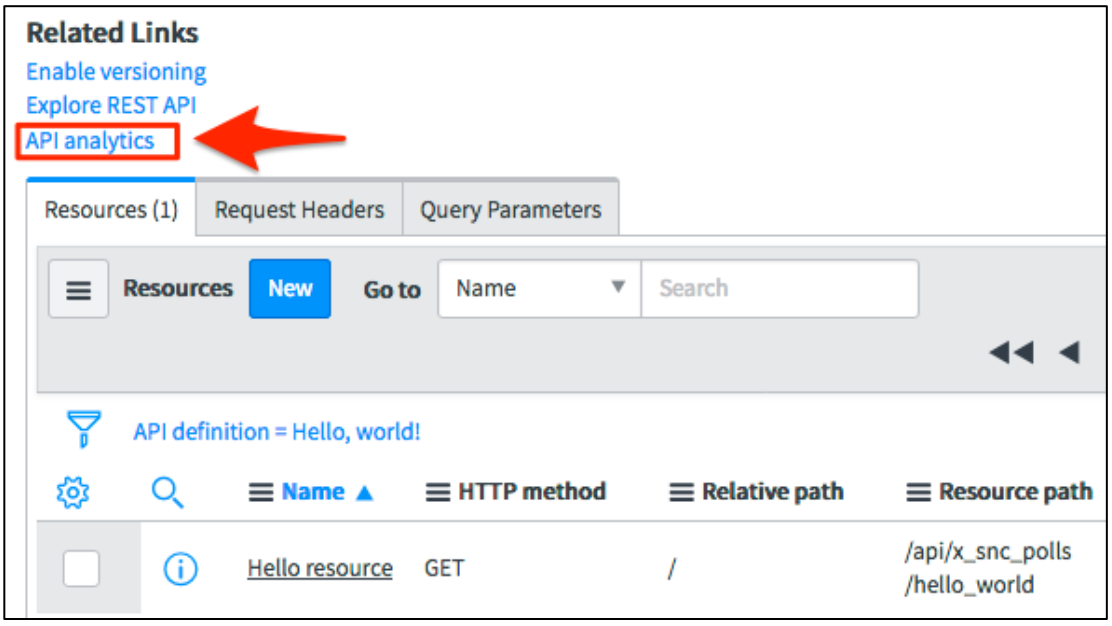
23. Validate response is successful by looking for the **200 OK** status code and message and that the response payload contains “Hello, world!”.



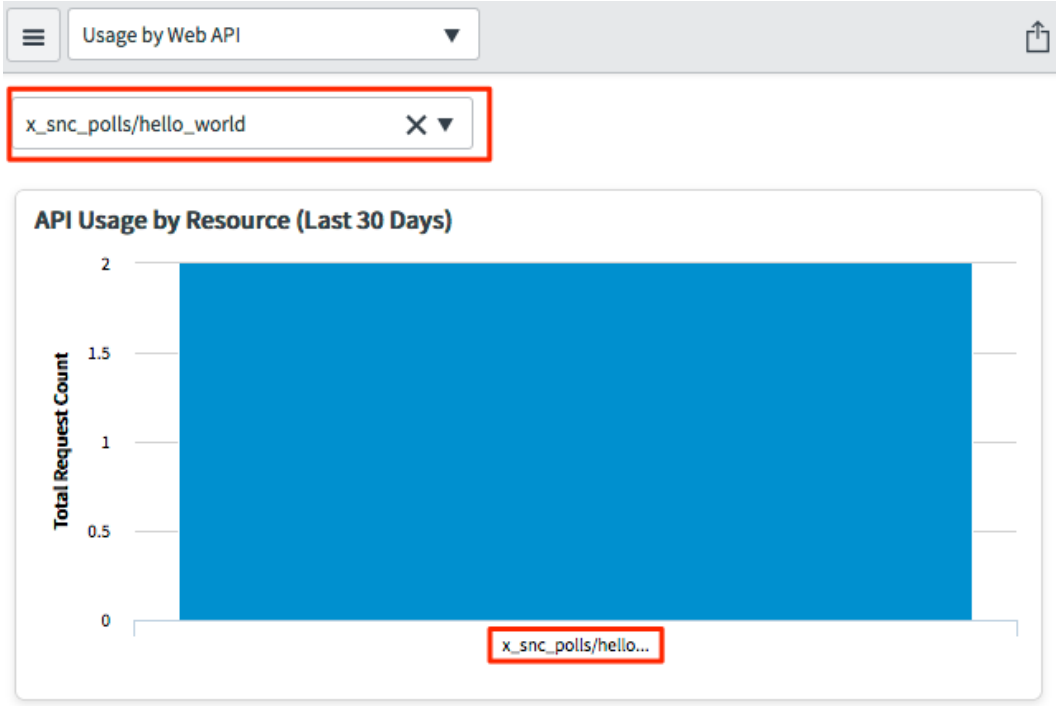
View API Analytics for Hello World

24. From **Explorer context menu**, or from Scripted REST API definition click **API Analytics**.

or



25. The **API Analytics** usage dashboard opens in a new browser tab, with the **Hello world** API pre-selected. Observe the API counts.



Note: There is up to a 60s delay between an API call and when it is reflected in API Analytics.

26. Close the REST API Explorer and API Analytics dashboard windows.

Lab 1 is complete. You are now ready to start lab 2.

Lab Goal

Lab 2
Building the Polls REST API

Having familiarized yourself with Scripted REST APIs in Lab 1, in Lab 2 we'll start building the "Polls" REST API that we'll use for the rest of this workshop. The Polls API you'll build provides a programmatic interface to interact with the Polls application on your ServiceNow instance.
The Polls app is a simple app that allows for the creation of Polls that allow participants to vote on answers to questions. Polls can have one or more questions associated with them. Questions can have one or more choices associated with them. As an example a simple poll could contain the question "What is your favorite color?". Choices that participants could choose would be; blue, red, yellow.

Create Lab 2 starting branch

1. In **Studio**, navigate to **Source Control > Create Branch**.
2. In the pop-up window, enter a branch name, then select **Lab2-start** from the **Create from Tag menu**, and click **Create Branch**.

   Branch: **my-Lab2-branch**
   Create from Tag: **Lab2-start**
3. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
4. Verify Studio is on branch **my-Lab2-branch**.
5. You are now ready to start **Lab 2**.

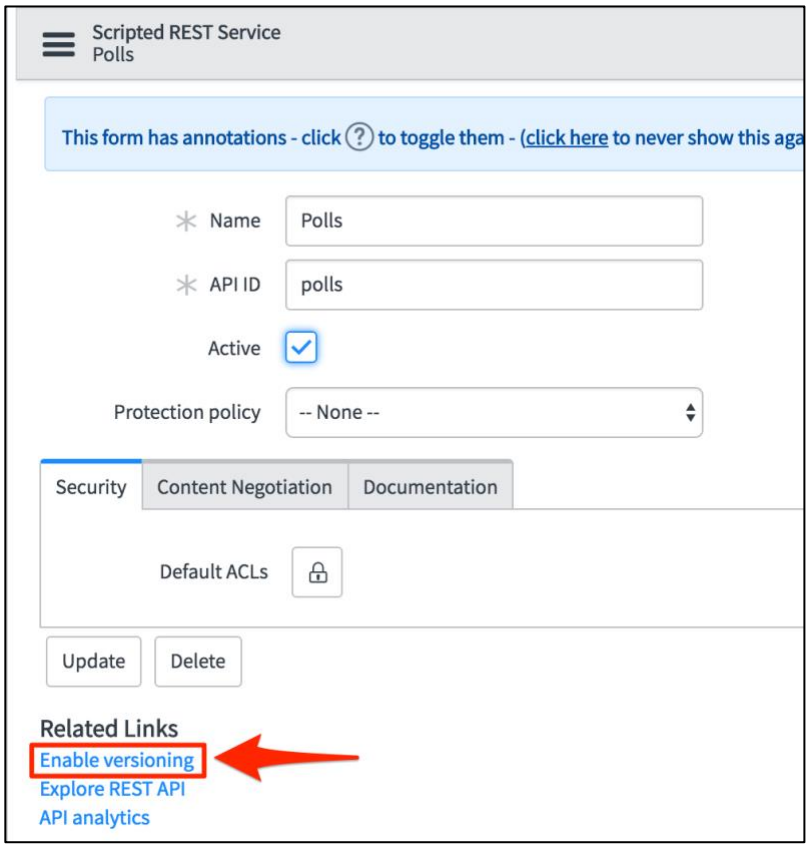
Create the Polls Scripted REST API

6. In Studio, click **Create New Application File**.
7. In the **Create New Application File** window, type **REST** in the filter then select **Scripted REST API** and click **Create**.
8. Give the **Scripted REST API** a name, then click **Submit**.

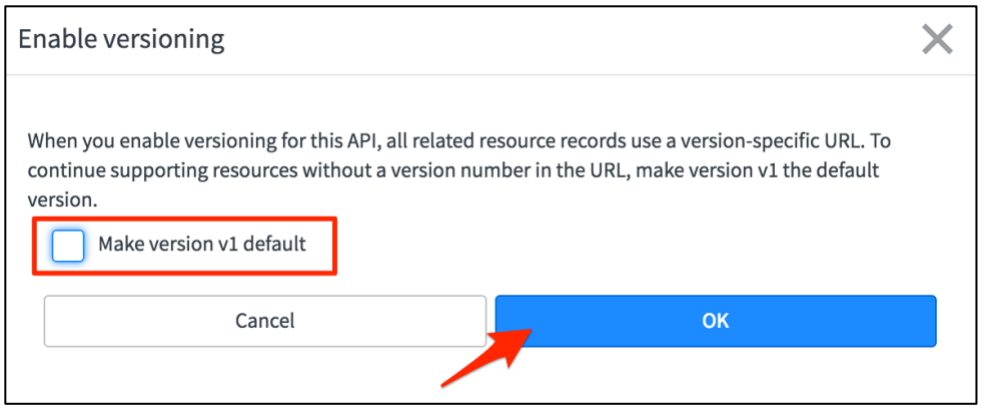
   Name: **Poll**
   API ID: **poll**

9. Click the related link **Enable versioning** to enable versioned URIs for the new API.

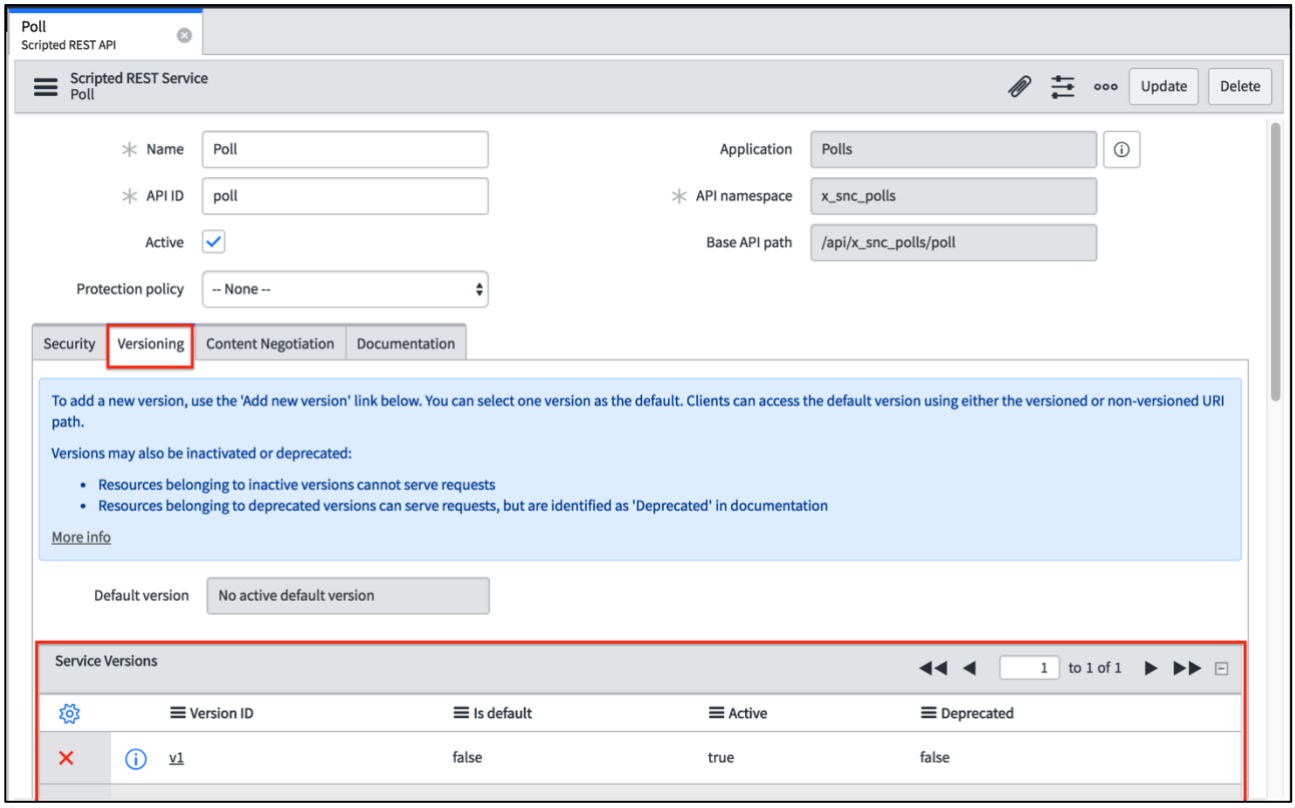
   BEST PRACTICES
   Do: *Use versioning to control API changes.*
   Do: *Encourage clients to integrate against specific versions.*
   Don't: *Make breaking changes in an existing version.*
   Do: *Release a new API version when introducing new behaviors.*



10. In the Enable versioning popup, uncheck the **Make version v1 default** checkbox, then click **OK**.



11. A new tab **Versioning** appears. Click to review the versioning tab contents.



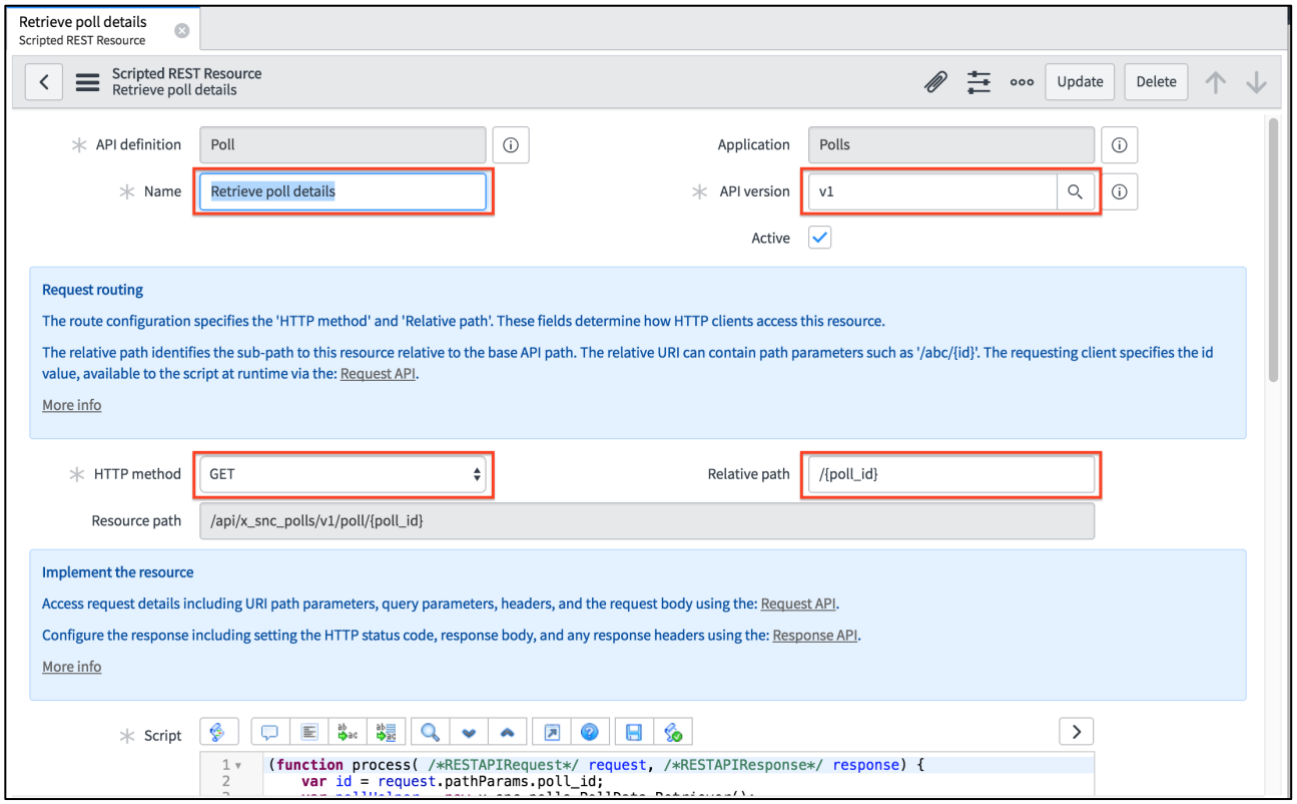
Note: The API versions are maintained here. Deactivate versions, mark a version **Is default=true** to allow non-versioned URIs to route to that version, or don't define a default version to force clients to specify the version when making requests to the API.

12. Add a **Resource** to the API. Click **New** on the **Resources** related list.

    This resource will return the details of a specific poll.

13. Specify the following properties for the new resource and complete the script.

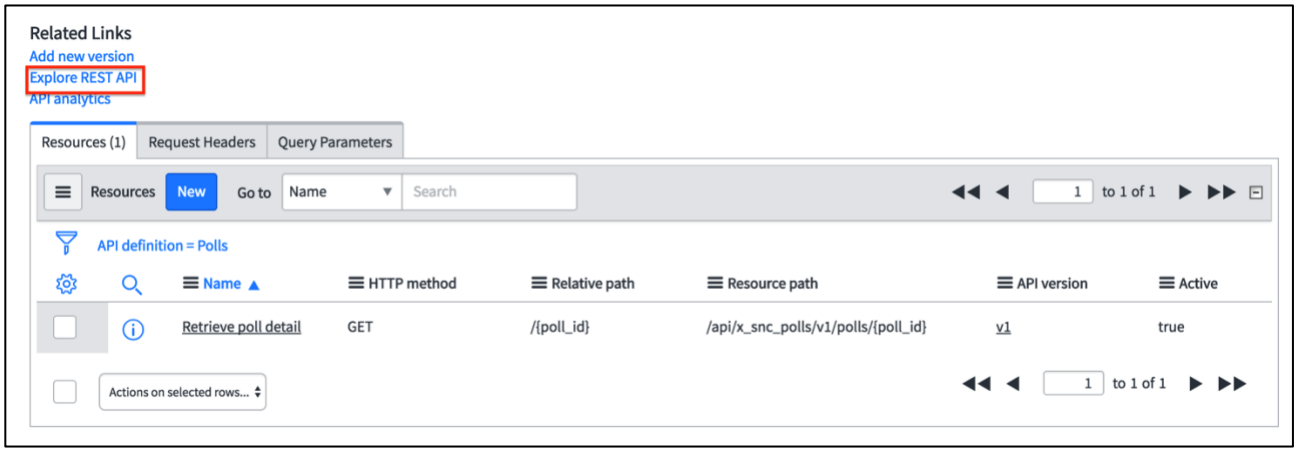
    Name: **Retrieve poll detail**
    API Version: **v1**
    HTTP method: **GET**
    Relative path: **/{poll_id}**
    Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab2_retrieve_poll_details



Click **Submit**.

Test with REST API Explorer

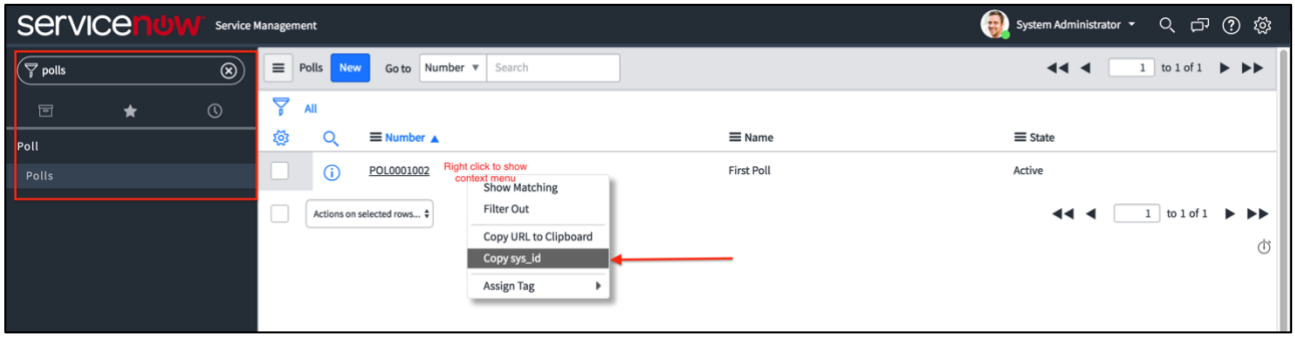
14. Click **Explore REST API**,



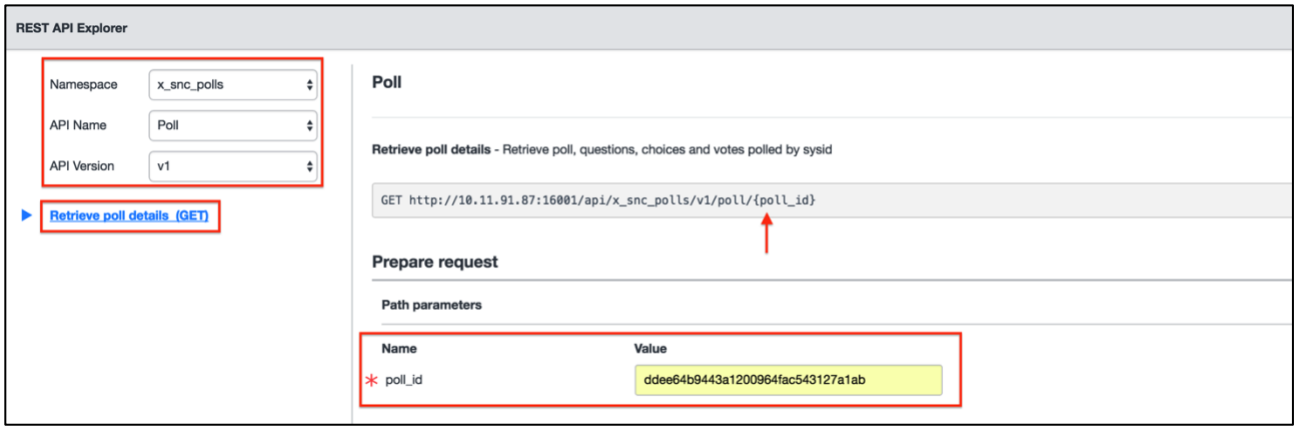
15. The **"Polls"** Scripted REST API is pre-selected in the Explorer menus and the **Retrieve Poll detail** resource is preselected.

16. Fill in sys id for a demo poll record and make a request.

    To get the sys_id of demo record. Open the Polls module from navigator. Right click on existing record to copy sys_id.



17. Fill in the sys_id on the REST API Explorer.



Click **Send**.

18. Verify the response status code is **200-OK**.

Get Caught Up

If you were unable to successfully complete the lab this far, you can "fast forward" using the following steps. Otherwise proceed to the next section **Test with Postman**.

19. Similar to creating the Lab2 starting branch, the completed lab can also be checked out from a tag (**Lab2-complete**) in Source control.
20. In **Studio**, navigate to **Source Control > Create Branch**.
21. In the pop-up window, enter a branch name, then select **Lab2-complete** from the **Create from Tag menu**, and click **Create Branch**.

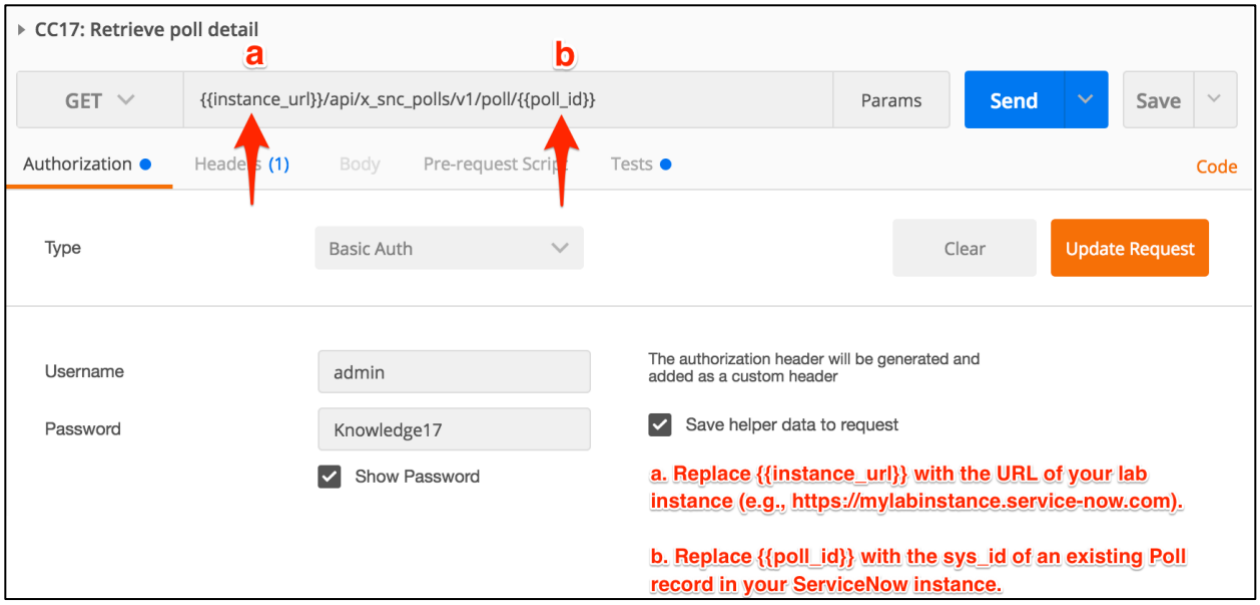
    Branch: **my-Lab2-branch-complete**
    Create from Tag: **Lab2-complete**
22. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
23. Verify Studio is on branch **my-Lab2-branch-complete**.
24. You are now ready to continue with the next section of **Lab 2**.

Test with Postman

So far in this lab you've used Postman to make requests to ServiceNow REST APIs, Postman also allows you to write and execute tests that evaluate response from a REST API and provide you with Pass/Fail information based on your test and the response from the request. Let's issue a request against the new resource **Retrieve poll detail** and write a few tests to verify the response we receive.

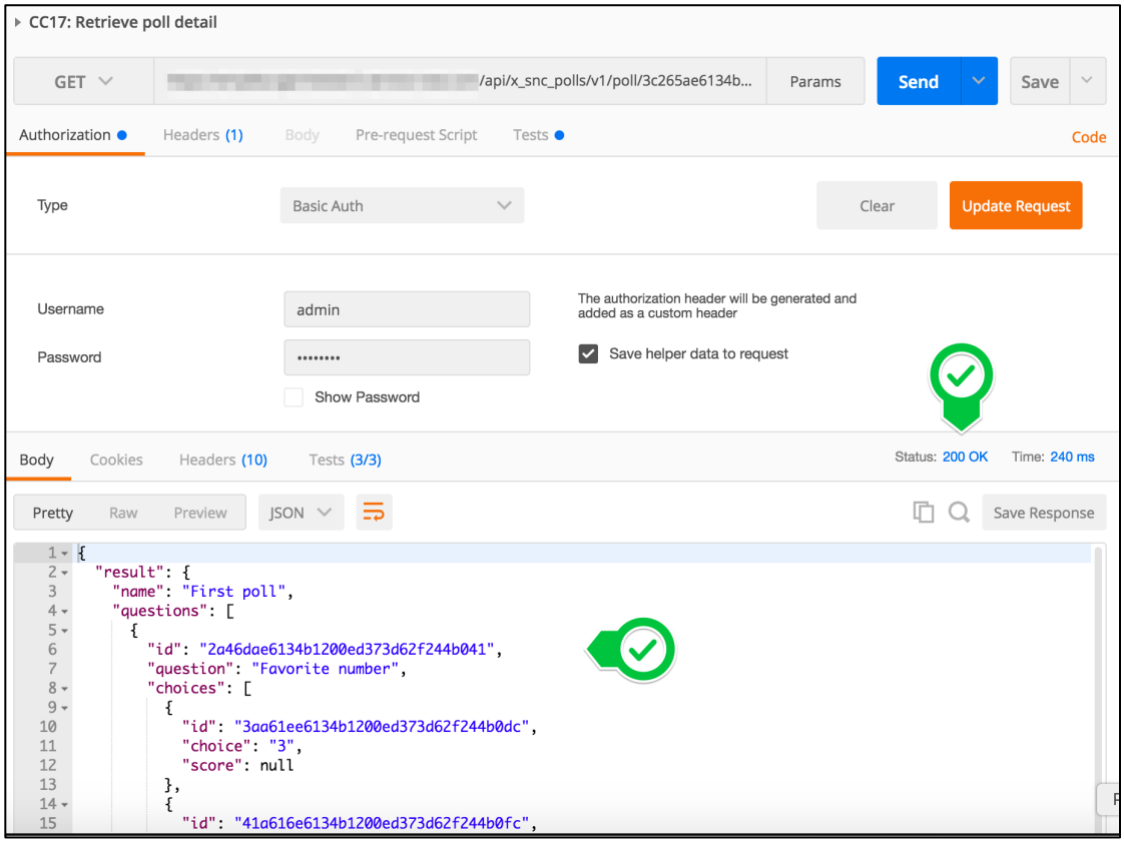
25. In Postman select the **CC17: Retrieve poll detail** request in the Scripted REST API Workshop collection. This request has been pre-built for you however you will need to update the {{instance_url}} and path including {{poll_id}} parameters in the URL replacing them with values from your lab instance. You will also need to update the Authorization section specifying your username and password.

    - instance_url: **URL of your lab instance**
    - poll_id: **Sys_id of a poll record in your lab instance**
    - Username: admin
    - Password: admin password for your lab instance



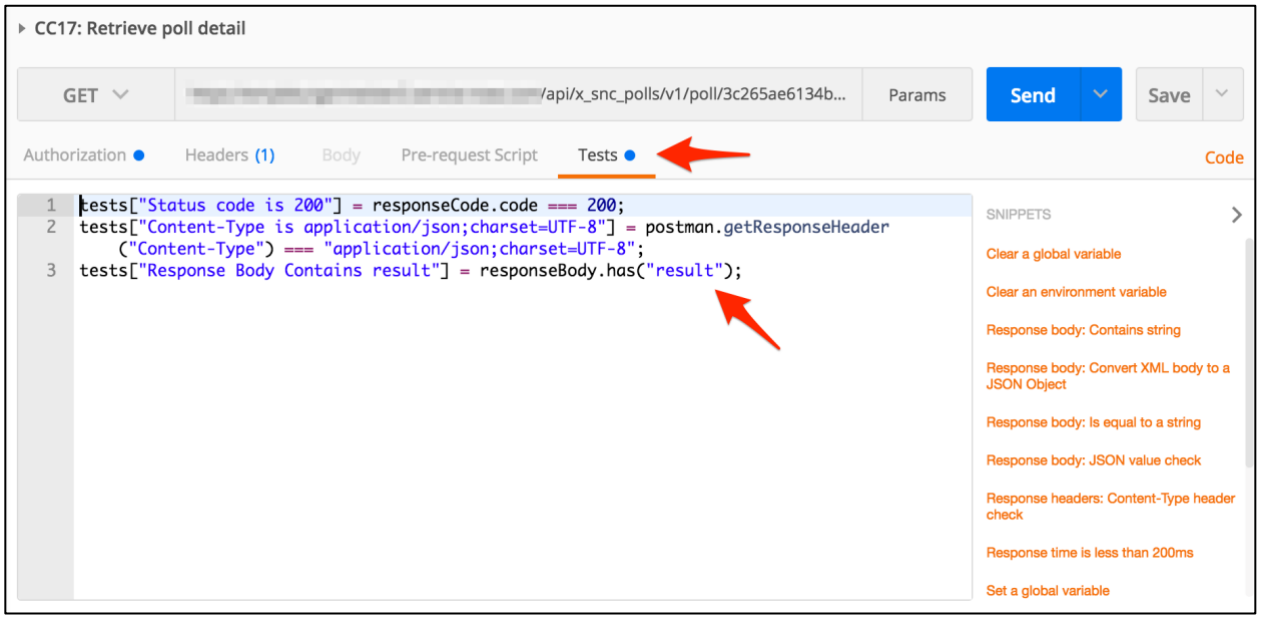
26. After populating your credentials and replacing the parameters click **Update Request** and then **Send** the request.

27. Check that you've received a successful response. You should see a status of **200 OK** and a JSON payload that includes at least one poll as shown below.



28. This request should return a status code of **200 OK**, with a JSON payload that represents the poll we requested. In addition the content-type header in the response should be **application/json;charset=UTF-8** and our JSON payload should contain a **result** object. Let's see how we can use Postman to verify this for us with tests that will be run as part of the request.

29. In Postman, in the CC:17 Retrieve poll detail request, open the **Tests** tab by clicking on **Tests.** Here you can specify tests that will be run as part of each request.



30. Postman has its own simple syntax for declaring tests. You can find out more about this syntax at the Postman website. For this lab we've provided you with 3 tests that validate that:

    - the response status code is **200**
    - the response includes a content-type header with a value of **application/json;charset=UTF-8**
    - the response body contains the text '**result**'

31. Now update your request in Postman to run these tests. Copy the test script from the following URL and paste it into the **Tests** area in Postman.
    - Postman test script: http://bit.ly/CC17_ScriptedRESTAPI_Lab2_postman_test_script

32. After copying click **Send** to issue the request. Now that we have tests specified as part of our request test results will be displayed in the response area. If all tests passed you will see a '**(3/3)**' in the header and then a green **PASS** image next to each test as shown below.

Save your request in Postman. You now have a saved request in Postman that allows you to easily issue a request to your 'Retrieve poll detail' resource and which will run test the response to validate that it includes the correct status code, header, and payload content. These were simple test cases but Postman will allow you to define more advanced test cases to verify you are receiving the correct response from your REST API.

BEST PRACTICES

Do: *Define test cases for each of your APIs resources to validate that the response is formatted correctly and that the response contains the intended content. Building test cases as part of your development process will help insure you're building the API as you designed it and provide you with a set of tests that can be run over time as you make changes to guarantee that your interface has not changed unintentionally.*

33. Close the REST API Explorer and API Analytics dashboard windows.

Lab 2 is complete. You are now ready to begin lab 3.

Lab Goal

Lab 3
Request & Response API

In Lab 3 you'll continue building out the REST API for the **Polls** application adding resources to support creating a new poll, editing an existing poll, and voting in a poll. In building out this additional functionality you will further use and familiarize yourself with the Request and Response APIs that allow you to interact with the request that your REST API receives and build the response that your REST API will return.

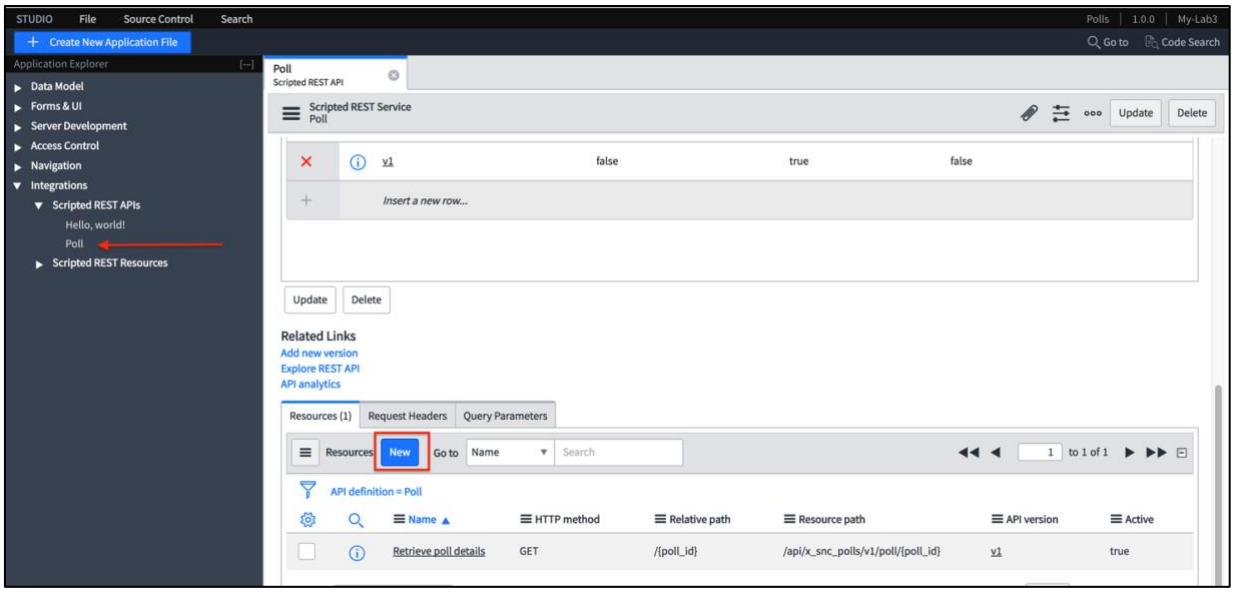
Create the Lab 3 starting branch

1. In Studio, navigate to Source Control > Create Branch.
2. In the pop-up window, enter a branch name, then select Lab3-start from the Create from **Tag menu**, and click **Create Branch**.

   Branch: **my-Lab3-branch**
   Create from Tag: **Lab3-start**
3. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
4. Verify Studio is on branch **my-Lab3-branch**.
5. You are now ready to start **Lab 3**.

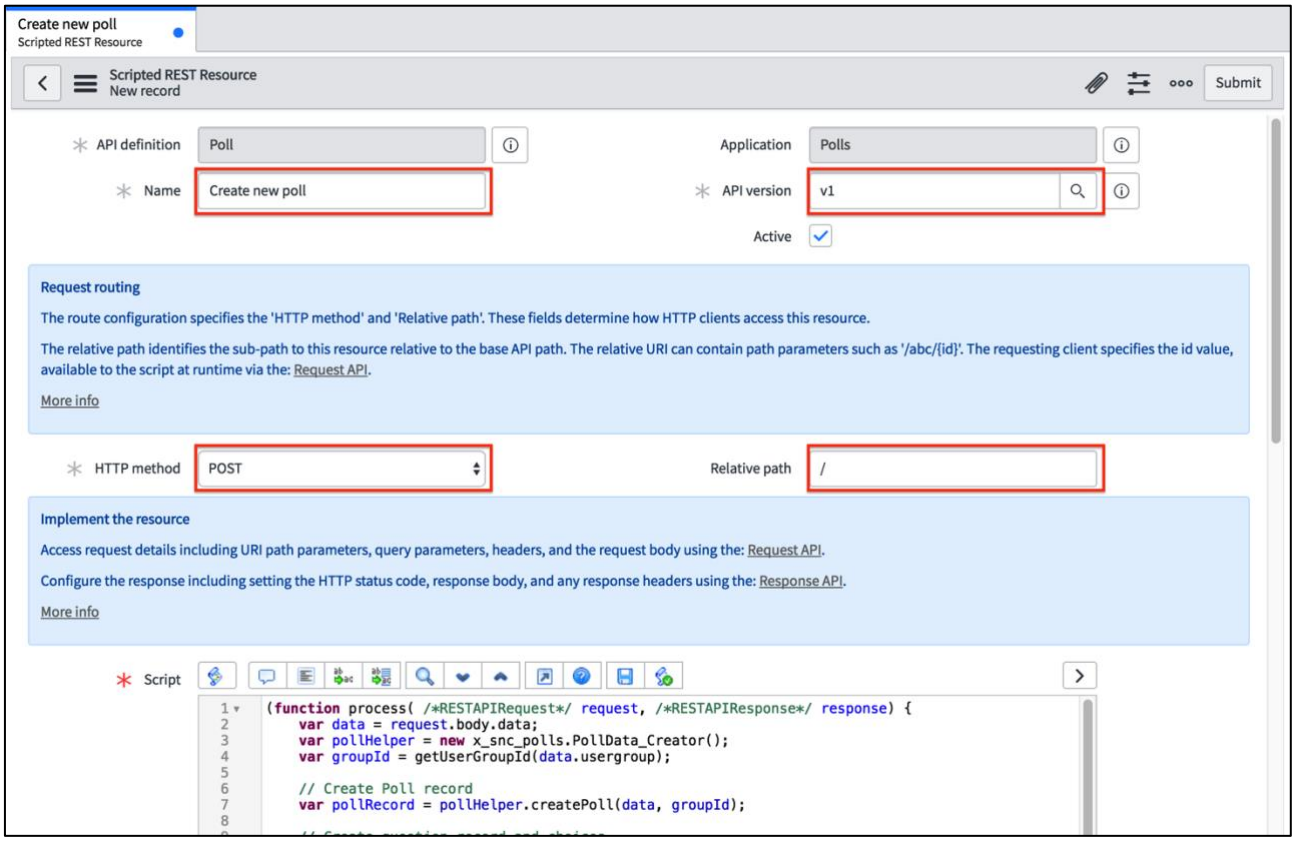
Create New Resource in Polls API - Create a poll

6. The ‘Create a poll’ resource will be used to create a new poll in the ‘Polls’ application.
   Open the Polls API in studio and click **New** on the **Resources** related list to create a new resource.



7. Specify the following properties for the new resource.

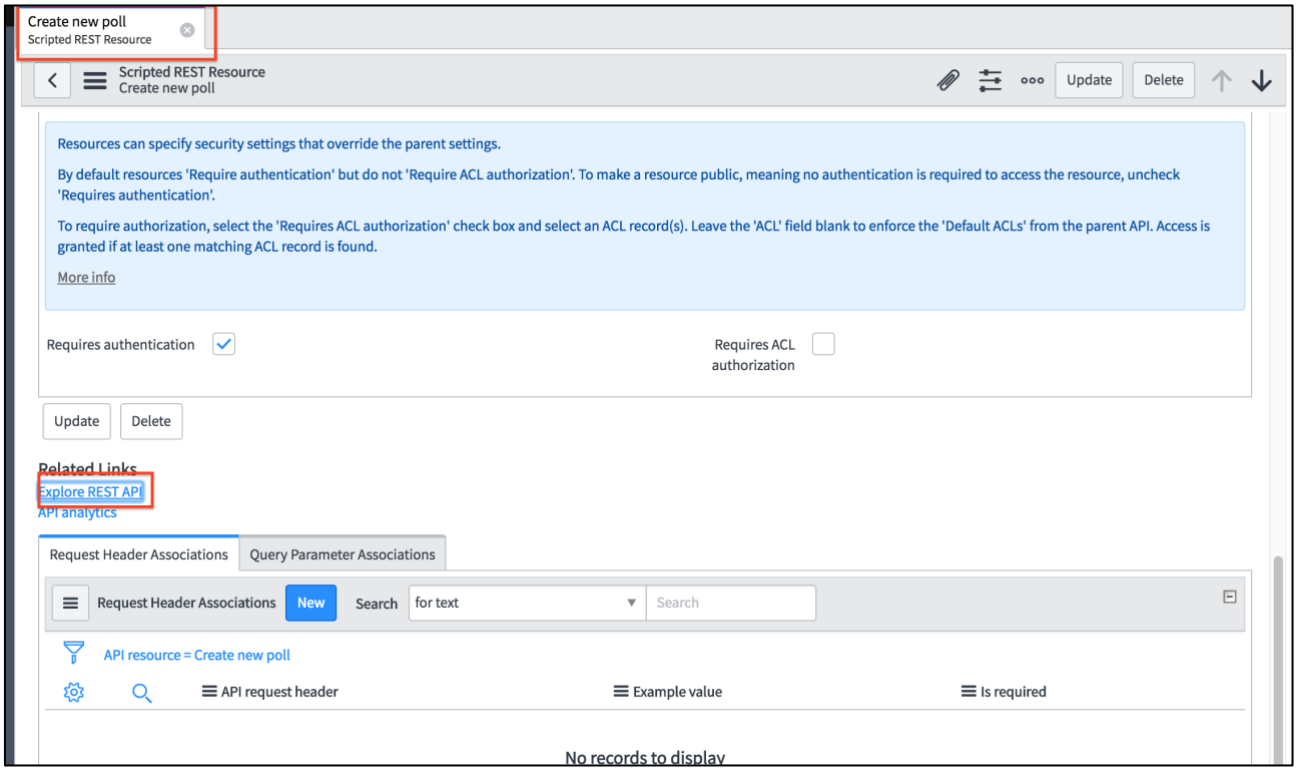
   Name: **Create new poll**
   API Version: **v1**
   HTTP method: **POST**
   Relative path: **/**
   Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab3_create_new_poll



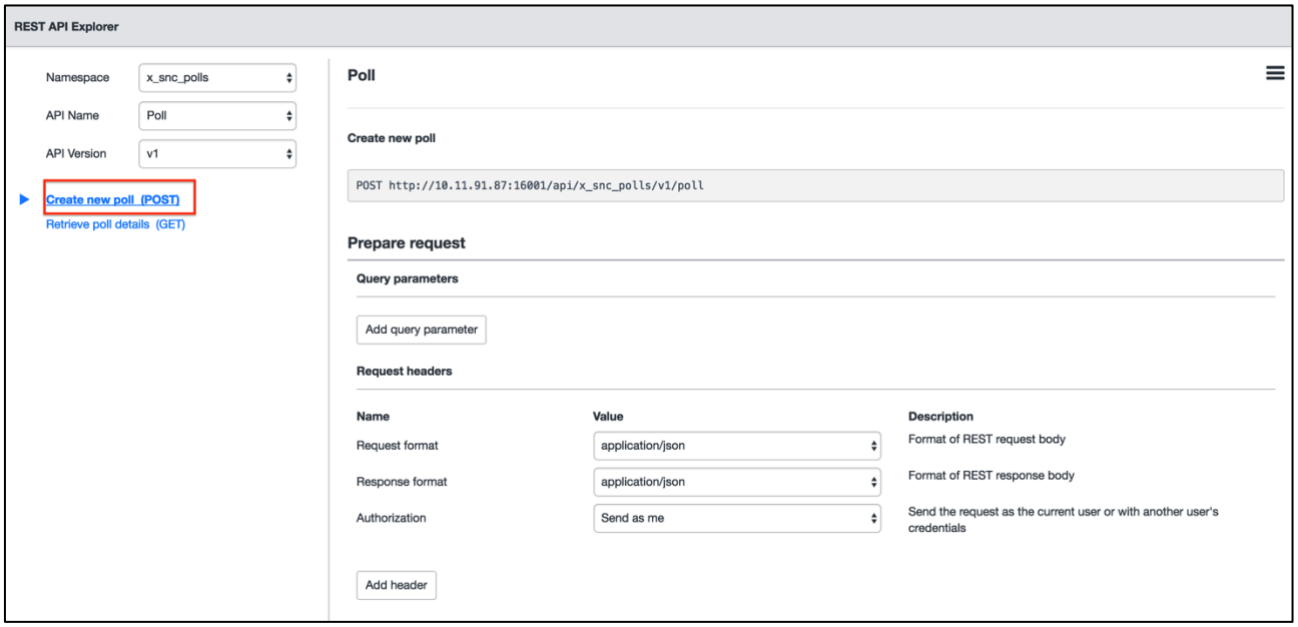
Click **Submit**.

Test with REST API Explorer

8. Open ‘Create new poll’ resource and Click **Explore REST API**.



9. **Create New poll** resource is shown in the REST API Explorer. Fill in request body in the raw tab under Request body section.

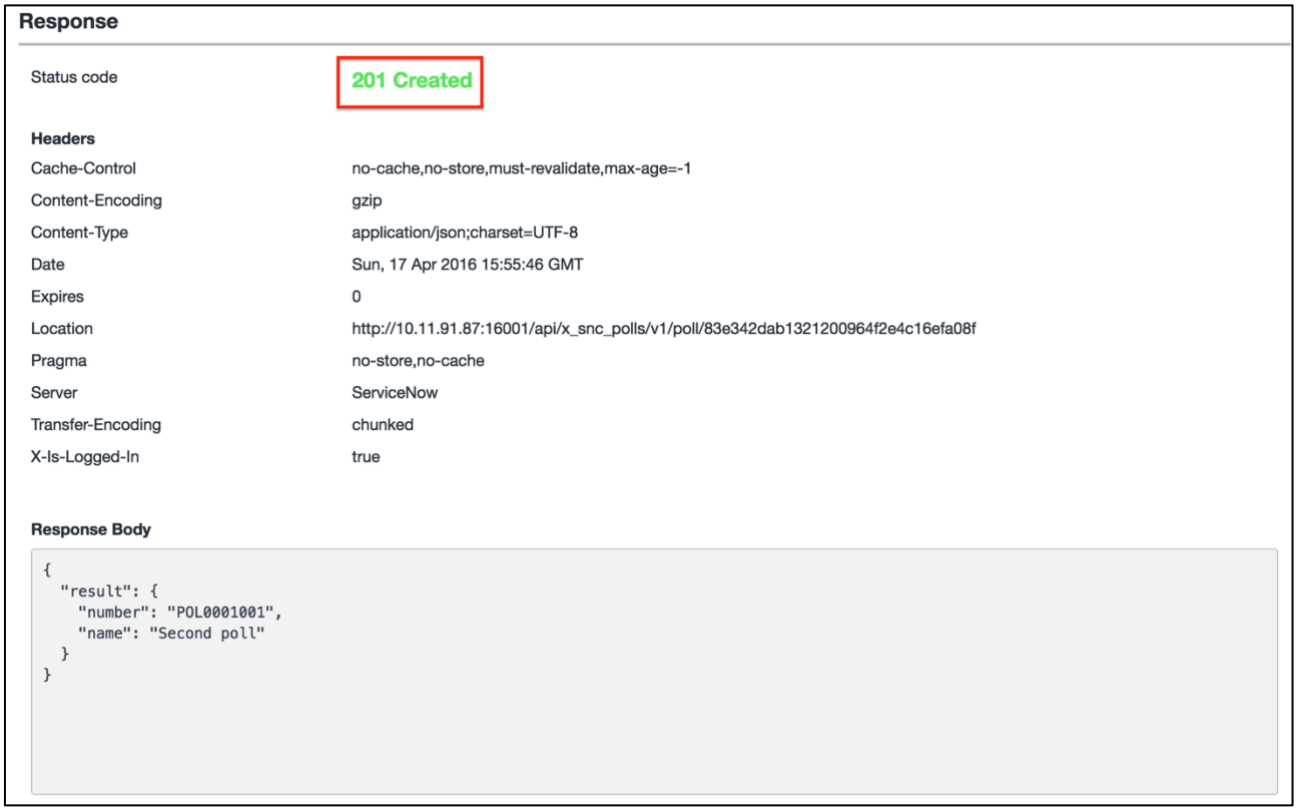


A sample request payload can be found at:
http://bit.ly/CC17_ScriptedRESTAPI_Lab3_create_new_poll_sample_request

10. Copy the sample payload into the 'Raw' tab.

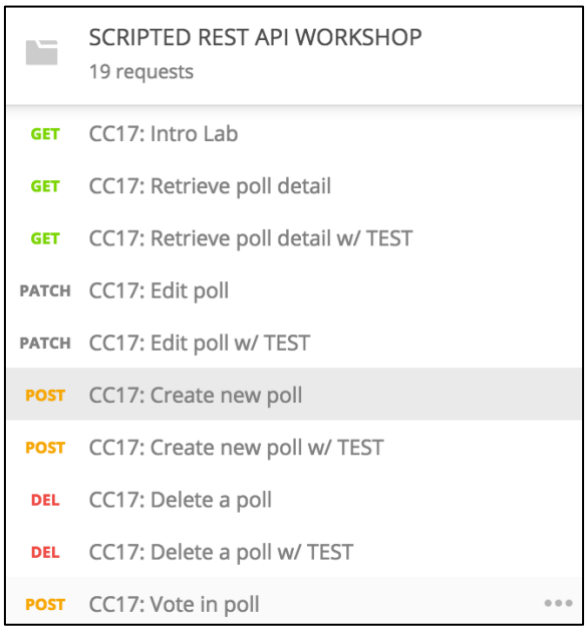
Click **Send**.

11. Verify response status code is **201 Created**



Create tests in Postman

12. In Postman select the 'CC17: Create new poll' request'. This is a pre-built request that already contains an appropriately formatted payload for the 'Create new poll' resource.



13. Update the request replacing the {{instance_url}} and authorization credentials appropriate for your lab instance. Use your admin credentials for this request. Once you've updated those values save and then send the request.

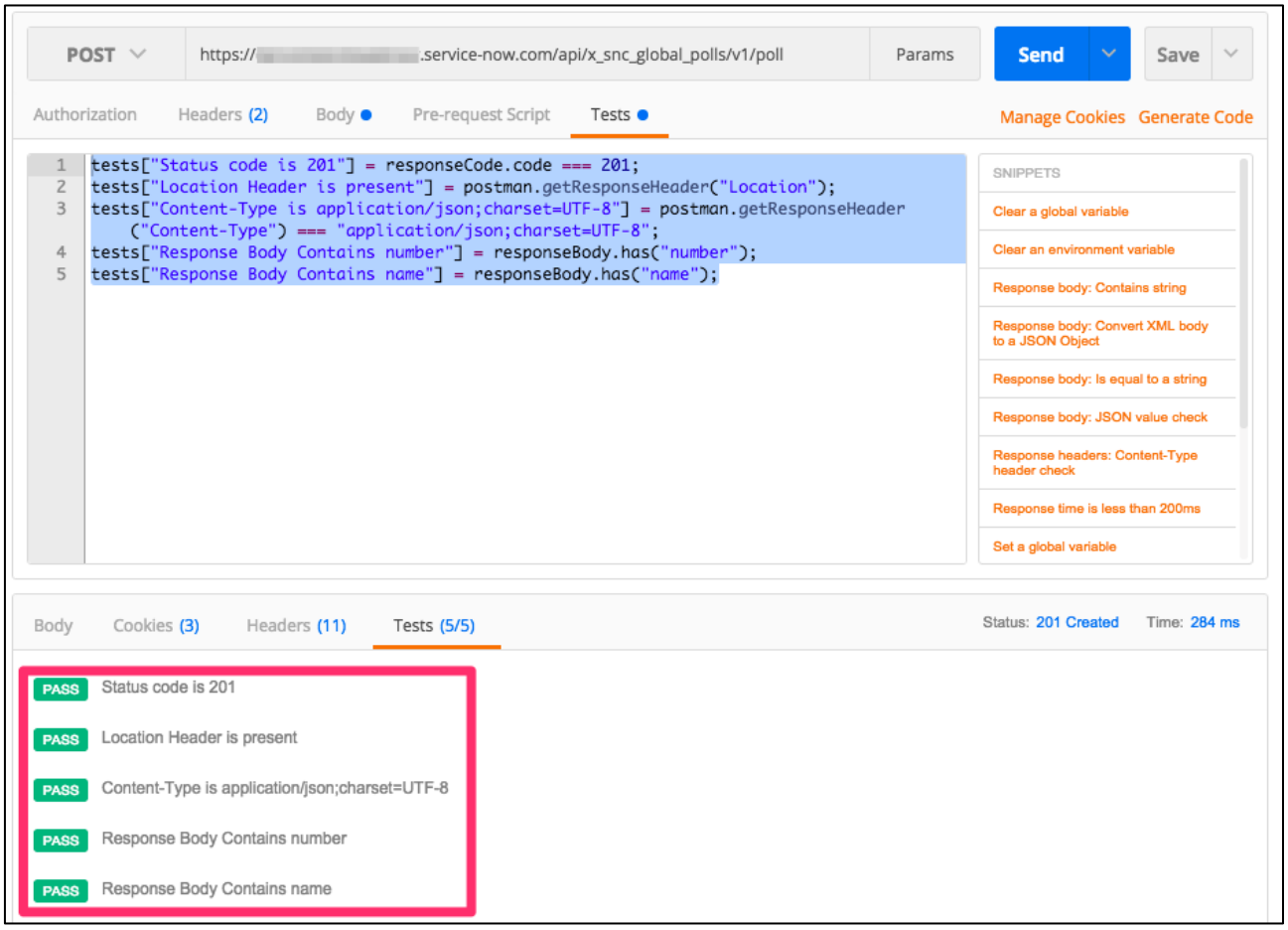
14. As you saw when you tested with the REST API Explorer a successful response will include a **201** status code, a JSON payload that includes the number for the newly created poll, and the response headers include a 'Location' header that provides the URL for this newly created record. Let's add tests in Postman that verify that the following details for in the response:
    - Response status code is 201
    - Response headers include Location
    - Response headers include Content-Type of application/json;charset=UTF-8
    - Response body contains the text 'number'
    - Response body contains the text 'name'

    Update the request in Postman to include the following:

```
tests["Status code is 201"] = responseCode.code === 201;

tests["Location Header is present"] = postman.getResponseHeader("Location");

tests["Response Body Contains number"] = responseBody.has("number");

tests["Response Body Contains name"] = responseBody.has("name");
    tests["Content-Type is application/json;charset=UTF-8"] = postman.getResponseHeader("Content-Type") ===
    "application/json;charset=UTF-8";
```

For ease you can also copy these from:
http://bit.ly/CC17_ScriptedRESTAPI_Lab3_create_new_poll_test_script

15. After adding the tests above save your request in Postman and Send the request. In the response you should see the following tests and results.

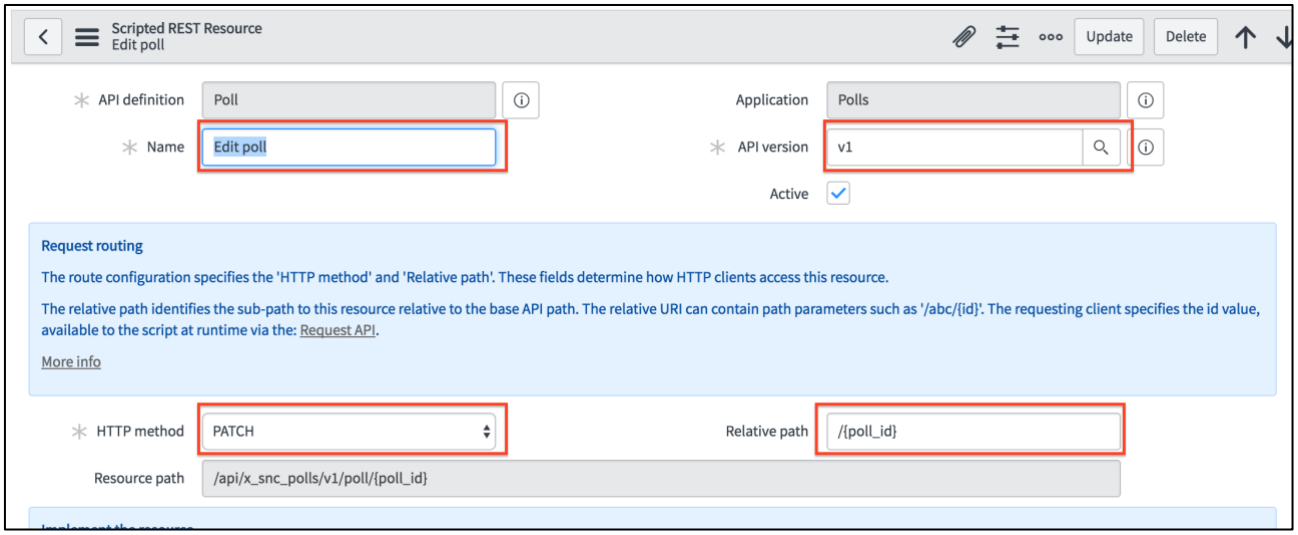


Create New Resource in Polls API – Edit poll

The ‘Edit poll’ resource will be used to modify an existing poll record in the ‘Polls’ application.

16. Open the Polls API in studio and click New on the Resources related list to create a new resource.
17. Specify the following properties for the new resource.

    Name: **Edit poll**
    API Version: **v1**
    HTTP method: **PATCH**
    Relative path: **/{poll_id}**
    Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab3_edit_poll

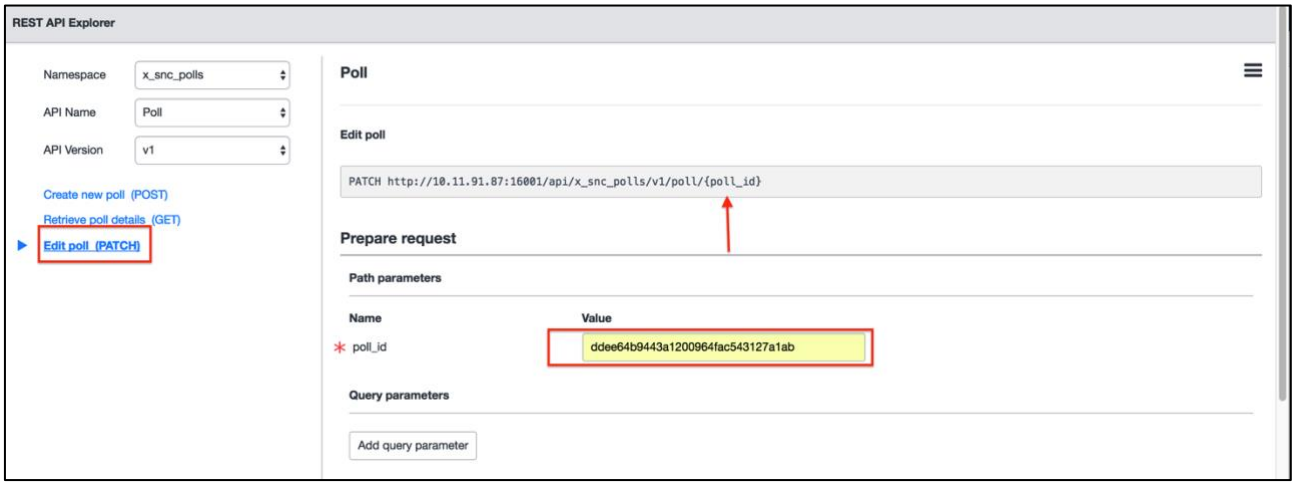


    Click **Submit**.

Test with REST API Explorer

Open ‘Edit poll’ resource and Click **Explore REST API** in related actions.

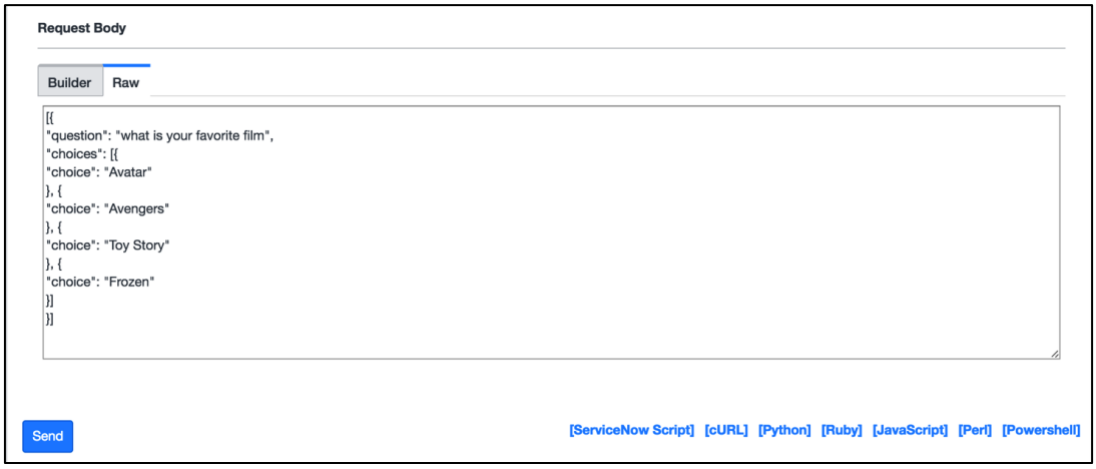
18. **Edit poll** resource is preselected in API Explorer.



a. Fill in request body in raw tab under Request body section.

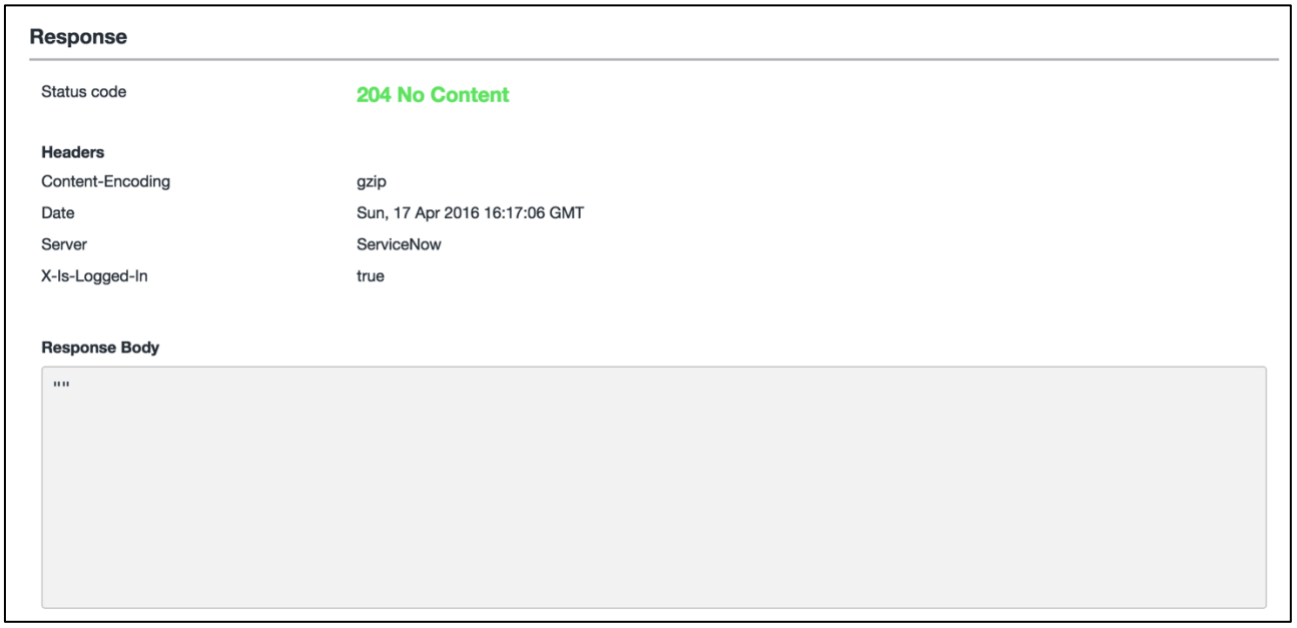
   A sample request payload can be found at:
   http://bit.ly/CC17_ScriptedRESTAPI_Lab3_edit_poll_sample_request

b. Copy the sample payload into the ‘Raw’ tab.



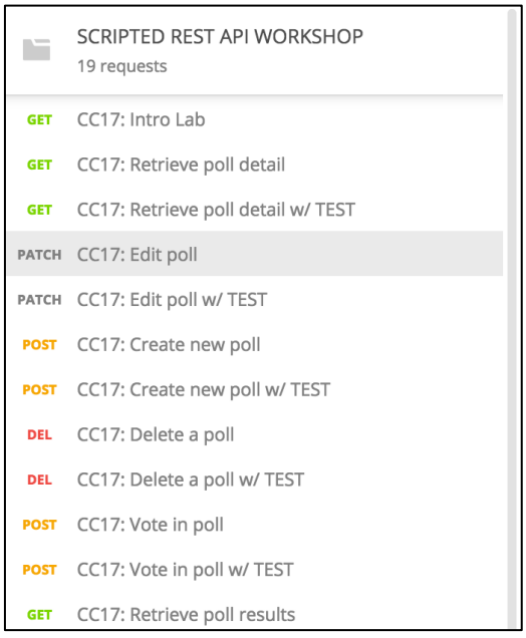
Click **Send**.

19. Verify response status code is **204-No content.**



Create tests in Postman

20. In Postman select the 'CC17:Edit poll' request. This is a pre-built request that already contains an appropriately formatted payload for calling the 'Edit poll' resource.



21. Update the request replacing the {{instance_url}}, authorization credentials, and {{poll_id}} with values appropriate for your lab instance. Use your admin credentials for this request. Once you've updated those values save and then send the request.

As you saw when you tested with the REST API Explorer a successful response will include a **204** status code and an empty payload. Let's add tests in Postman that verify that the following details in the response:

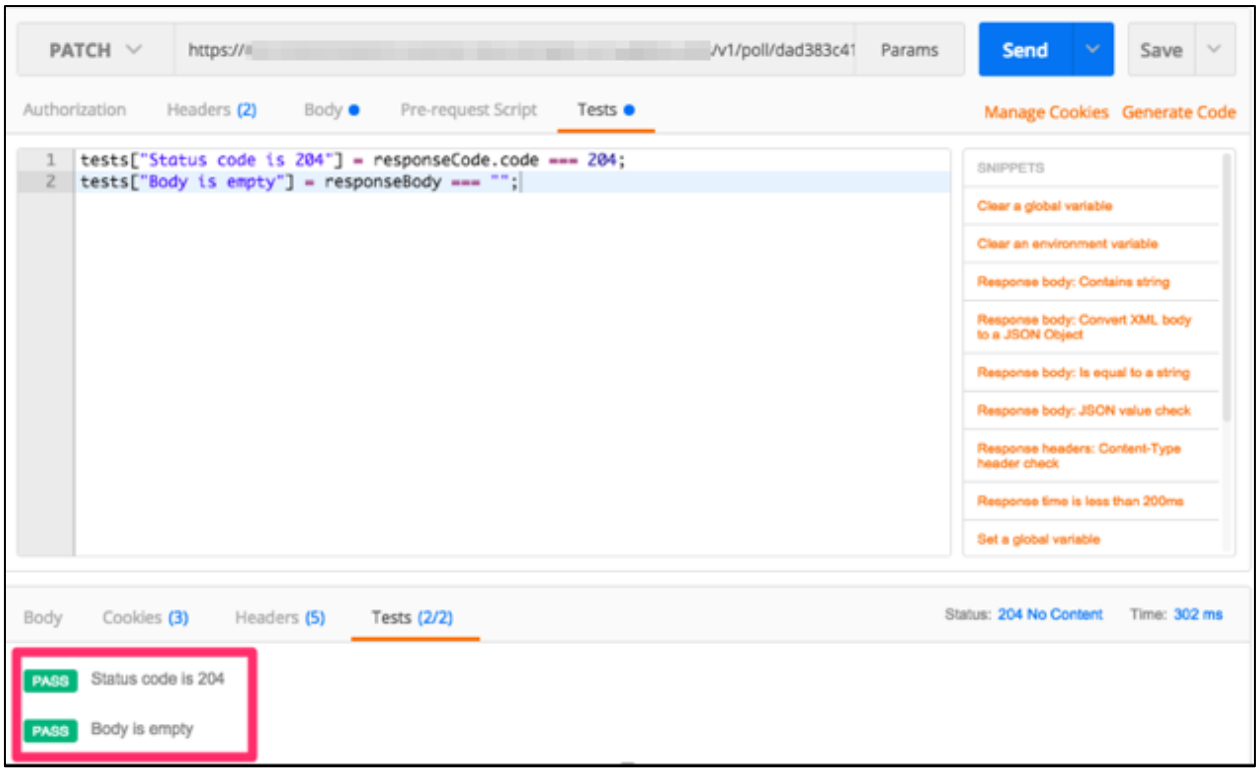
- Response status code is 204
- Response payload is empty

22. Update the request in Postman to include the following:

```
tests["Status code is 204"] = responseCode.code === 204;

tests["Body is empty"] = responseBody === "";
```

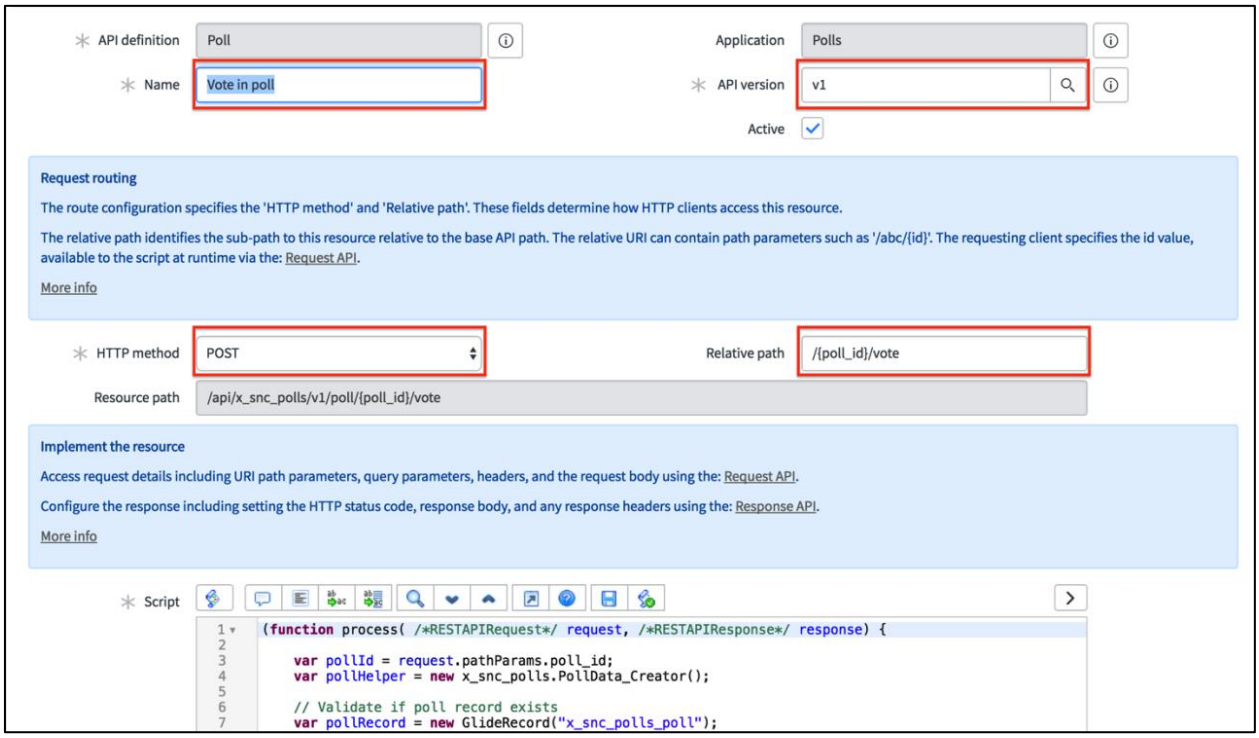
For ease you can also copy these from:
http://bit.ly/CC17_ScriptedRESTAPI_Lab3_edit_poll_test_script

23. After adding the tests above save your request in Postman and Send the request. In the response you should see the following tests and results.

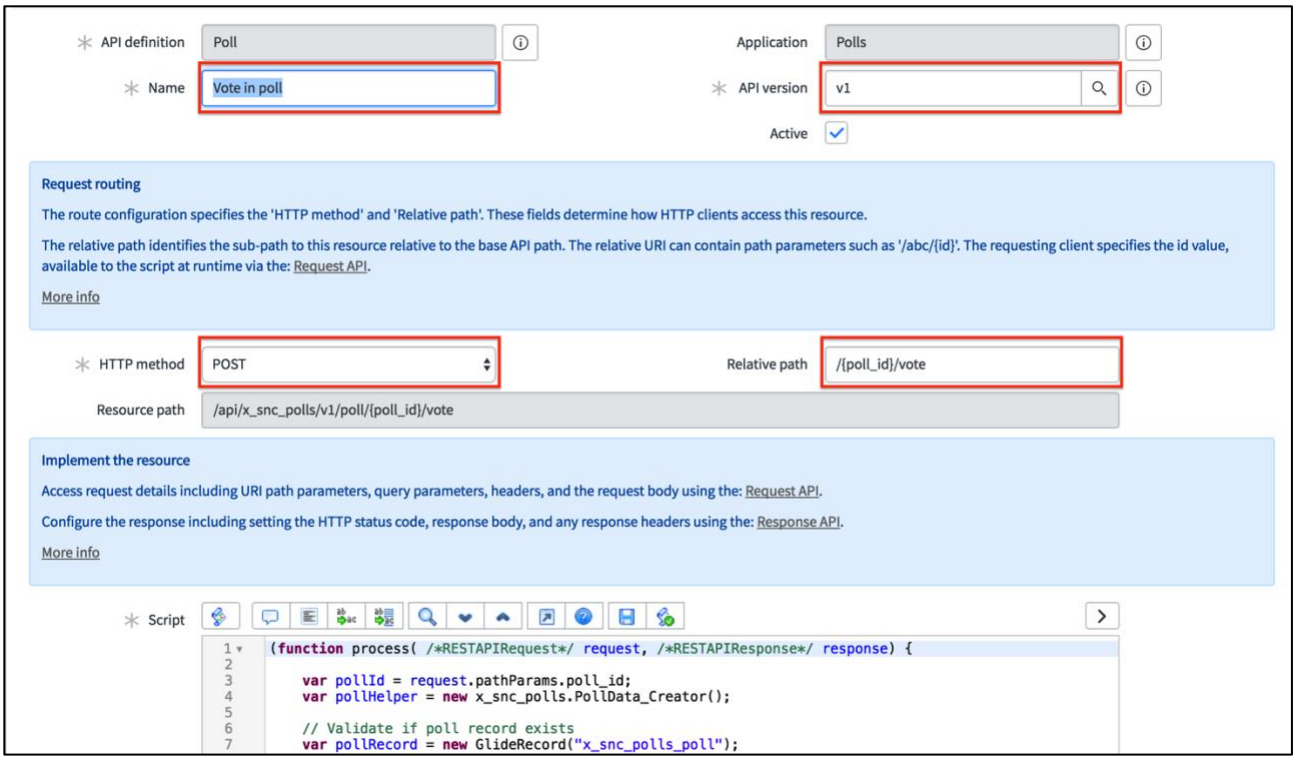


Create New Resource in Polls API – Vote in poll

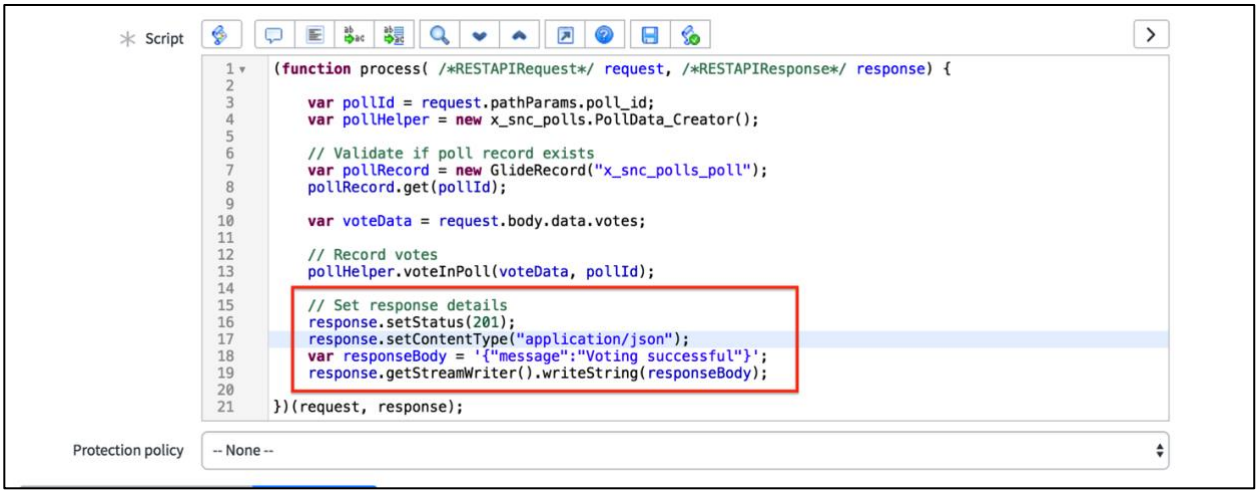
24. The 'Vote in poll' resource will be used to cast a vote for an answer to a specific question or set of questions that are part of a poll in the 'Polls' application. Open Polls API in studio and add a **Resource** to the API. Click **New** on the **Resources** related list.



Specify the following properties for the new resource.
Name: **Vote in poll**
API Version: **v1**
HTTP method: **POST**
Relative path: **/{poll_id}/vote**
Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab3_vote_in_poll



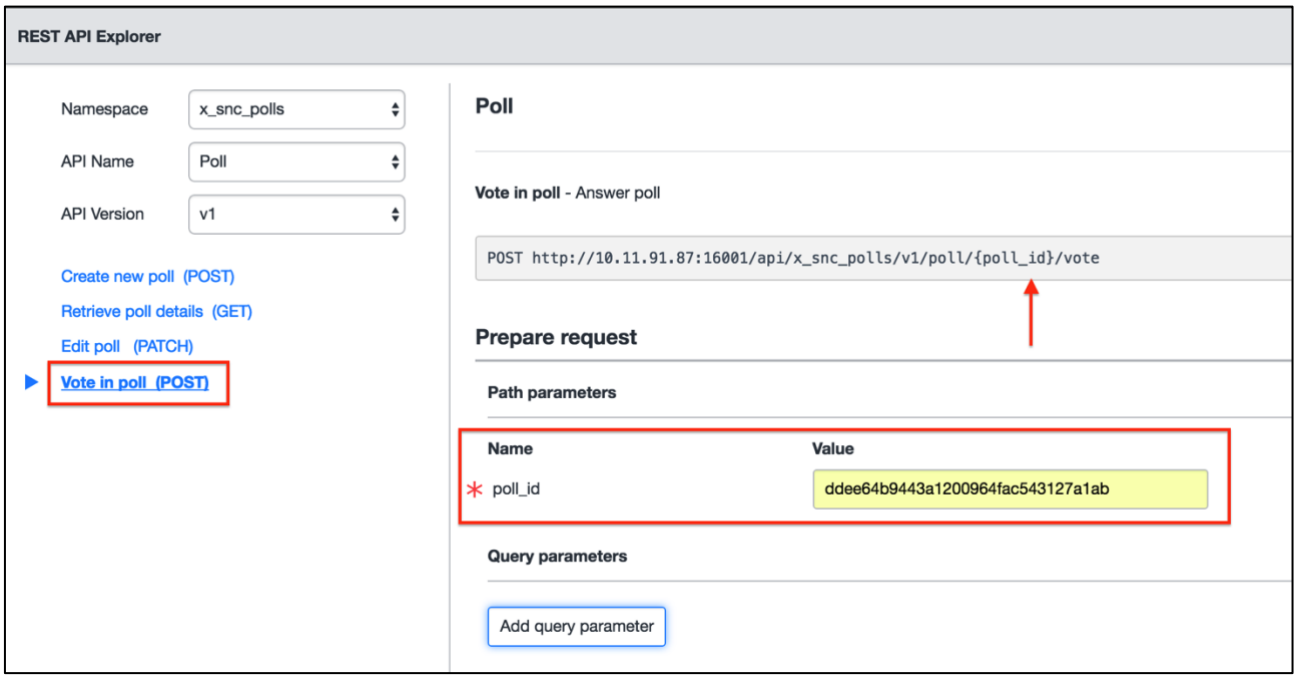
NOTE: Observe the custom response string being written to the response using the 'getStreamWriter' method. The getStreamWriter method is used to produce a custom response in Scripted REST APIs and allows you (the API creator) to precisely specify the format of the response. It is important to set content type and status code if writing to stream directly.



Click **Submit**.

Test with REST API Explorer

25. Open ‘Vote in poll’ resource and Click **Explore REST API** in related actions.
26. **Vote in poll** resource is preselected in API Explorer. Fill in request body in raw tab under Request body section.

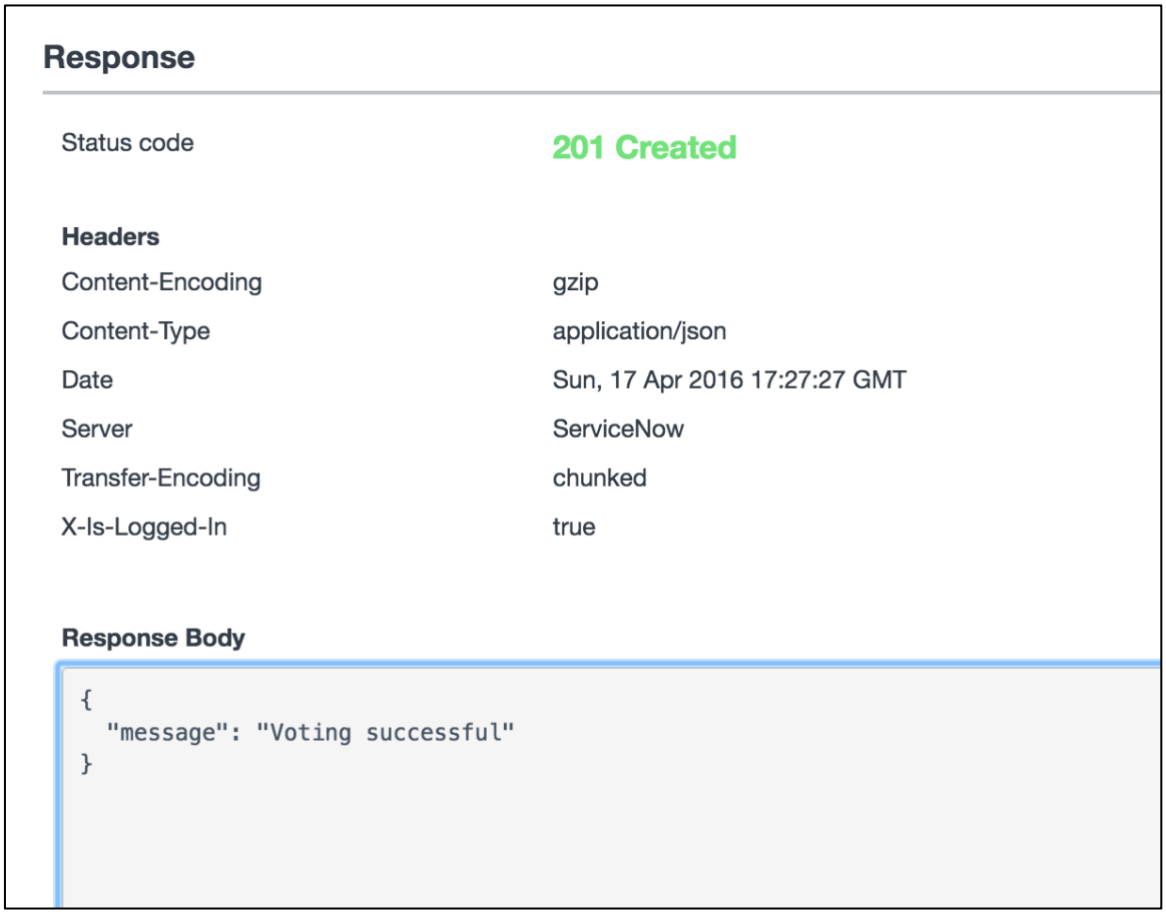


A sample request payload can be found at:
http://bit.ly/CC17_ScriptedRESTAPI_Lab3_vote_in_poll_sample_request

NOTE: you will need to update the ‘poll_id’ to be that of a specific poll that exists in the Polls application on your lab instance.

Click **Send**.

27. Verify response status code is **201 Created.**

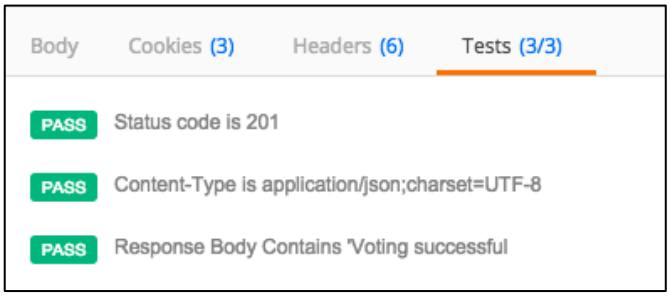


Create tests in Postman

27. In Postman select the 'CC17:Vote in poll' request. This is a pre-built request that already contains an appropriately formatted payload for calling the 'Vote in poll' resource.
28. Update the request replacing the {{instance_url}}, authorization credentials, and {{poll_id}} with values appropriate for your lab instance. Use your admin credentials for this request. Once you've updated those values save and then send the request.
29. As you saw when you tested with the REST API Explorer a successful response will include a **201** status code and an JSON payload informing you that voting was successful. Add tests in Postman that verify that the following details in the response:

    - Response status code is 201
    - Response headers include Content-Type of application/json;charset=UTF-8
    - Response body contains the text: "Voting successful"

30. You are on your own to create these tests in Postman. You can refer back to the tests you've created in the previous steps for help.
31. Once you've added the tests save the request and send it. If you were successful you should see all the tests passing.



Note: If you are really stuck here you can refer to the pre-built request in the Postman collection named "CC17: Vote in poll w/ TEST" to see this request with tests fully specified.

Get Caught Up

If you were unable to successfully complete the lab this far, you can “fast forward” using the following steps. Otherwise proceed to the next section **Test with Postman**.

32. Similar to creating the Lab3 starting branch, the completed lab can also be checked out from a tag (**Lab3-complete**) in Source control.
33. In **Studio**, navigate to **Source Control > Create Branch**.
34. In the pop-up window, enter a branch name, then select **Lab3-complete** from the **Create from Tag menu**, and click **Create Branch**.

    Branch: **my-Lab3-branch-complete**
    Create from Tag: **Lab3-complete**
35. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
36. Verify Studio is on branch **my-Lab3-branch-complete**.

Lab 3 is complete. You are now ready to begin lab 4.

Lab Goal

Lab 4
Enforcing Security

In Lab 4 you'll continue building out the REST API for the **Polls** application adding resources to support retrieving the results of a poll which includes details of individuals votes as well as the ability to delete a poll. These operations expose functionality that should be restricted to users with an additional role so that we can limit access to see how individual users voted as well as be able to delete polls.

Scripted REST APIs allow you to specify ACLs that requestors must have to be able to make a request both at the API and Resource level. These ACLs can then be associated users or groups via the standard access control mechanism in ServiceNow.

Scripted REST APIs allow you to configure, at both the API and Resource level, if a requestor needs to **authenticate** (via Basic Auth or OAuth2.0) to ServiceNow to make requests. In addition, you can configure if the requestor must be authorized, via specific ACLs, to make a request to your API.

In building out these additional resources you will familiarize yourself with how you can use the security features of Scripted REST APIs to secure your REST API.

Create Lab 4 starting branch

1. In **Studio**, navigate to **Source Control > Create Branch**.
2. In the pop-up window, enter a branch name, then select **Lab4-start** from the **Create from Tag menu**, and click **Create Branch**.

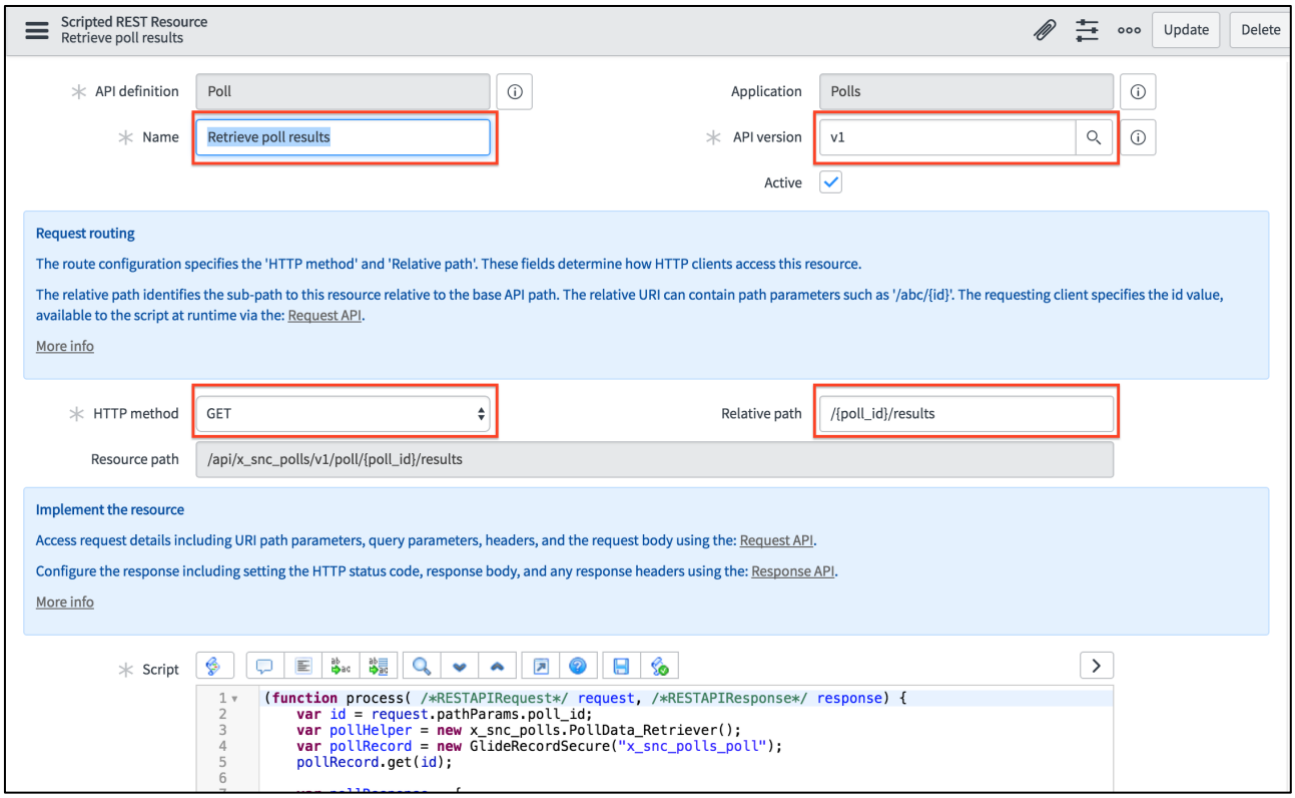
   Branch: **my-Lab4-branch**
   Create from Tag: **Lab4-start**
3. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
4. Verify Studio is on branch **my-Lab4-branch**.
5. You are now ready to start **Lab 4**.

Create New Resource in Polls API – Retrieve poll results

6. Open Polls API from studio. Add a **Resource** to the API. Click **New** on the **Resources** related list.

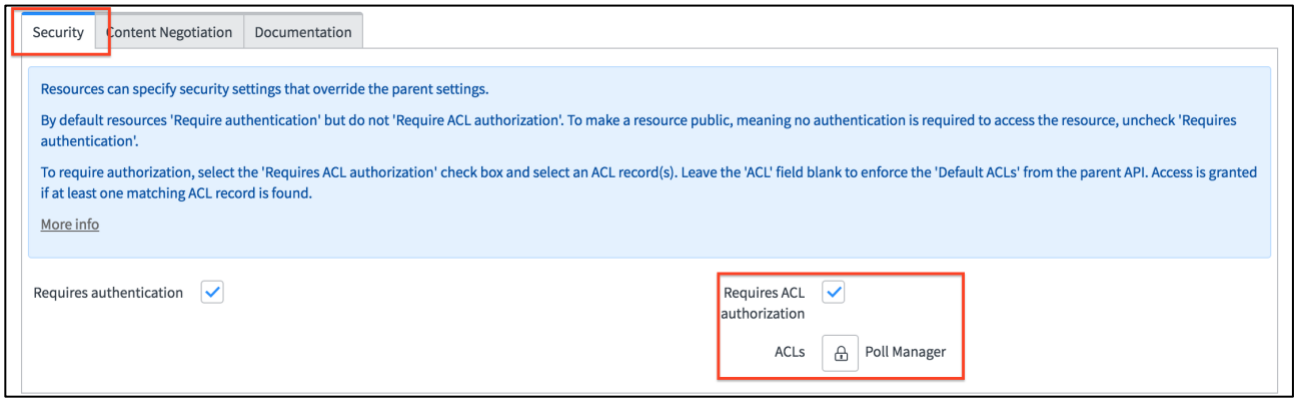
   Give the resource a **Name**. Complete the script.
   Name: **Retrieve poll results**
   API Version: **v1**
   HTTP method: **GET**
   Relative path: **/{poll_id}/results**
   Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab4_retrieve_poll_results

   NOTE: Notice that the script is using the GlideRecordSecure API.



7. Enable ACL authorization on the resource by setting an ACL. ACL settings are available under Security tab

   Requires ACL authorization: **checked**
   ACLs: Click to unlock, and browse to select the **Poll Manager** ACL



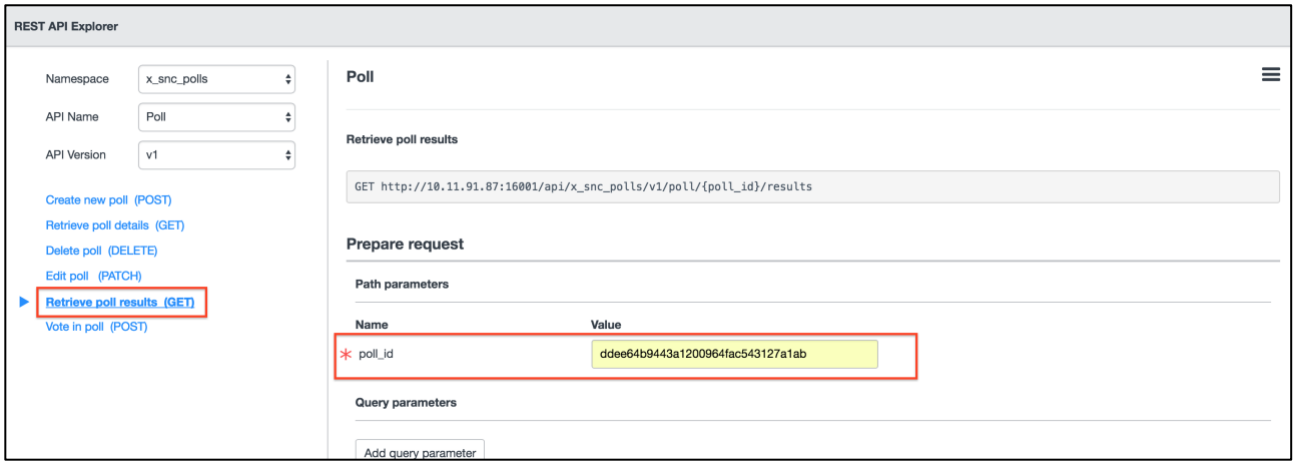
   NOTE: Only ACLs of type REST Endpoint can be used.

Click **Submit**.

BEST PRACTICES
Do: *Use the GlideRecordSecure API in your Scripted REST API Resource scripts to ensure that you are enforcing existing access controls on the requesting user when interacting with ServiceNow records.*
Do: *Test your access controls, both Authentication and Authorization, before making your API available to consumers.*

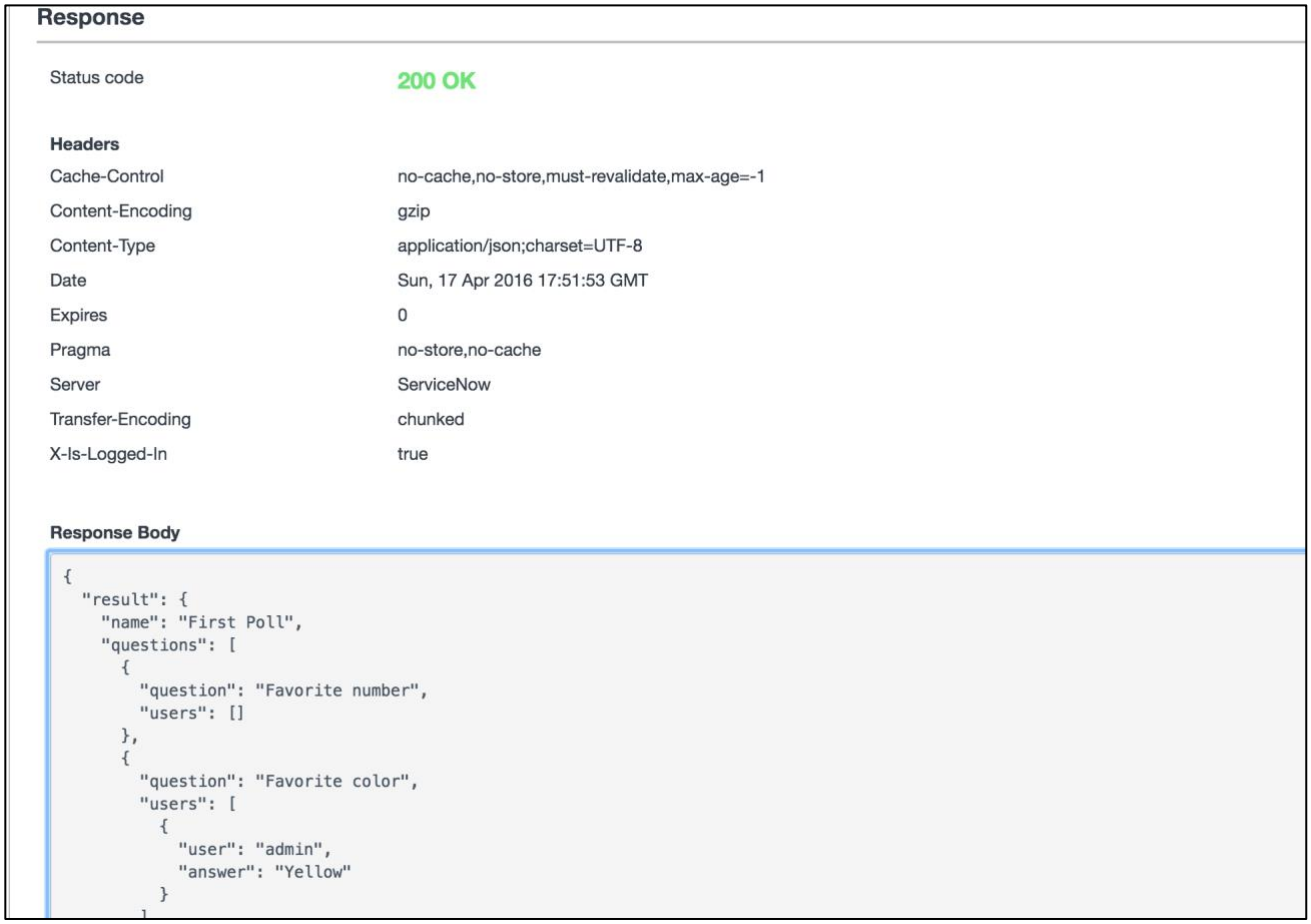
Test with REST API Explorer

8. Open 'Retrieve poll results' resource and Click **Explore REST API** in related actions.
9. **Retrieve poll results** resource is preselected in API Explorer. Fill in sys_id of poll.



   Click **Send**.

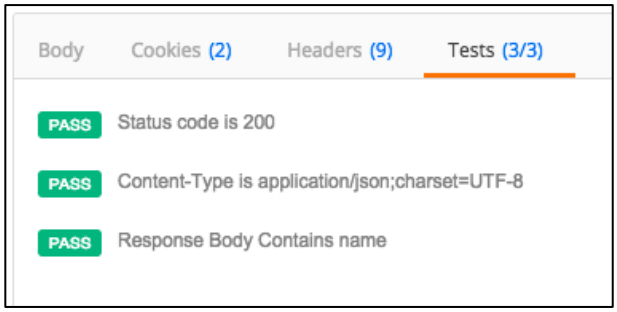
10. Verify response status code is **200-Ok.**



Create tests in Postman

11. In Postman select the 'CC17: Retrieve poll results' request. This is a pre-built request that will make a request to the 'Retrieve poll results' resource.
12. Update the request replacing the {{instance_url}}, authorization credentials, and {{poll_id}} with values appropriate for your lab instance. Use your admin credentials for this request. Once you've updated those values save and then send the request. If the request is successful (200 OK) you will see a response similar to the one you saw when testing in the REST API Explorer.
13. Now that you've made a successful request add tests to your Postman requests to validate the request matches the expected results. Add tests that verify the following details in the response:
    - Response status code is 200
    - Response headers include Content-Type of application/json;charset=UTF-8
    - Response body contains the text: "name"
14. You are on your own to create these tests in Postman. You can refer back to the tests you've created in the previous exercises for help.

15. Once you've added the tests save the request and send it. If you were successful you should see all the tests passing.

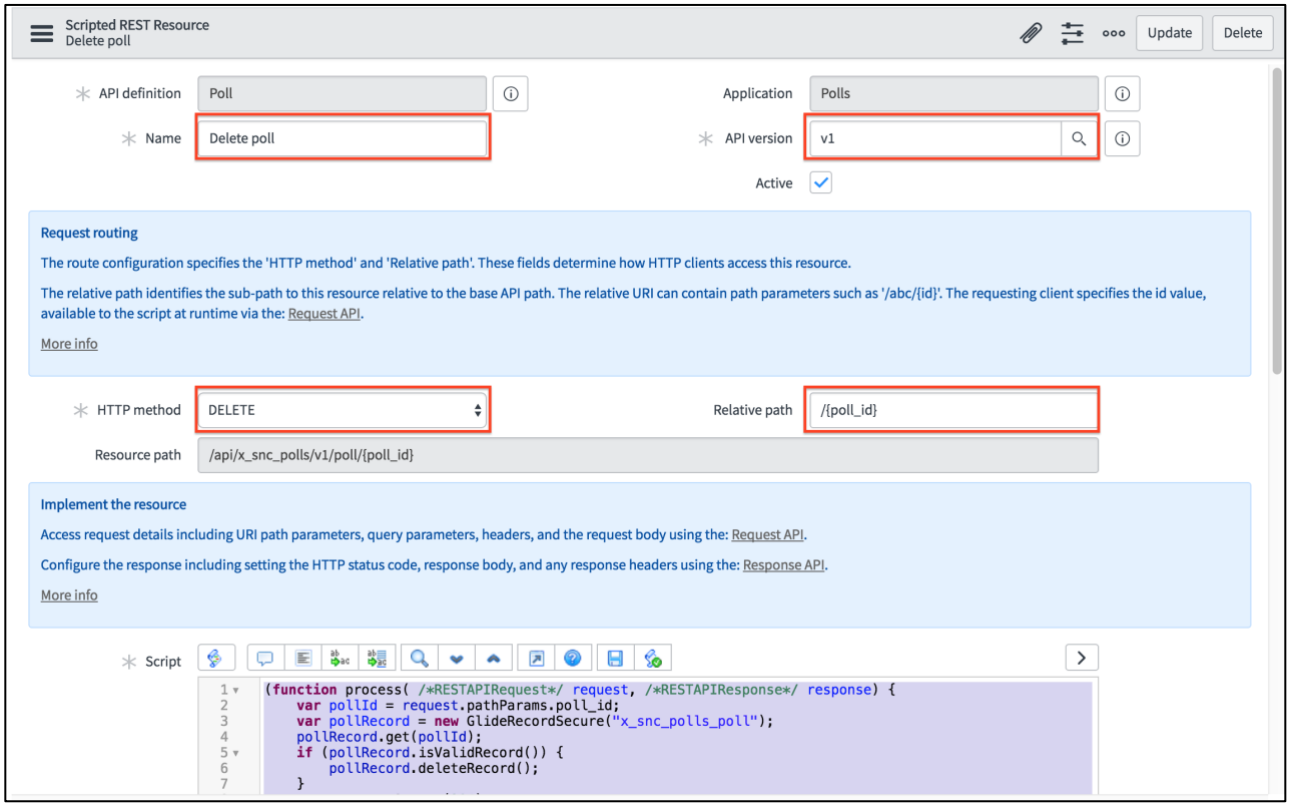


Note: If you are really stuck here you can refer to the pre-built request in the Postman collection named "CC17: Retrieve poll results w/ TEST" to see this request with tests fully specified.

Create New Resource in Polls API – Delete poll

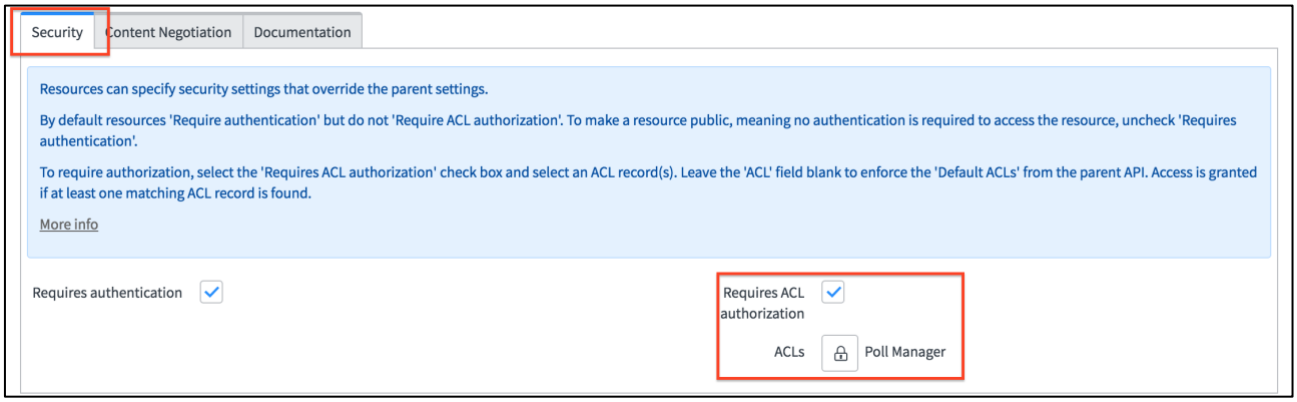
16. Open Polls API from studio. Add a **Resource** to the API. Click **New** on the **Resources** related list.

17. Give the resource a **Name**. Complete the script.
    Name: **Delete poll**
    API Version: **v1**
    HTTP method: **DELETE**
    Relative path: **/{poll_id}**
    Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab4_delete_poll



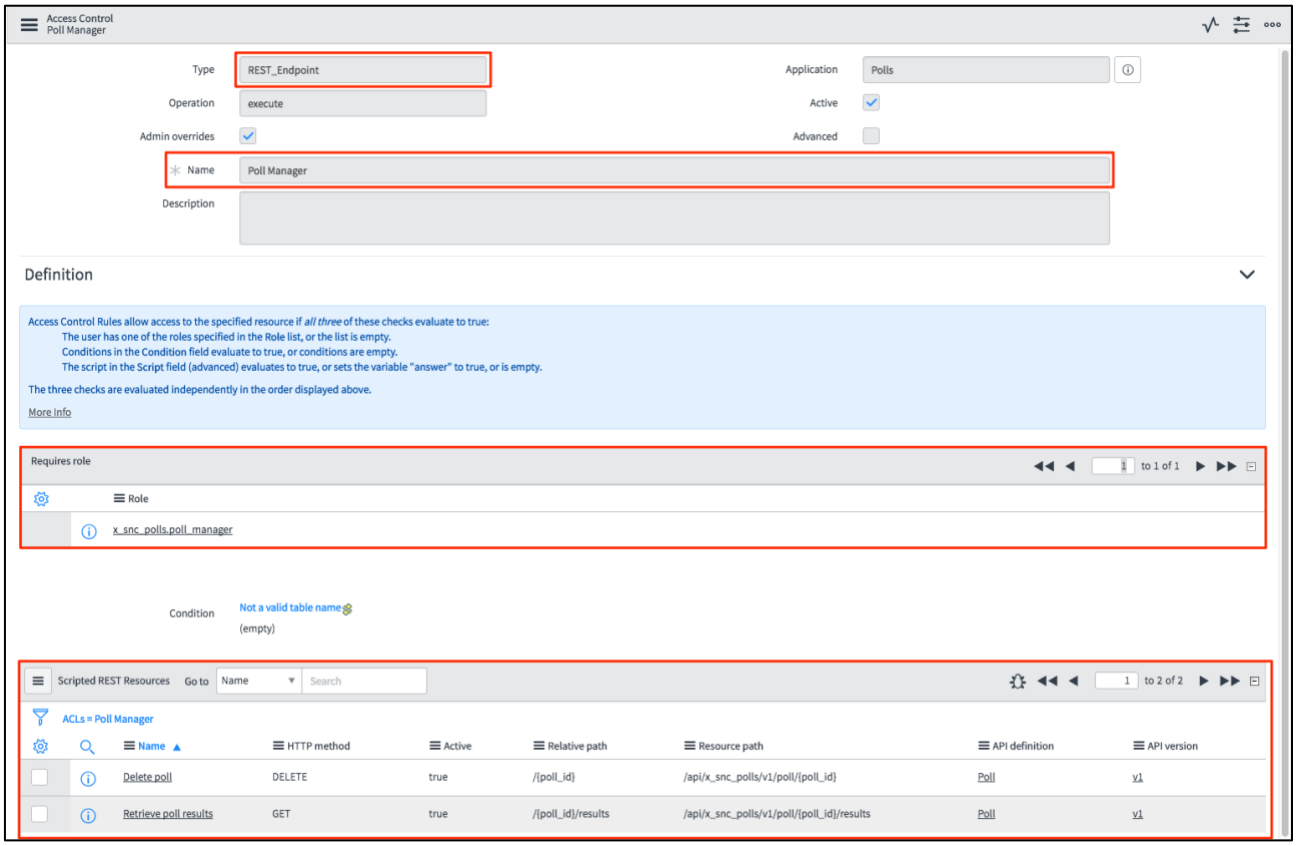
18. Enable ACL authorization on the resource by setting an ACL. ACL settings available under Security tab.

    Requires ACL authorization: **checked**
    ACLs: Click to unlock, and browse to select the **Poll Manager** ACL



    NOTE: Only ACLs of type REST Endpoint can be used

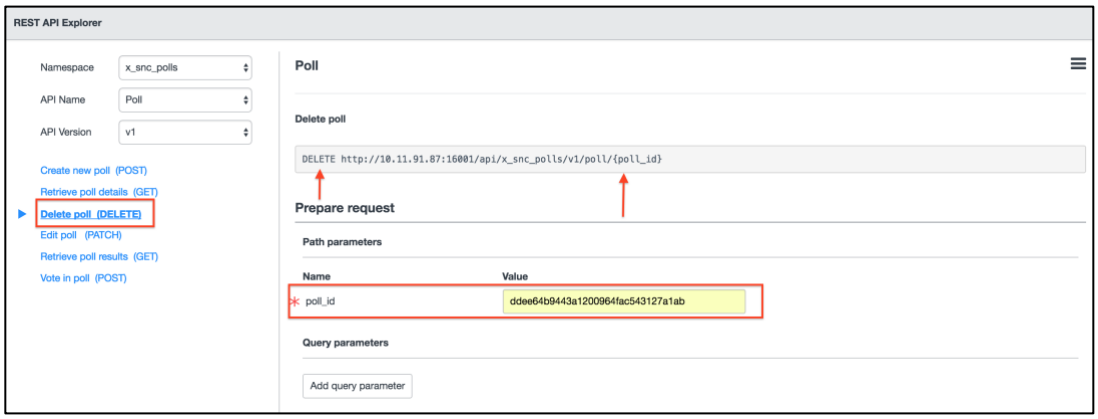
    Click **Submit**.



    NOTE: REST_Endpoint type ACLs (as shown above) are used to restrict access to Scripted REST API Resources. The 'Poll Manager' ACL has been specified on the 'Delete poll' resource and restricts access to this resource to users who have the role 'x_snc_polls.poll_manager'. **Only** users with this role can make requests to the 'Delete poll' resource.

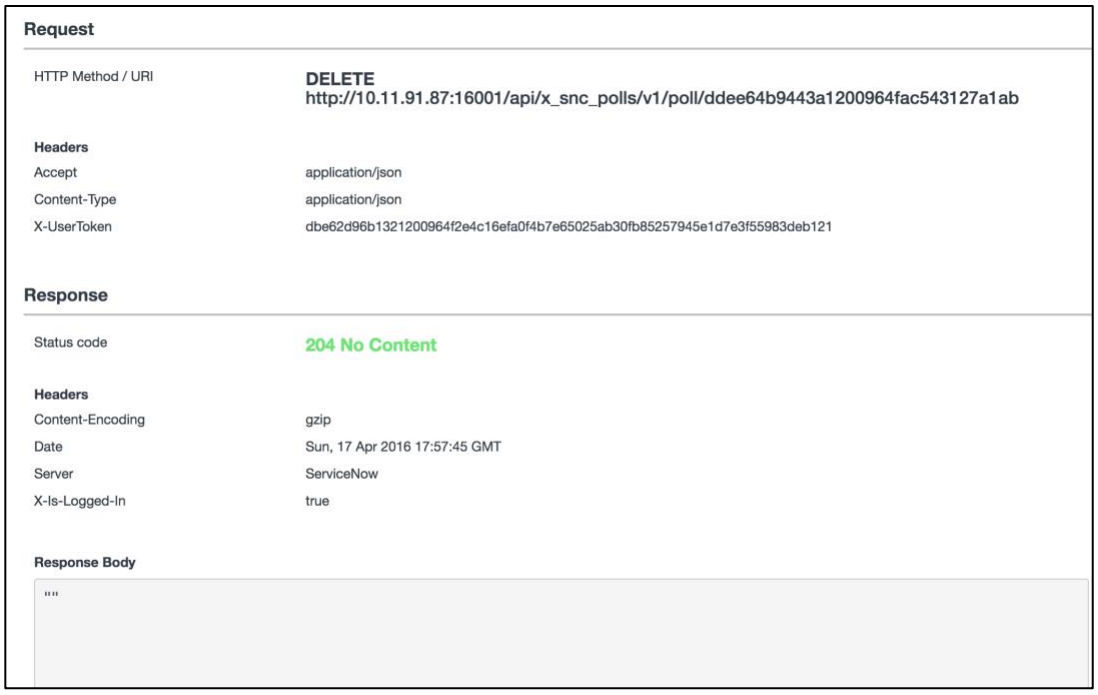
Test with REST API Explorer

19. Open Delete poll resource and click **Explore REST API** in related actions.
20. **Delete poll** resource is preselected in API Explorer. Fill in request body in raw tab under Request body section.



Click **Send**.

21. Verify response status code is **204-No content.**



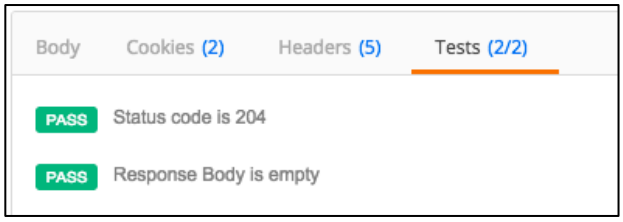
Create tests in Postman

22. In Postman select the ‘CC17:Delete poll’ request’. This is a pre-built request that will make a request to the ‘Delete poll’ resource.

23. Update the request replacing the {{instance_url}}, authorization credentials, and {{poll_id}} with values appropriate for your lab instance. Use your admin credentials for this request. And make sure that the admin user has the 'x_snc_polls.poll_manager' role. Once you've updated those values save and then send the request. If the request is successful (204 No Content) you will see a response similar to the one you saw when testing in the REST API Explorer.

24. Add tests that verify the following details in the response:
    - Response status code is 204
    - Response body is empty

25. After adding these tests issue the request and verify that your tests are passing as shown below.



26. Create a new poll in your instance and then update this request in Postman to use the new poll id and update the user credentials to use a user that **does not have** the 'x_snc_polls.poll_manager' role. Update the request in Postman and send the request.

27. **NOTE:** you have been deleting polls so you may need to go back and create some additional poll records in your instance so that there are polls that you can delete (hint use insert and stay to quickly create new polls for testing).

28. Send your updated request now and verify that for a user that when making a request with a user that **does not have** the 'x_snc_polls.poll_manager' role you receive a status code of **403 Forbidden** and that your test fail case 'Status code is 204' **fails** as shown below.

29. **NOTE:** If you are really stuck here you can refer to the pre-built request in the Postman collection named "CC17: Delete a poll w/ TEST" to see this request with tests fully specified.

Get Caught Up

If you were unable to successfully complete the lab this far, you can "fast forward" using the following steps. Otherwise proceed to the next section **Test with Postman**.

30. Similar to creating the Lab4 starting branch, the completed lab can also be checked out from a tag (**Lab4-complete**) in Source control.
31. In **Studio**, navigate to **Source Control > Create Branch**.
32. In the pop-up window, enter a branch name, then select **Lab4-complete** from the **Create from Tag menu**, and click **Create Branch**.

    Branch: **my-Lab4-branch-complete**
    Create from Tag: **Lab4-complete**
33. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
34. Verify Studio is on branch **my-Lab4-branch-complete**.
35. Close the REST API Explorer and API Analytics dashboard windows.

Lab 4 is complete. You are now ready to begin lab 5.

Lab Goal

Lab 5
Versioning

Versioning a REST API is a common task when you want to introduce new functionality or behaviors to your REST API but don't want to break existing clients. Scripted REST APIs support easily versioning your resources. With versioning support you can quickly create new versions of existing resources to introduce new functionality. You have the ability to specify what version of a resource is the default and to which requests will be routed if the client does not specify a version in the URL or to force the clients of your REST API to include a version in the URL they make requests to.
In this lab you will add a new version to the Polls REST API you have been creating to familiarize yourself with the versioning functionality in Scripted REST APIs.

BEST PRACTICES
Do: *Version your REST API. By default, you do not need to create a version when creating a Scripted REST API in ServiceNow. Best practice is to version your REST API and disable the default route so that consumers must explicitly include the version number in their request URL. In this way you allow clients to decide if and when they want to use a later version of your REST API. If you need to force them to move to a new version at a later point in time you have the ability to disable versions.*

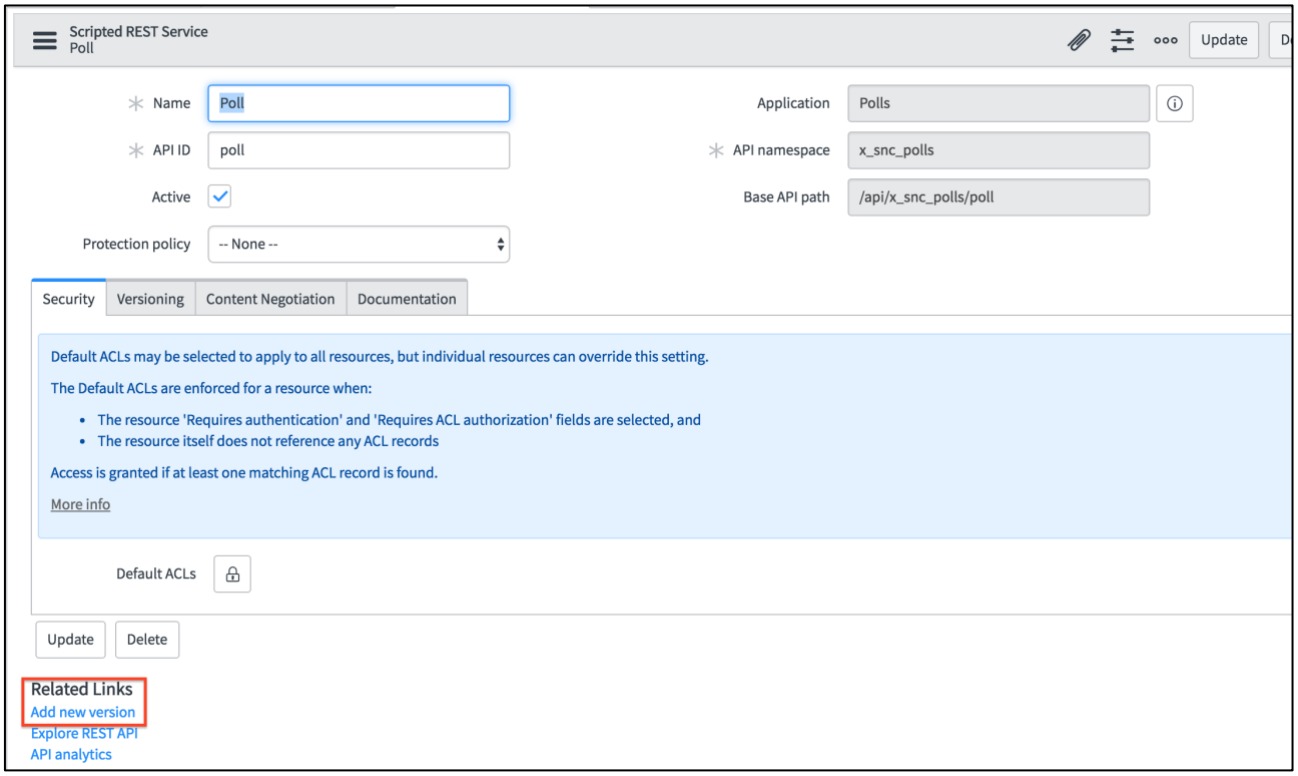
Create Lab 5 starting branch

1. In Studio, navigate to Source Control > Create Branch.
2. In the pop-up window, enter a branch name, then select **Lab5-start** from the **Create from Tag menu**, and click **Create Branch**.

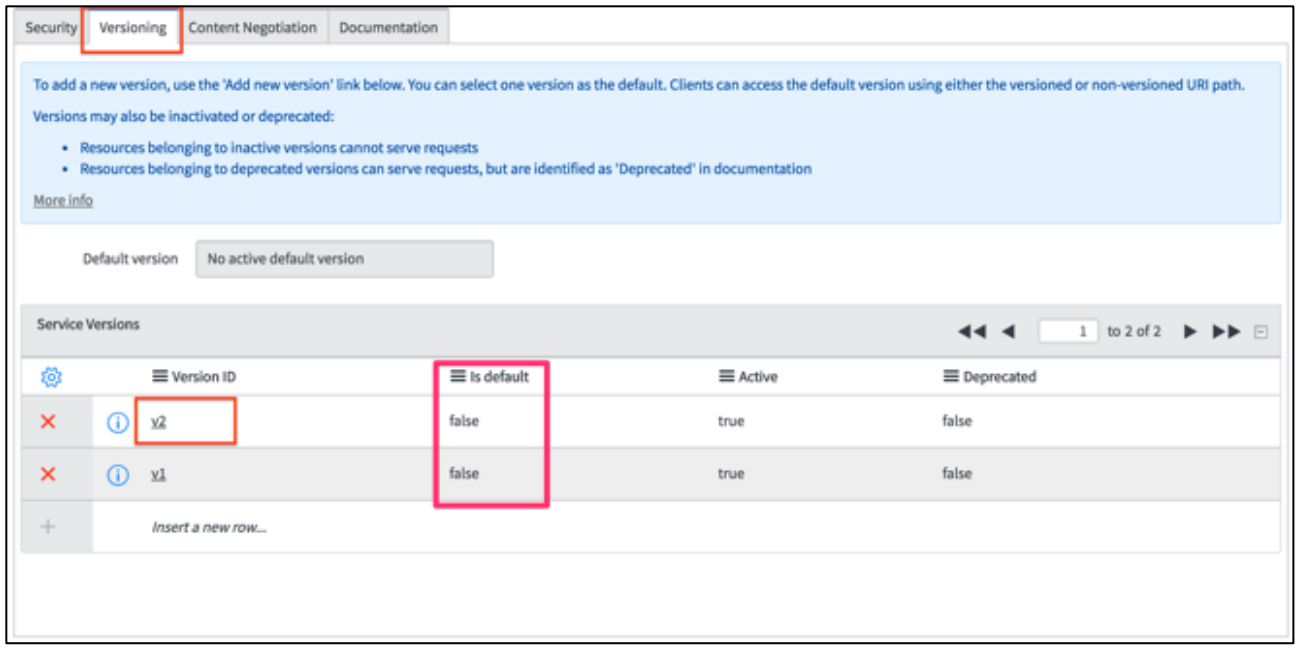
   Branch: **my-Lab5-branch**
   Create from Tag: **Lab5-start**
3. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
4. Verify Studio is on branch **my-Lab5-branch**.
5. You are now ready to start **Lab 5**.

Add Version to Poll REST API

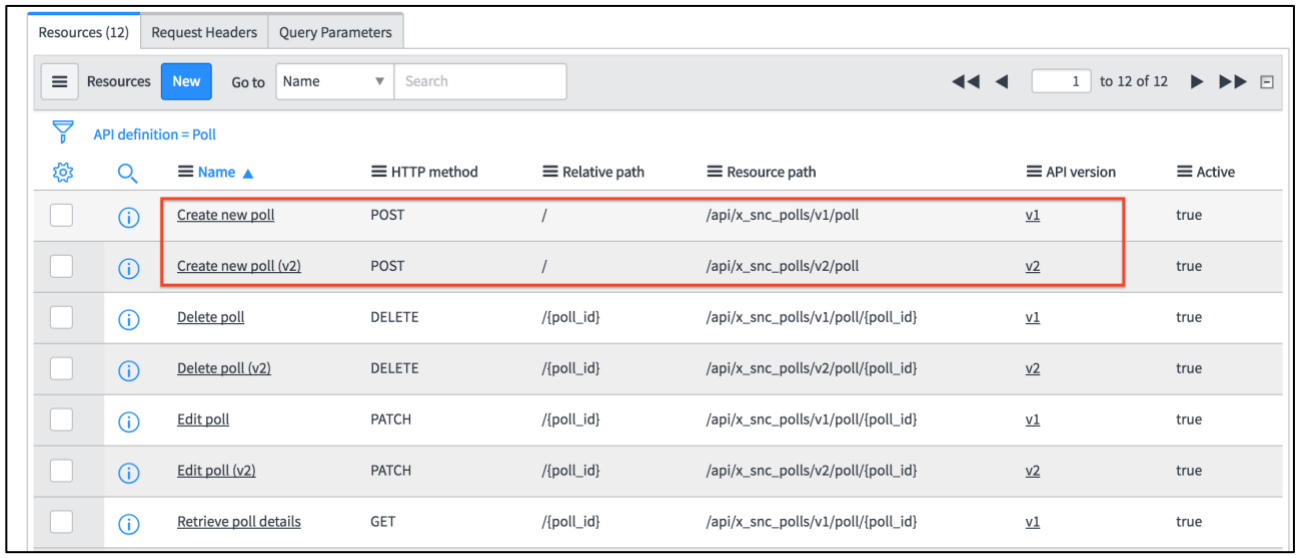
6. Open Poll API and click Add New version.



7. Select version 1 to copy resources. Click Ok.



NOTE: Both versions of the API have default set to **false.** This means that clients consuming this API must include the resource version in the URL.



NOTE: Observe every resource in v1 is copied and added to v2

Test with Postman

8. In Postman, review the requests named ‘CC17: Create new poll v1’ and ‘Create new poll v2’ noting the version number is explicitly specified in the URL for these two requests.
9. Add the tests that you added to the ‘CC17: Create new poll’ request to this the V1 and V2 requests. The behavior between the V1 and V2 resources has not been updated so you can copy and paste your test cases from the ‘CC17: Create new poll’ request and the tests should pass.
10. Verify that your test cases pass successfully.

Get Caught Up

If you were unable to successfully complete the lab this far, you can “fast forward” using the following steps. Otherwise proceed to the next section **Test with Postman**.

11. Similar to creating the Lab2 starting branch, the completed lab can also be checked out from a tag (**Lab5-complete**) in Source control.
12. In **Studio**, navigate to **Source Control > Create Branch**.
13. In the pop-up window, enter a branch name, then select **Lab5-complete** from the **Create from Tag menu**, and click **Create Branch**.

    Branch: **my-Lab5-branch-complete**
    Create from Tag: **Lab5-complete**
14. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
15. Verify Studio is on branch **my-Lab5-branch-complete**.
16. Close the REST API Explorer and API Analytics dashboard windows.

Lab 5 is complete. You are now ready to begin lab 6.

Lab Goal

Lab 6
Error handling

Errors… they happen to the best of us. Whether you are making requests to a 3rd party REST API or your own REST API there are times when you receive errors. Ideally the error message provides you (the client) with enough information to realize what went wrong, if it was your fault (client) or their fault (REST API) and how you can proceed.
Scripted REST APIs provide a helper API (sn_ws_err) to make it easier for you as the REST API designer to easily to return consistent and informative error messages from your REST API.

Create Lab 6 starting branch

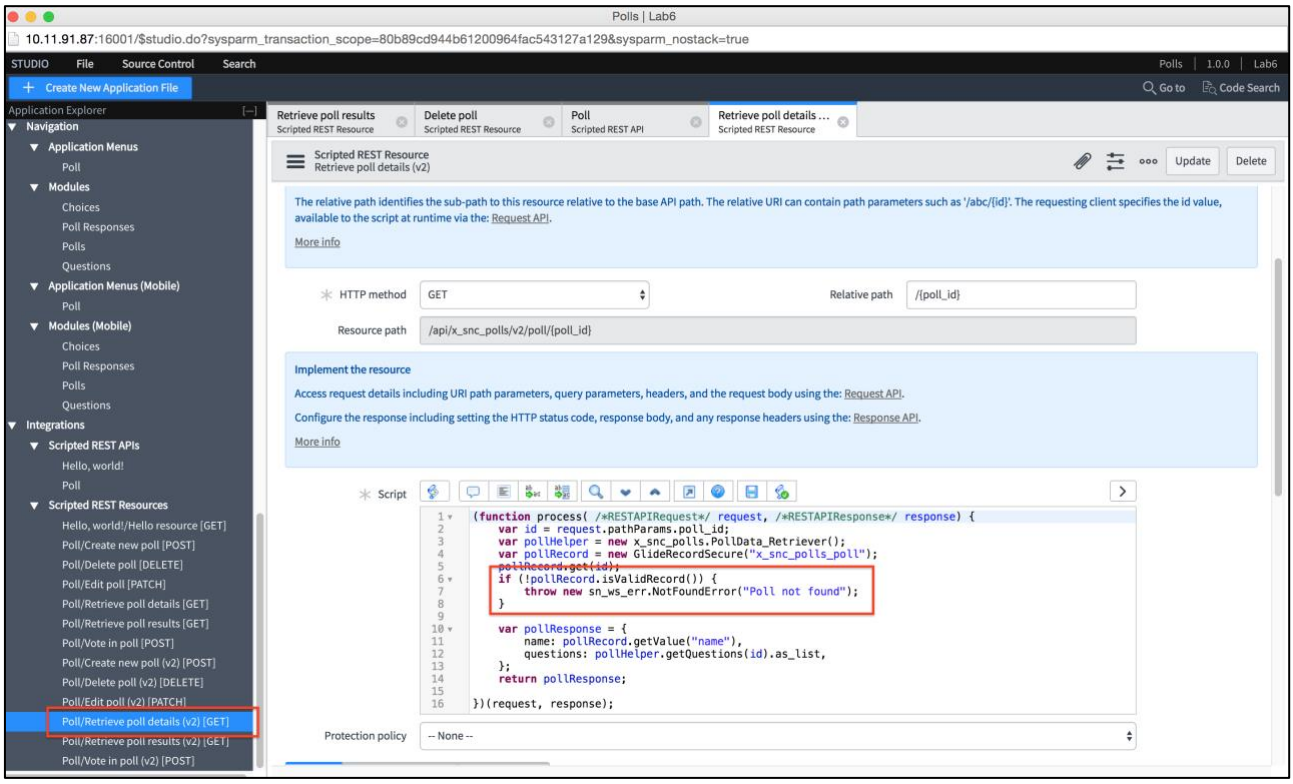
1. In **Studio**, navigate to **Source Control > Create Branch**.
2. In the pop-up window, enter a branch name, then select **Lab6-start** from the **Create from Tag menu**, and click **Create Branch**.

   Branch: **my-Lab6-branch**
   Create from Tag: **Lab6-start**
3. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
4. Verify Studio is on branch **my-Lab6-branch**.
5. You are now ready to start **Lab 6**.

Add Error handling to API – Retrieve poll detail

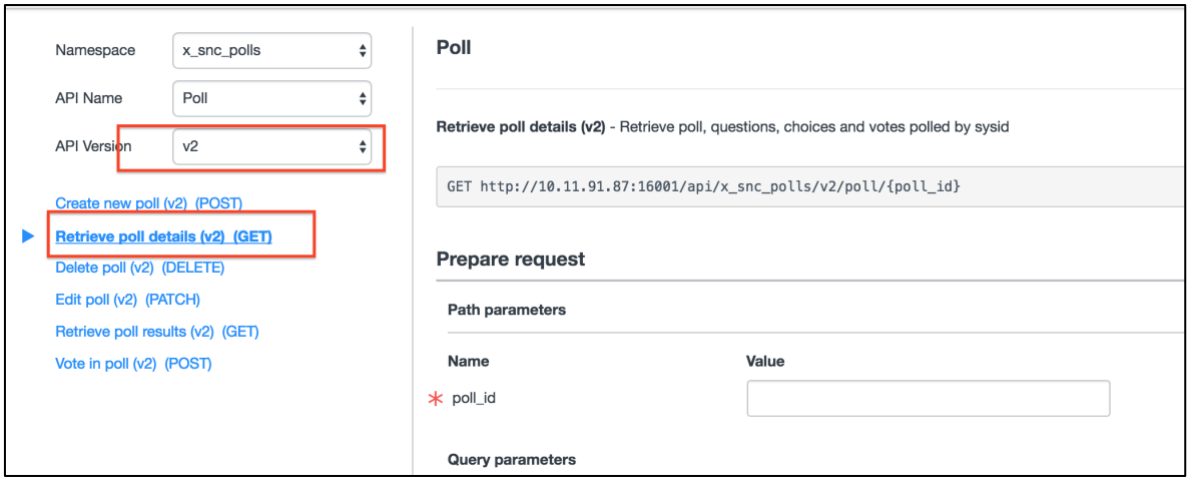
6. Open Retrieve poll detail (v2).
7. Modify script to check if poll record exists and send a 404 error response.

   Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab6_retrieve_poll_detail_v2

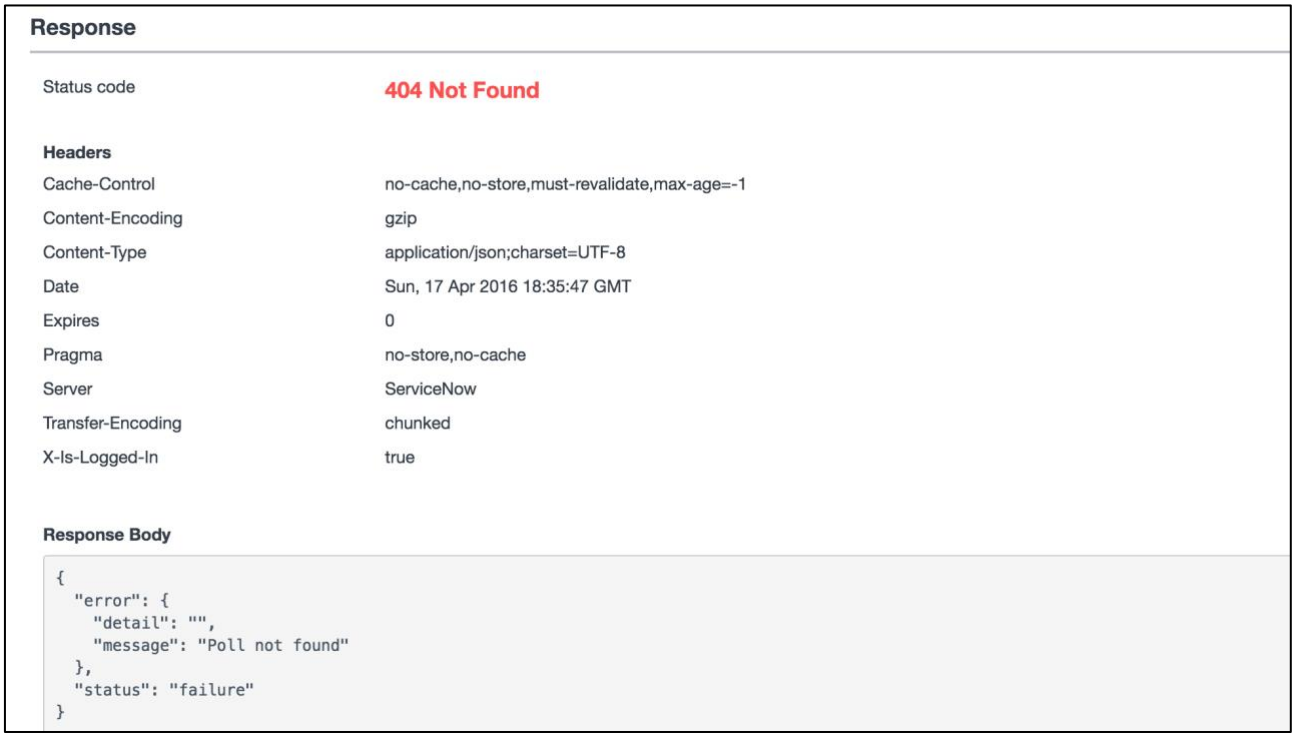


Test with REST API Explorer

8. Open 'Retrieve poll detail (v2)' resource and Click **Explore REST API** in related actions.
9. **Version v2** of **Retrieve poll results** resource is preselected in API Explorer.



10. Fill in **invalid** sys_id of poll and click Send.
11. Verify response status code is **404-Not Found.**



Create tests in Postman

12. In Postman select the 'CC17: Retrieve poll detail V2' request'. This is a pre-built request that will make a request to the 'Retrieve poll detail V2' resource.
13. Update the request replacing the {{instance_url}}, authorization credentials, and {{poll_id}} with values appropriate for your lab instance. Be sure to specify an **invalid** poll_id. Send a request and verify that you receive a **404 Not Found** status code and that the response body contains the same error message you received in REST API Explorer.
14. Add tests that verify the following details in the response:

    - Response status code is 404
    - Response body contains "error"
    - Response headers include Content-Type of application/json;charset=UTF-8
    - Response body is JSON and contains a status property with a value of 'failure'.

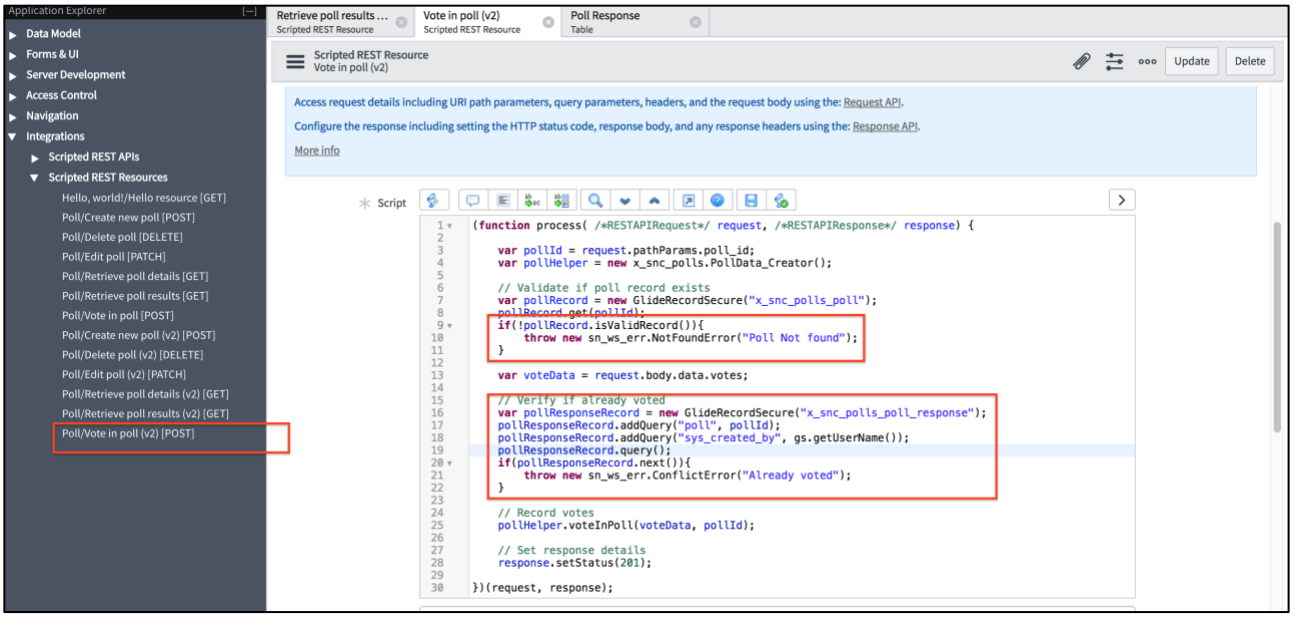
    NOTE: Postman has a test feature that allows you to parse a JSON response and verify it contains a specific property and value. See if you can figure out how to use it. More info can be found at https://www.getpostman.com/docs/testing_examples.

If you are stuck here you can refer to the pre-built request in the Postman collection named "CC17: Retrieve poll detail V2 w/ TEST" to see this request with tests fully specified.

Add Error handling to API – Vote in poll

15. Open Vote in poll(v2).
16. Modify script to check whether poll record exists as well as to see user already voted for the poll.

Script: Copy script from http://bit.ly/CC17_ScriptedRESTAPI_Lab6_vote_in_poll_v2

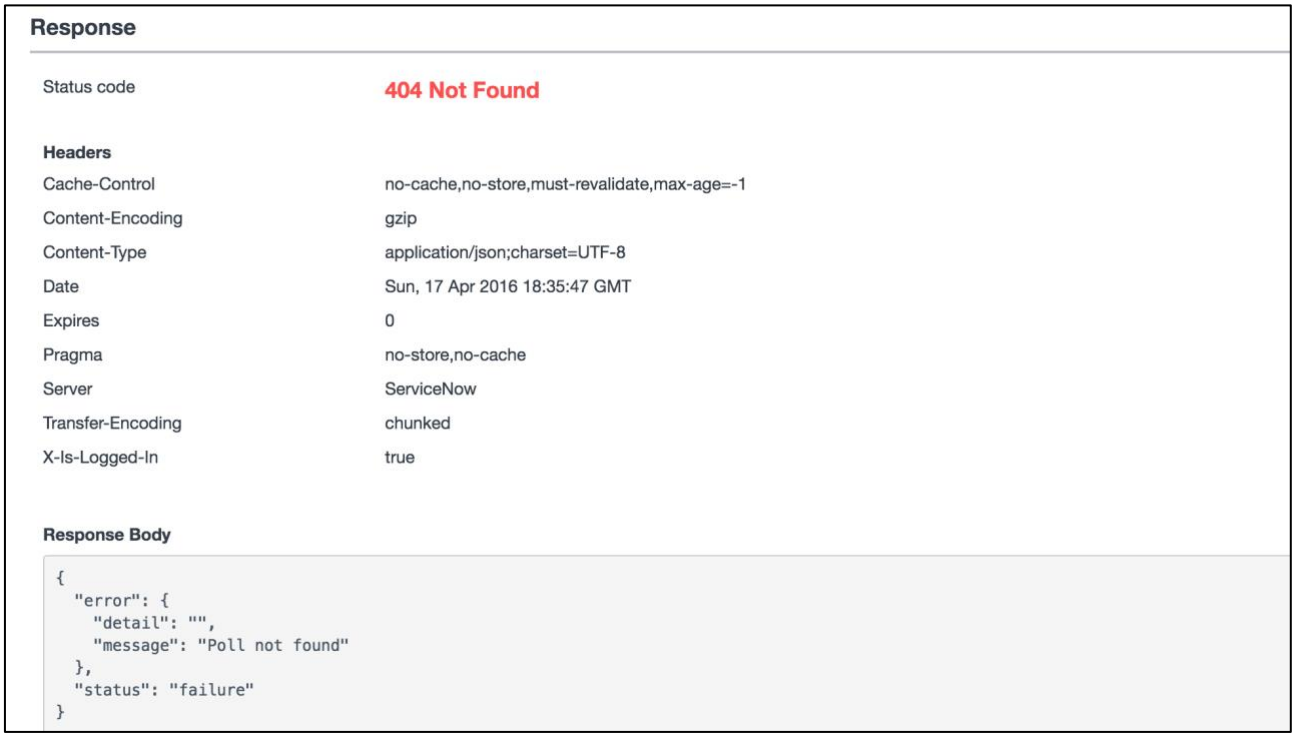


Test with REST API Explorer

Open 'Vote in poll (v2)' resource and Click **Explore REST API** in related actions.

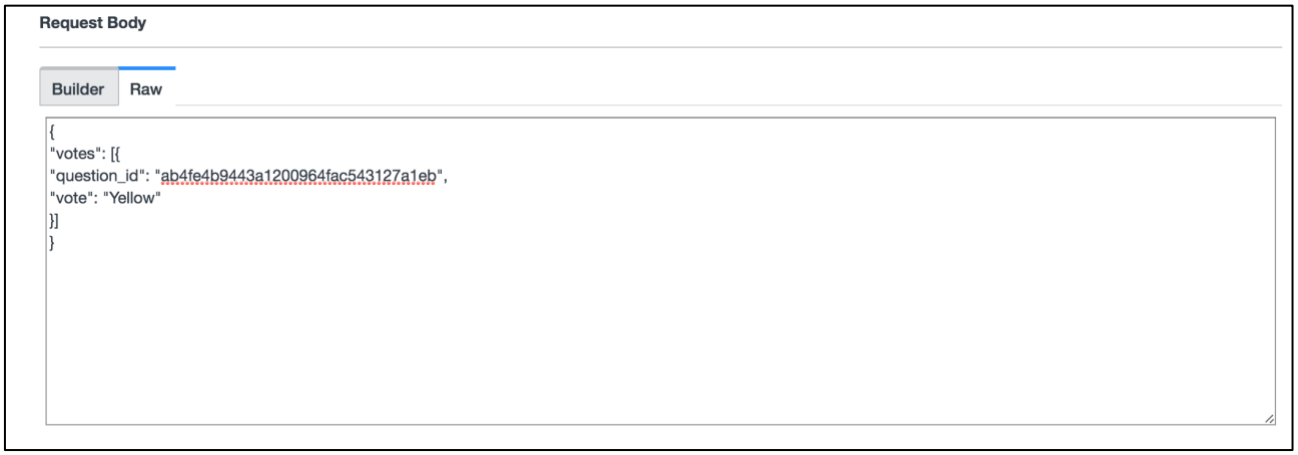
17. **Version v2** of **Vote in poll** resource is preselected in API Explorer.
18. Fill in an **invalid** sys_id of poll and click Send

19. Verify response status code is **404-Not Found**



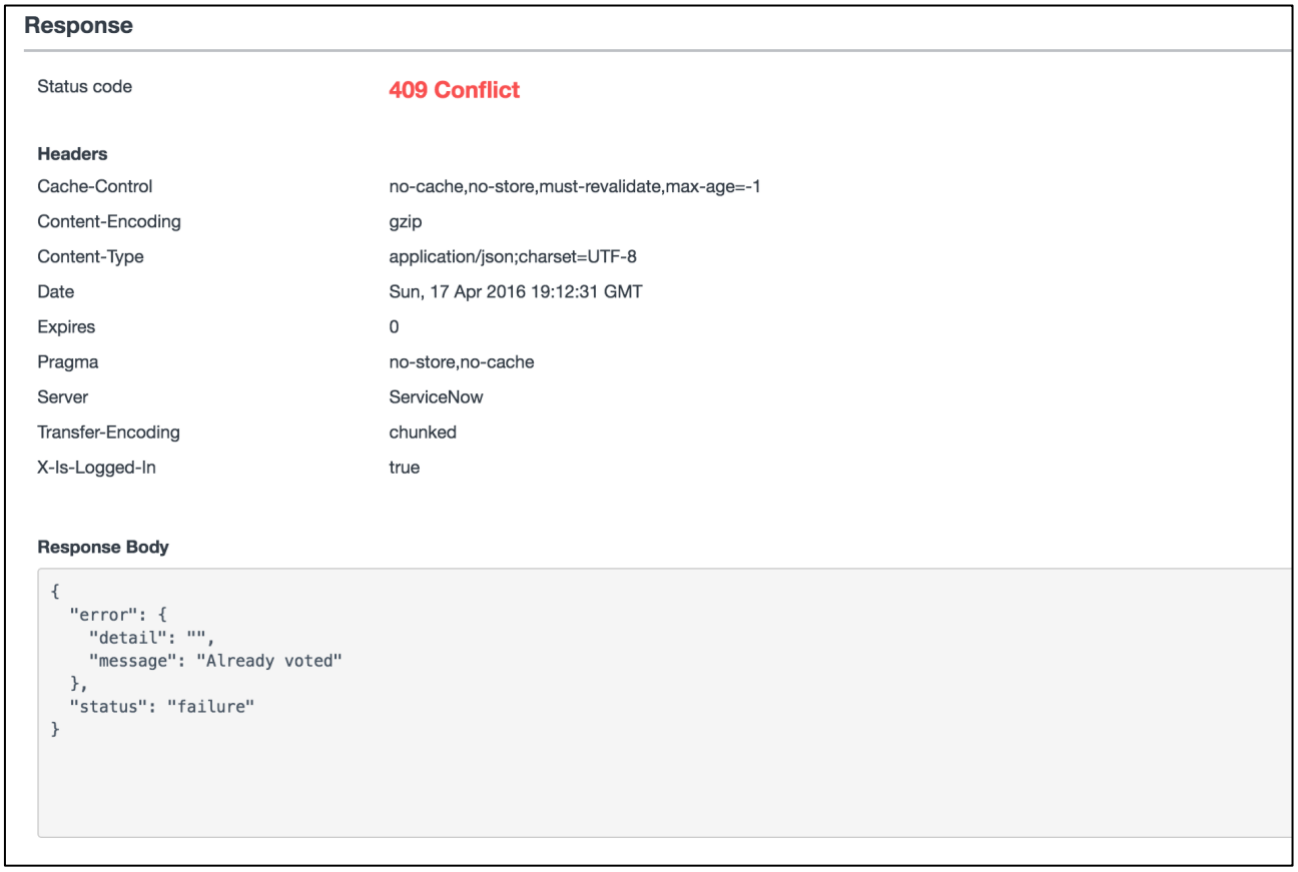
20. Now fill in a **valid** sys_id and request body and click Send to vote.

    Specify the request body as shown below. Sample script available for you to copy at: http://bit.ly/CC17_ScriptedRESTAPI_Lab6_vote_in_poll_v2_sample_request



21. Verify response status code is **409-Conflict.**

    NOTE: Depending on state of instance, you might need to fire the request twice. The first request is a valid vote while the second request results in a conflict response.



Create tests in Postman

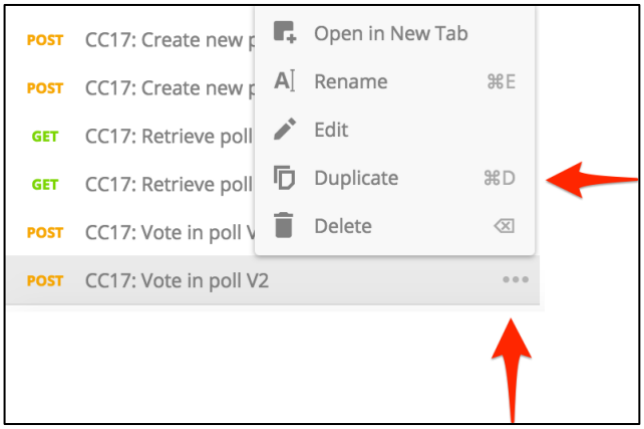
22. In Postman select the ‘CC17: Vote in poll V2’ request’. This is a pre-built request that will make a request to the ‘Vote in poll V2’ resource.

23. Update the request replacing the {{instance_url}}, authorization credentials, {{poll_id}}, and body of the request with vote details (question_id and vote value) appropriate for your lab instance.

    NOTE: Depending on the state of instance, you might need to fire the request twice. The first request is a valid vote while the second request results in a conflict response. This is because you have added a constraint that users can only vote for a question once.

24. Add tests that verify the following details in the response:
    - Response status code is 409 Conflict
    - Response body contains ‘error’
    - Response headers include Content-Type of ‘application/json;charset=UTF-8’
    - Response body is JSON and contains an error property with message property with a value of ‘Already voted’.

25. In Postman create a copy of this request by clicking on the ‘…’ icon to the right of the request name in the left-hand list of requests as shown below and then clicking ‘Duplicate’. This will create a copy of your request that you will update to verify the error message that is returned when you attempt to vote on the same question twice.



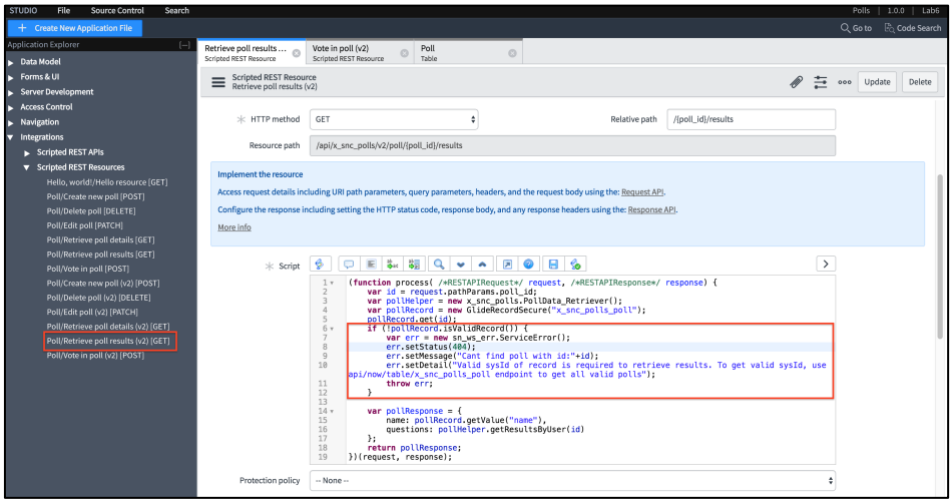
26. Update this new request adding tests that verify the following details in the response:
    - Response status code is 409
    - Response body contains “error”
    - Response headers include Content-Type of application/json;charset=UTF-8
    - Response body is JSON and contains a status property with a value of ‘failure’.

27. Issue the request and verify that all of your tests have completed successfully.

Add Error handling to API – Retrieve poll results

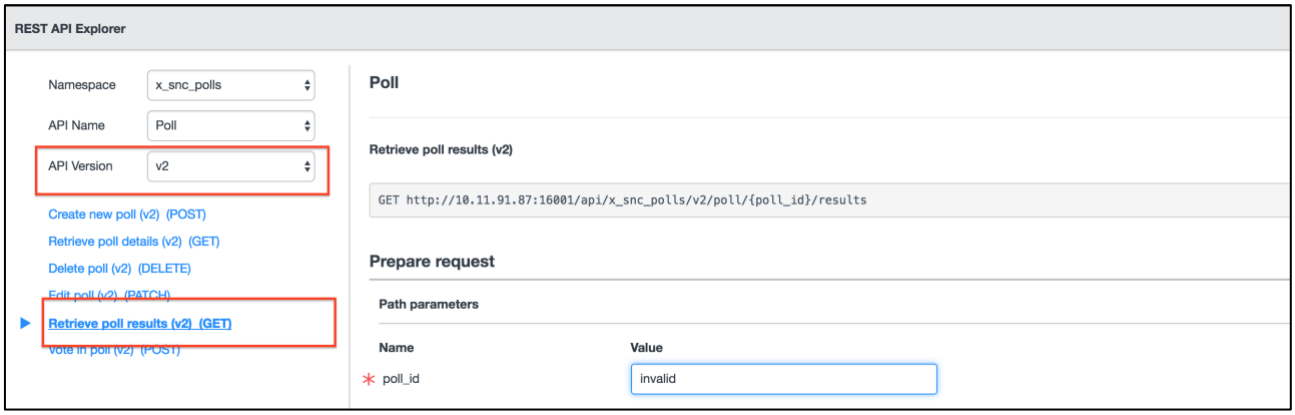
28. Open Retrieve poll results (v2).
29. Modify script to check if poll exists and send customized error response.

    Script: Copy script: http://bit.ly/CC17_ScriptedRESTAPI_Lab6_retrieve_poll_results_v2



Test with REST API Explorer

30. Open 'Retrieve poll results (v2)' resource and Click **Explore REST API** in related actions.
31. **Version v2** of **Retrieve poll results** resource is preselected in API Explorer.



32. Fill in invalid sys_id of poll and click Send.

33. Verify response status code is **404-Not Found** with custom Error message.

Create tests in Postman

34. Now that you have experience using Postman to build requests and create tests you are on your own for this last Postman test.
35. Create a new Postman request, either brand new or by duplicating an existing request, that makes a request to the **Retrieve poll results V2 resource** with an **invalid** poll id.
36. Add tests to this request verifying the following:
    - Response status code is 404
    - Response body contains “error”
    - Response headers include Content-Type of ‘application/json;charset=UTF-8’
    - Response body is JSON and contains an error property with with a message child property that has a value of ‘Can’t find poll with id:invalid’.
37. Issue request and verify that all tests pass successfully.

Get Caught Up

If you were unable to successfully complete the lab this far, you can “fast forward” using the following steps. Otherwise proceed to the next section **Test with Postman**.

38. Similar to creating the Lab6 starting branch, the completed lab can also be checked out from a tag (**Lab6-complete**) in Source control.
39. In **Studio**, navigate to **Source Control > Create Branch**.
40. In the pop-up window, enter a branch name, then select **Lab6-complete** from the **Create from Tag menu**, and click **Create Branch**.

    Branch: **my-Lab6-branch-complete**
    Create from Tag: **Lab6-complete**
41. When the switch is complete, click **Close Dialog** in the Create Branch pop-up.
42. Verify Studio is on branch **my-Lab6-branch-complete**.
43. Close the REST API Explorer and API Analytics dashboard windows.

Lab 6 is complete. You are now ready to begin lab 7.

Lab Goal

Lab 7 Challenge Lab

Congratulations you've made it to the challenge lab. If you've made it this far you have either whizzed right through labs 1-6 and are a Scripted REST API and Postman expert or you're working on this challenge lab after the CreatorCon Workshop.

Throughout this workshop you have built a Scripted REST API for the Polls application and created requests with tests in the Postman collection that allow you to quickly issues requests to your REST API that also have test cases that verify the REST API is returning valid responses and behaving as intended.

Creating tests that validate your REST APIs behavior is a **BEST PRACTICE** that allows you to easily verify your REST API is behaving as intended and quickly identify any breaking changes in your REST API in the future as you make changes to add functionality or patch bugs.

Postman makes it easy to run these tests manually but **Nobody** enjoys running tests manually all the time. Wouldn't it be nice if you could automate these tests? I certainly think so. Let's take this one step further and see if you can get these tests to run from the command line. Postman provides a tool called **Newman** that allows you to run the tests in your existing Postman collection from the command line and see the results either at the command line or write them to a file. For this challenge lab your task is to install Newman and run your Postman collection from the command line. Check out the links below to download and install Newman and get more information on how to run a collection from the command line.

If you can get Newman running from the command line you are only a few steps away (bash, cron, PowerShell script) from automating your REST API test cases with Postman and Newman. This is outside the scope of this lab but you can imagine how you could integrate this with a continuous integration testing tool to have these tests run on a regular basis or as part of a build process.

Newman: http://blog.getpostman.com/2014/05/12/meet-newman-a-command-line-companion-for-postman/

Running collections from the command line:
https://www.getpostman.com/docs/newman_intro

Integrating automated API tests with Jenkins:
https://www.getpostman.com/docs/integrating_with_jenkins