CREATORCON™

May 9–11 • Orange County Convention Center • Orlando, FL

Lab Guide

Working with Webhooks

Josh Nerius & Martin Wood

Lab instance: **http://clabs.link/ccw3954**

Default Login / Password:

admin / Knowledge17

itil / Knowledge17

employee / Knowledge17

© 2017 ServiceNow, Inc. All rights reserved.

This

Page

Intentionally

Left

Not

Blank

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 0
Import Repo from GitHub

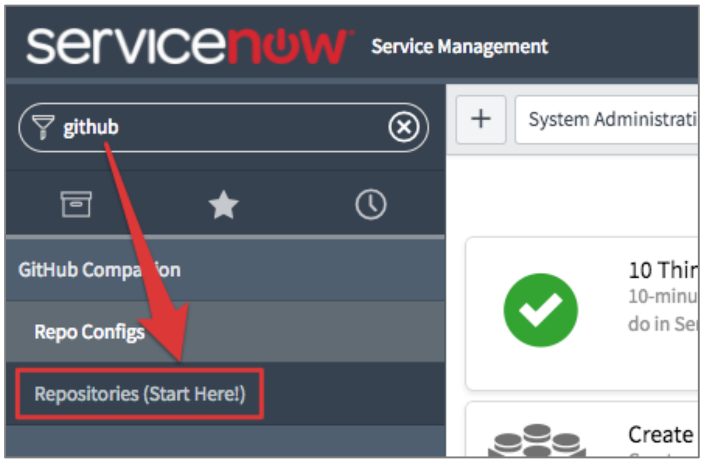
The creators of this lab have built a Source Control repository hosted on GitHub containing the finished versions of each lab. By utilizing ServiceNow's Source Control integration with Git, you can import this repository into your lab instance. Doing so will give you a "fast-forward" mechanism in case you get stuck at any point and wish to move on to the next lab.

This lab also utilizes a utility called "GitHub Companion" to simplify the process of forking, importing and performing other source control operations. Before you get started with the main lab, follow these steps to import the source control repository into your lab instance.

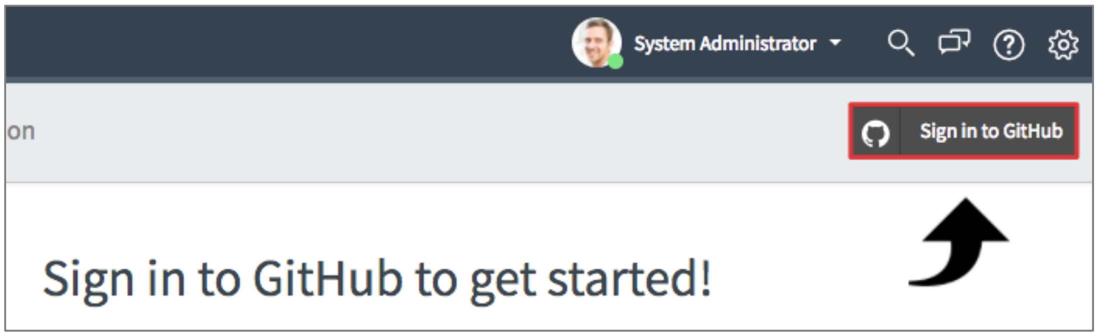
Sign In to GitHub Companion

Note: *If you do not have a GitHub account, please create one before completing these steps by visiting https://github.com*

1. Navigate to **GitHub Companion > Repositories (Start Here!)**.

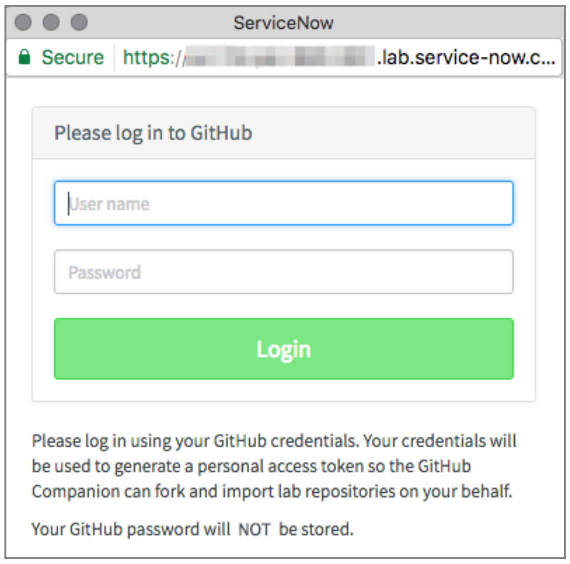


2. Click **Sign in to GitHub**.



Sign in to GitHub to get started!

3. Enter your GitHub **username** and **password** and click the **Login** button.

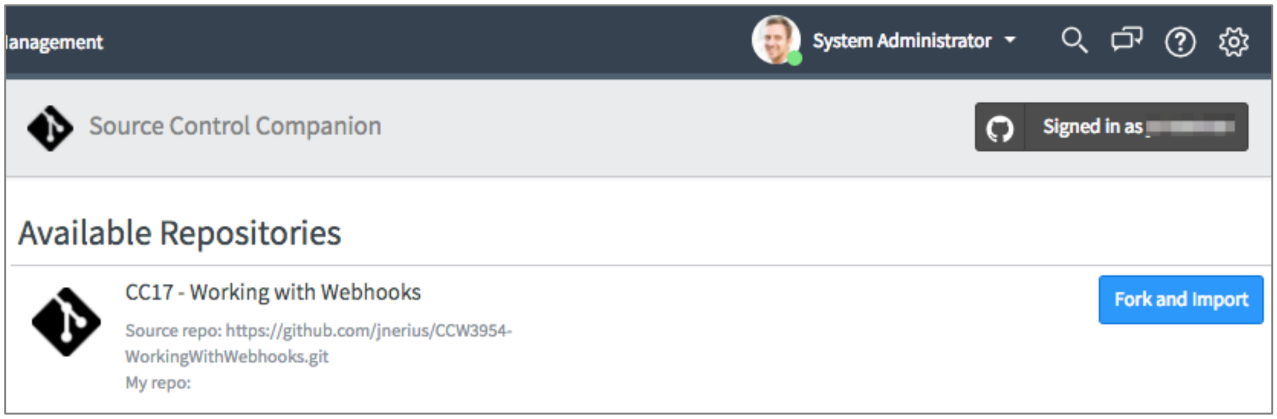


Please log in to GitHub

Please log in using your GitHub credentials. Your credentials will be used to generate a personal access token so the GitHub Companion can fork and import lab repositories on your behalf.

Your GitHub password will NOT be stored.

4. Upon logging in, you will be taken to a list of available repositories. You will see the repository that corresponds with this workshop there.



Available Repositories

CC17 - Working with Webhooks

Source repo: https://github.com/jnerius/CCW3954-WorkingWithWebhooks.git

My repo:

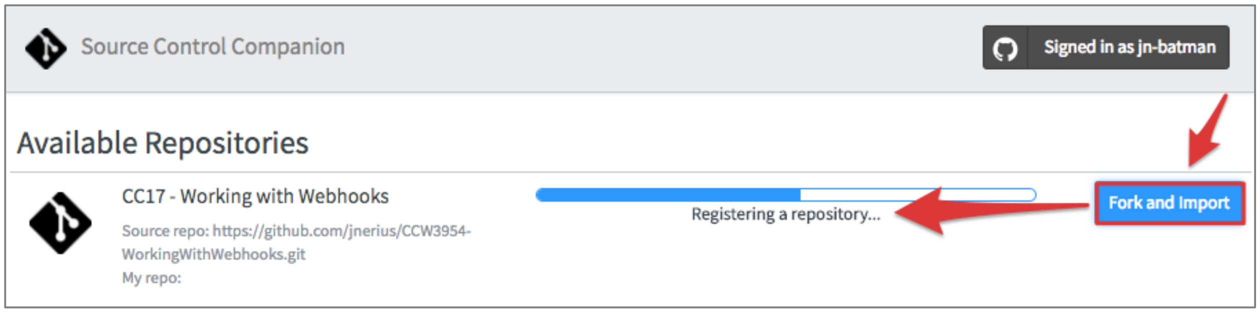
© 2017 ServiceNow, Inc. All rights reserved.

Fork and Import the Repository

Next, you will fork and import the repository. Forking the repository will give you your own copy of the application and associate it with your GitHub account. You can then make and commit changes to this repository as you work through the lab.

This would normally be a several step process, but with GitHub Companion, this is accomplished with a single click!

1. Click the **Fork and Import** button next to the repository in the list. A status bar will indicate the progress of the fork/import operations. This may take a minute or two depending on the size of the application.

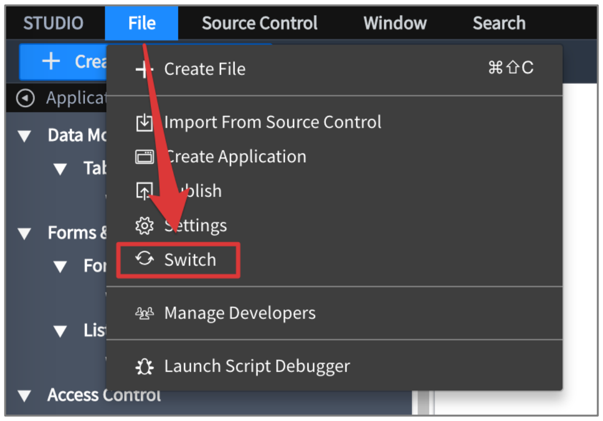

Source Control Companion
Signed in as jn-batman
Available Repositories
CC17 - Working with Webhooks
Source repo: https://github.com/jnerius/CCW3954-WorkingWithWebhooks.git
My repo:
Registering a repository...
Fork and Import

Select the Application in Studio

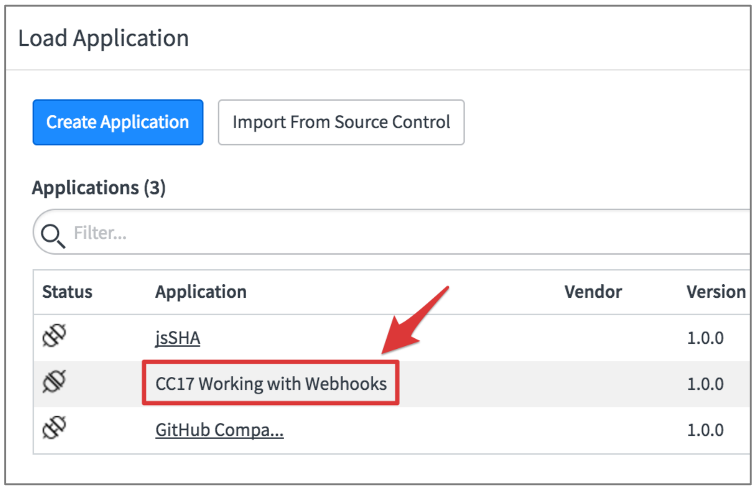
Now that you have forked and imported the application, you will work in this application throughout the lab. At this stage of the lab, the application will be empty, but the completed versions of the labs are captured in Git Tags behind the scenes.

If the **CC17 Working with Webhooks** application isn't already active in Studio, follow these steps to select it.

1. From Studio, click the **File** menu. Then, click the **Switch** menu item.



2. Click the name of the imported application from the list.



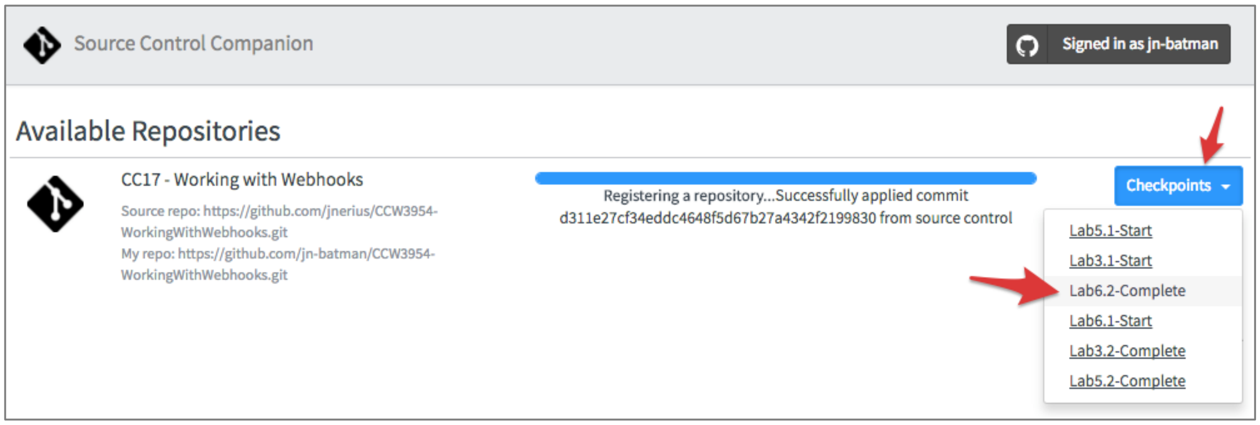
Status	Application	Vendor	Version
	jsSHA		1.0.0
	CC17 Working with Webhooks		1.0.0
	GitHub Compa...		1.0.0

Fast-forwarding or Switching Between "Checkpoints"

If at any time you wish to fast-forward to the next lab, you can use the Checkpoints button in the repository list. Behind the scenes, Checkpoints are **Git Tags**. Clicking a checkpoint will automatically create a **branch** from that tag and switch to that branch. If you're not familiar with Git and aren't sure what all of this means, don't worry - in short, switching to a Checkpoint just loads all of the necessary code needed to proceed with the lab.

Switching to a checkpoint:

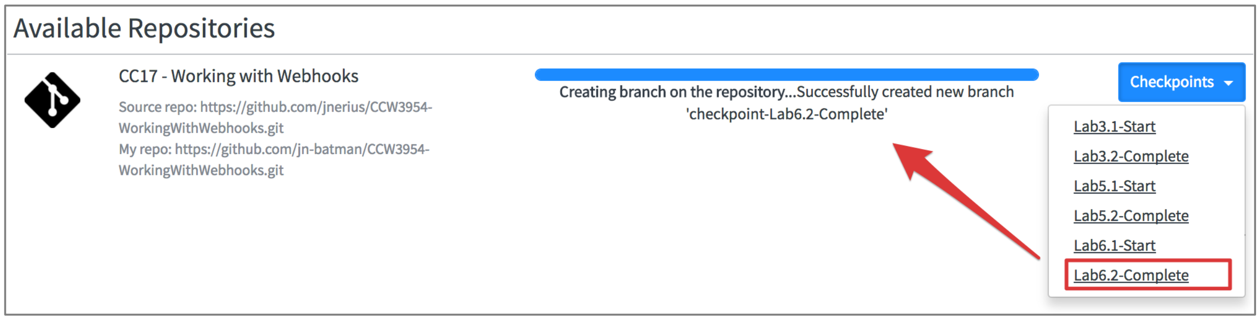
1. Click the **Checkpoints** dropdown next to the repository in GitHub Companion.



Source repo: https://github.com/jnerius/CCW3954-WorkingWithWebhooks.git
My repo: https://github.com/jn-batman/CCW3954-WorkingWithWebhooks.git

Registering a repository...Successfully applied commit d311e27cf34eddc4648f5d67b27a4342f2199830 from source control

2. Click the name of the checkpoint you wish to switch to. After confirming that you wish to switch to the selected checkpoint, you will see a progress indicator. When the branch/switch operation is complete, all code related to that checkpoint will now be available to you! Switch back to the Studio UI and resume the lab.



Source repo: https://github.com/jnerius/CCW3954-WorkingWithWebhooks.git
My repo: https://github.com/jn-batman/CCW3954-WorkingWithWebhooks.git

Creating branch on the repository...Successfully created new branch 'checkpoint-Lab6.2-Complete'

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

In this lab, you will create a GitHub repository and prepare it to send Webhooks to ServiceNow.

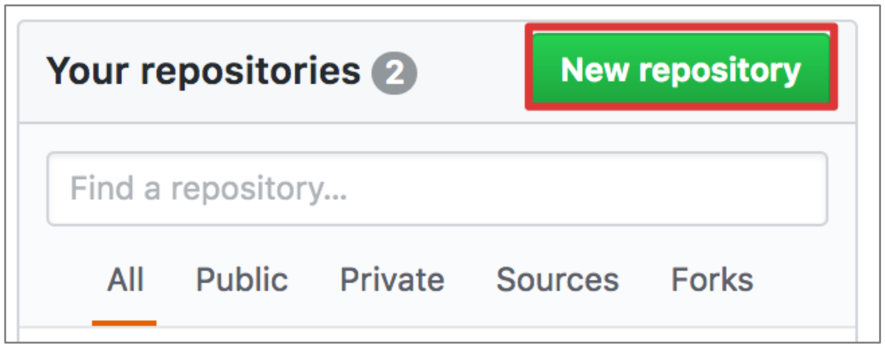
Lab 1
Get Started with GitHub Webhooks

Log in to GitHub

If you have not already done so, log in to your GitHub account. If you do not have a GitHub account, please register for one by visiting https://github.com.

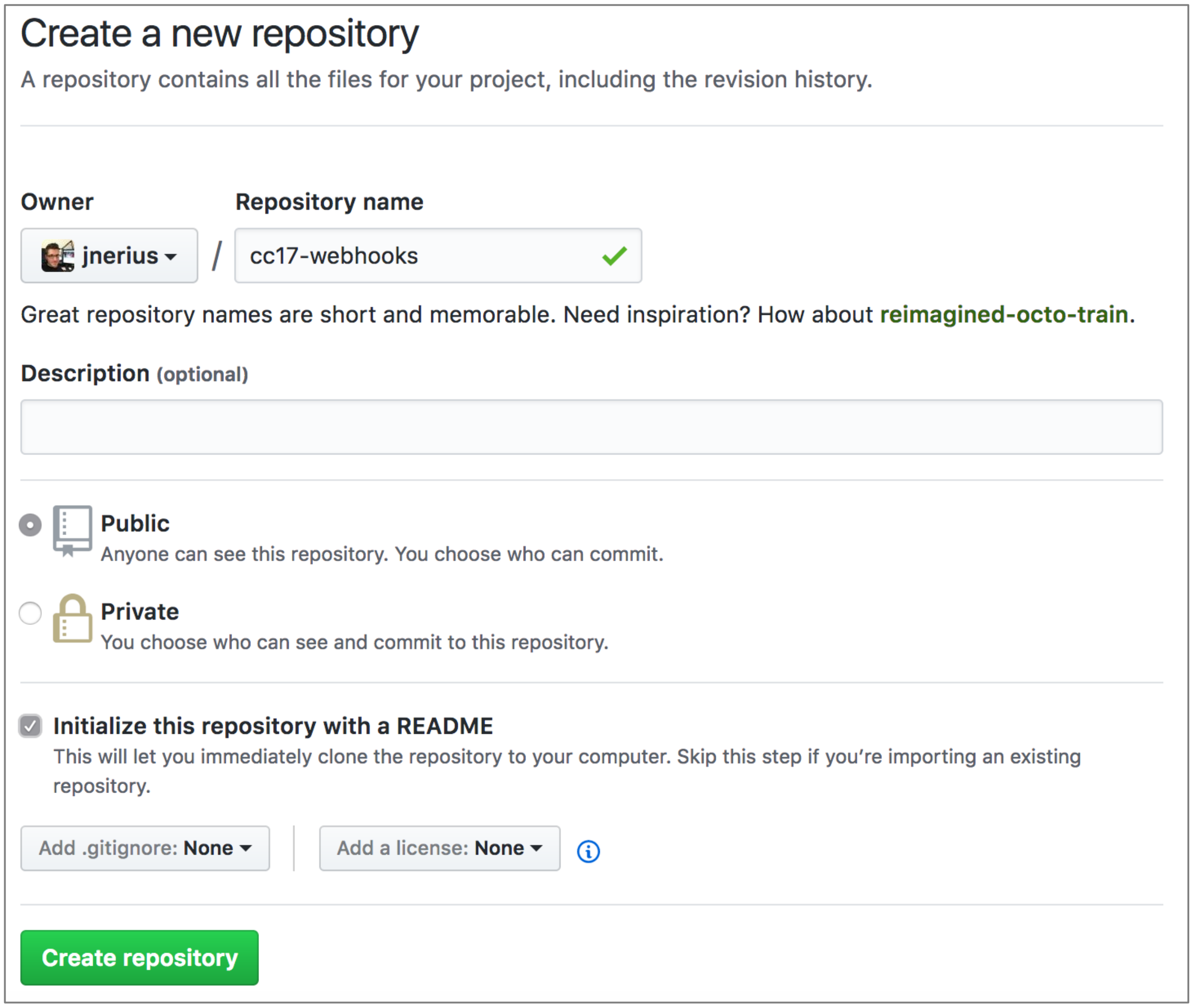
Create a New Repository

1. From your repository list, click the **New repository** button.



2. Configure the repository with the following settings:

- Repository name: **cc17-webhooks**
- Public: **Selected**
- Initialize this repository with a README: **Checked**



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner	Repository name
jnerius	cc17-webhooks

Great repository names are short and memorable. Need inspiration? How about **reimagined-octo-train**.

Description (optional)

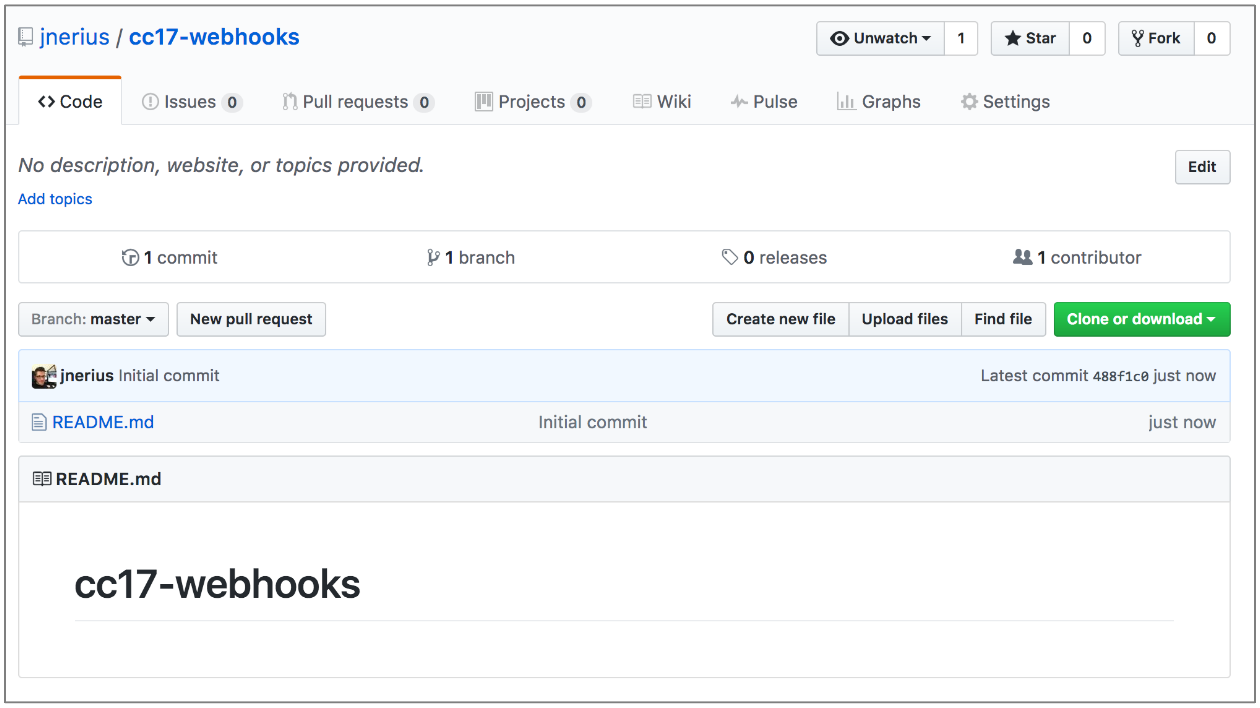
● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

✓ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

© 2017 ServiceNow, Inc. All rights reserved.

3. Click the **Create repository** button. You will be taken to your newly created repository.


jnerius / cc17-webhooks

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

No description, website, or topics provided. Edit

Add topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jnerius Initial commit — Latest commit 488f1c0 just now

README.md — Initial commit — just now

README.md

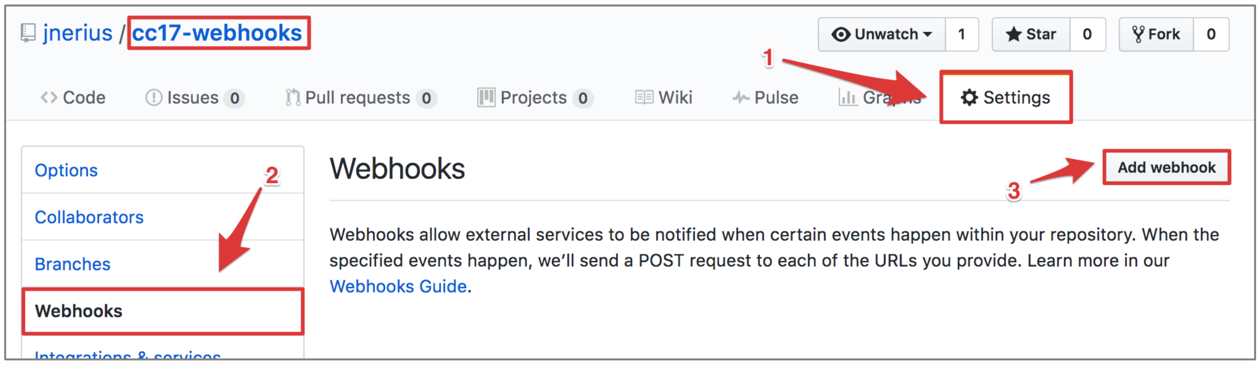
cc17-webhooks

Explore Webhook Options

1. With the cc17-webhooks repository open, click the **Settings** tab.

2. Click the **Webhooks** item in navigation menu on the left.

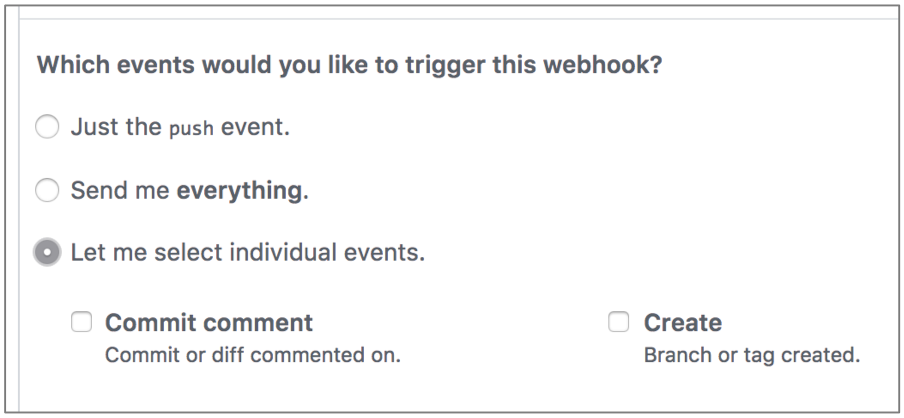
3. Click the **Add webhook** button in the resulting pane on the right.



Webhooks

Webhooks allow external services to be notified when certain events happen within your repository. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

4. Under **Which events would you like to trigger this webhook**, click the **Let me select individual events** radio button and examine the various options available to you.



Which events would you like to trigger this webhook?

( ) Just the push event.
( ) Send me everything.
(●) Let me select individual events.

☐ **Commit comment**
Commit or diff commented on.

☐ **Create**
Branch or tag created.

5. Change the radio button to **Send me everything**.

   Leave the form as-is for now. We'll come back to it in the next lab after we've established an endpoint.

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

In this lab, you will create a dummy endpoint using RequestBin (https://requestb.in) and configure GitHub to send Webhooks to this endpoint.

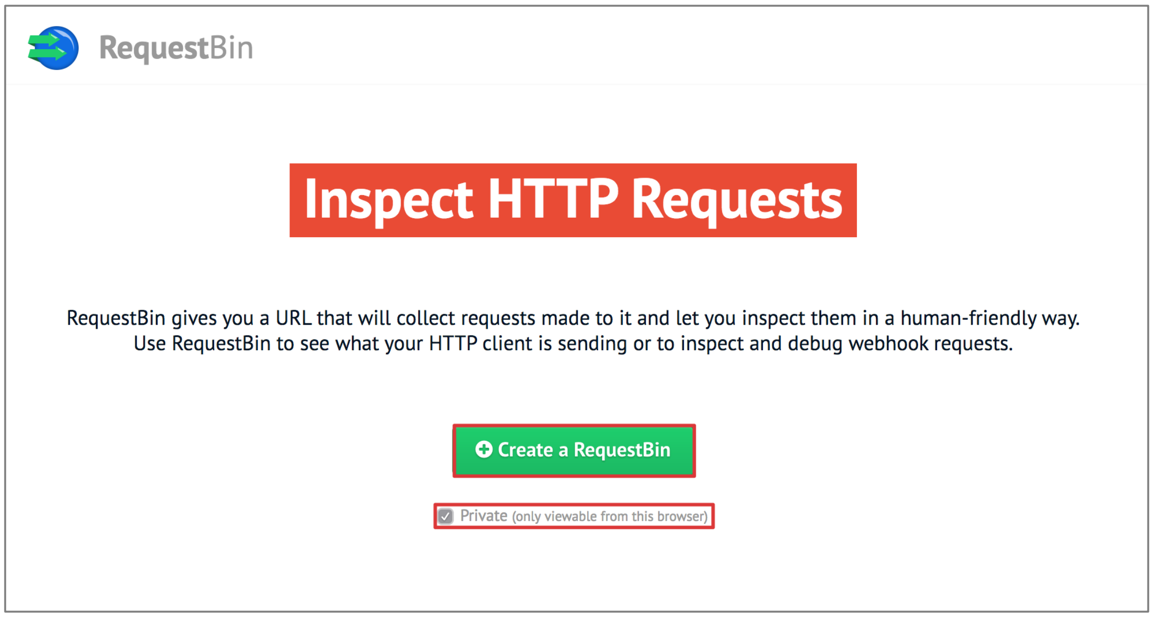
Lab 2 Test Webhooks with RequestBin

About RequestBin

RequestBin is a free testing tool provided by Runscope. It allows you to create a unique mock endpoint that will accept arbitrary web service requests. You can then examine the details of the requests. We will use RequestBin in this lab to accept Webhooks from GitHub and then we will inspect the values sent by GitHub.

Create a new RequestBin

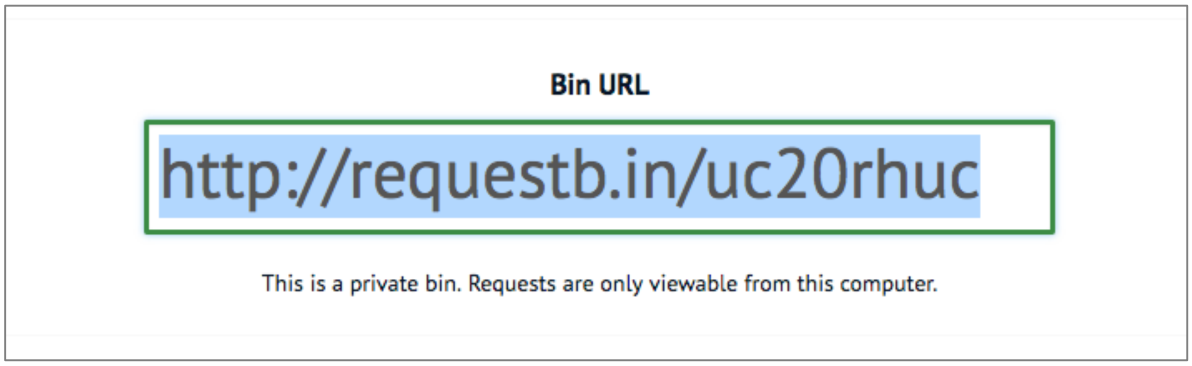
1. Navigate to https://requestb.in/
2. Click the **Private** checkbox.
3. Click the **Create a RequestBin** button.



RequestBin gives you a URL that will collect requests made to it and let you inspect them in a human-friendly way.
Use RequestBin to see what your HTTP client is sending or to inspect and debug webhook requests.

© 2017 ServiceNow, Inc. All rights reserved.

4. Copy the **Bin URL**. This is the endpoint you will set in GitHub.



Bin URL

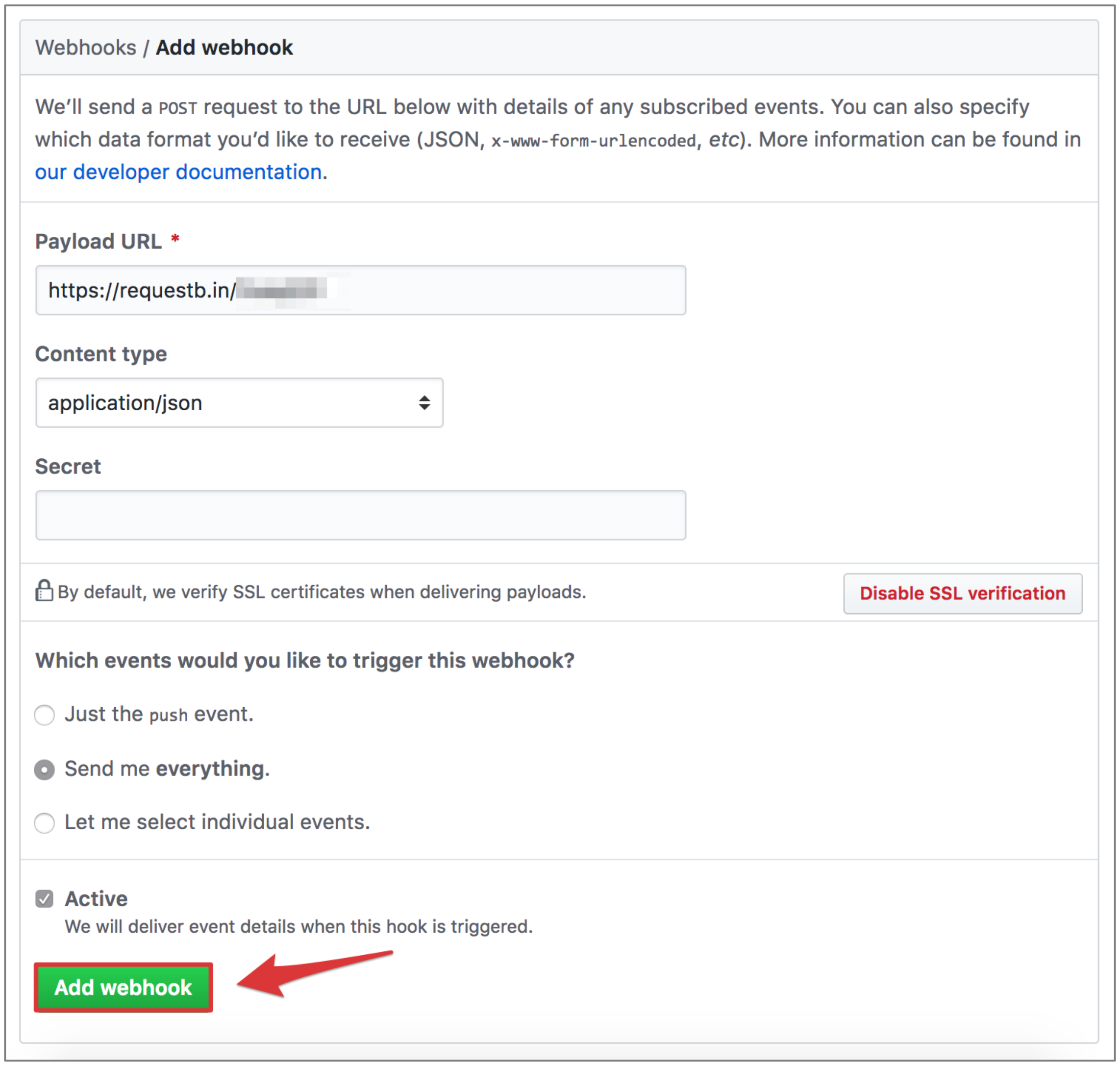
http://requestb.in/uc20rhuc

This is a private bin. Requests are only viewable from this computer.

5. Keep the RequestBin window open in its own tab. You will return to this tab several times.

Finish Configuring the GitHub Webhook

1. Return to the GitHub browser tab.
2. Paste the RequestBin URL into the **Payload URL** field.
3. Set the **Content type** to application/json.
4. Click the **Add webhook** button.



Webhooks / **Add webhook**

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://requestb.in/[redacted]

Content type

application/json

Secret

By default, we verify SSL certificates when delivering payloads.

Disable SSL verification

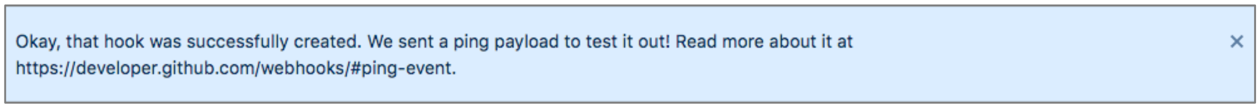
Which events would you like to trigger this webhook?

- [ ] Just the `push` event.
- [✓] Send me **everything**.
- [ ] Let me select individual events.

[✓] **Active**
We will deliver event details when this hook is triggered.

Add webhook

5. You should see the following message:



Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://developer.github.com/webhooks/#ping-event.

Check for the "ping" from GitHub

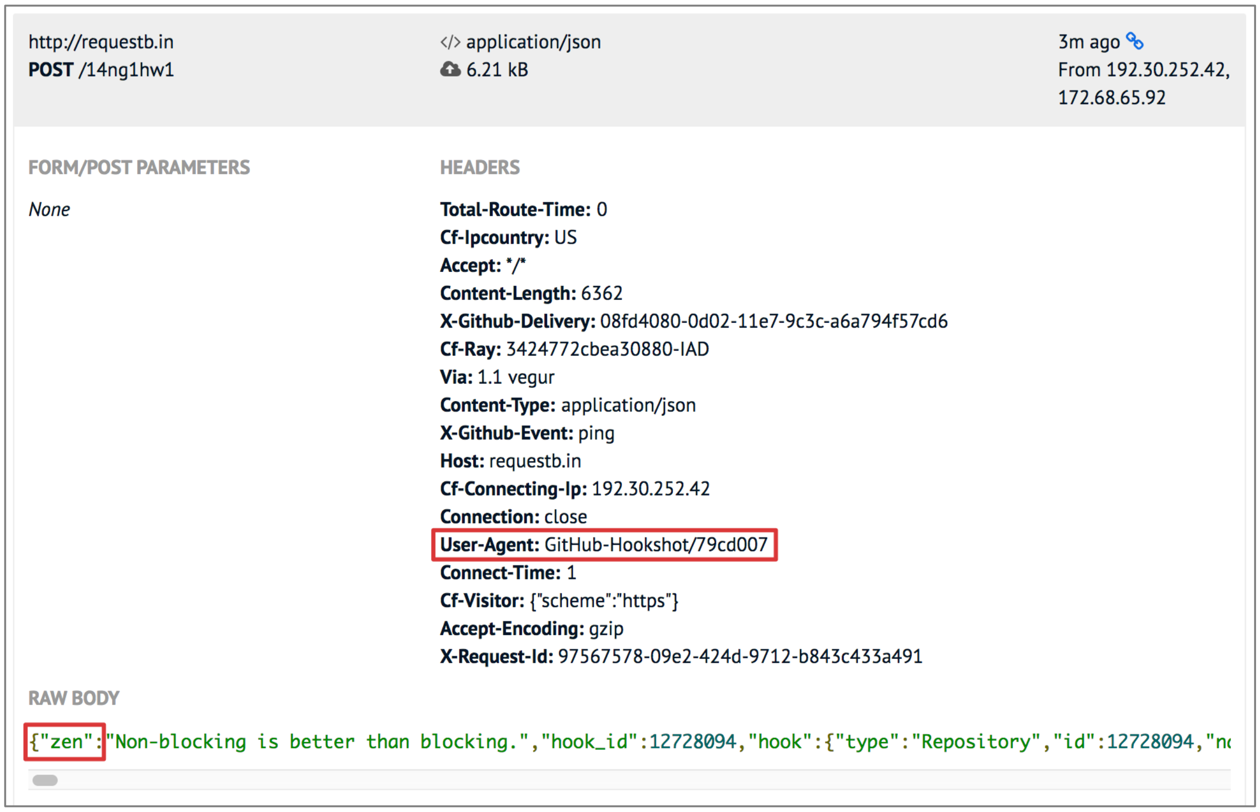
After you create a new webhook in GitHub, GitHub will send a "ping" request to the endpoint you specified to verify that things are working. You will take a look at this request now.

1. Return to the RequestBin tab.

2. Refresh the page.

3. You should now see a request in the list with a **GitHub-Hookshot** User-Agent.

4. Examine the **Raw Body** section of the request and make sure there is a JSON payload present. GitHub sends a zen payload with the ping request.



http://requestb.in
POST /14ng1hw1

application/json
6.21 kB

3m ago
From 192.30.252.42, 172.68.65.92

FORM/POST PARAMETERS

None

HEADERS

```
Total-Route-Time: 0
Cf-Ipcountry: US
Accept: */*
Content-Length: 6362
X-Github-Delivery: 08fd4080-0d02-11e7-9c3c-a6a794f57cd6
Cf-Ray: 3424772cbea30880-IAD
Via: 1.1 vegur
Content-Type: application/json
X-Github-Event: ping
Host: requestb.in
Cf-Connecting-Ip: 192.30.252.42
Connection: close
User-Agent: GitHub-Hookshot/79cd007
Connect-Time: 1
Cf-Visitor: {"scheme":"https"}
Accept-Encoding: gzip
X-Request-Id: 97567578-09e2-424d-9712-b843c433a491
```

RAW BODY

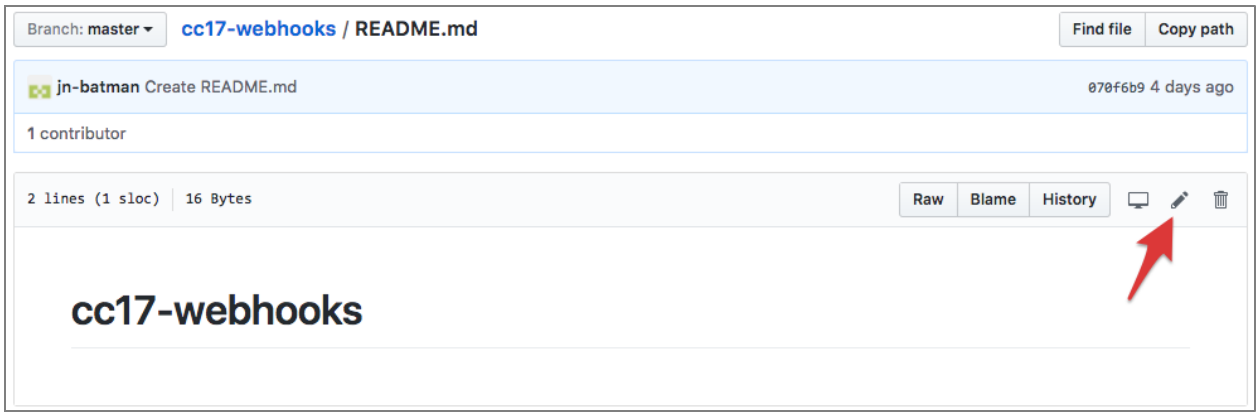
```
{"zen":"Non-blocking is better than blocking.","hook_id":12728094,"hook":{"type":"Repository","id":12728094,"n
```

Trigger a Webhook by updating README file in GitHub

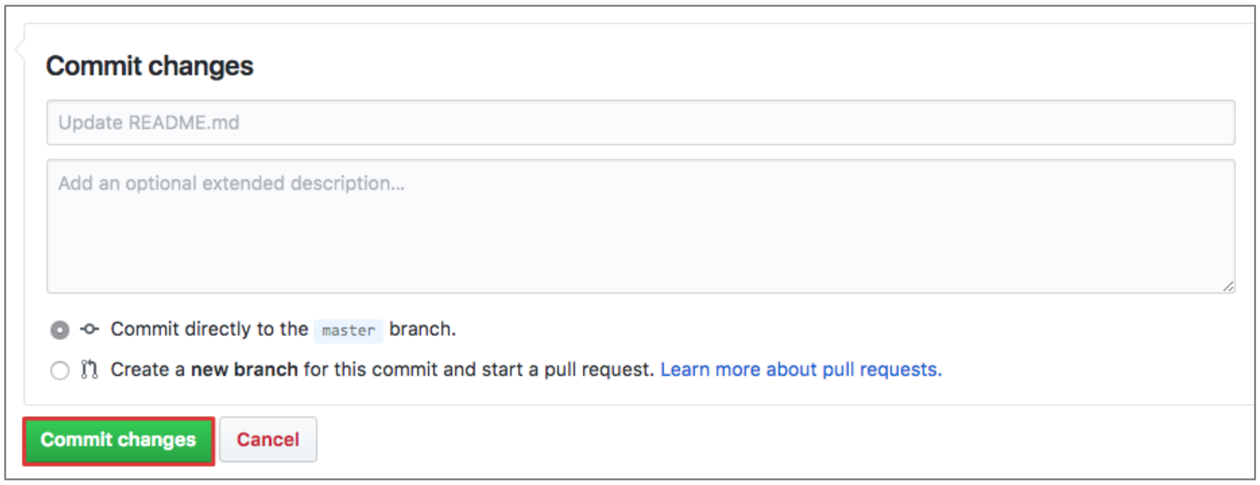
Now that you've verified the "ping" request from GitHub, you will trigger a *real* webhook. When you created the Webhook, you configured it to send **everything**. You will now trigger a **push** operation and inspect the resulting request from GitHub flow into RequestBin.

1. Navigate back to your cc17-webhooks GitHub repository and open the **README.md** file.

2. Click the **Edit this file** button (Pencil icon next to the Trash Can icon).



3. Add something interesting (or boring) to the file and click the **Commit changes** button.


Commit changes
Update README.md
Add an optional extended description...
Commit directly to the master branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel

4. Return to the RequestBin window and refresh the page to show the latest request.

5. Copy the JSON from the **Raw Body** section into your editor of choice and format the output for easier reading.

What just happened?

When you updated the README.md file and committed the changes, a Webhook was triggered. GitHub sent this webhook to the endpoint you specified. We triggered this directly from the GitHub interface, but the same thing would have happened if we pushed code from other Git clients.

CHALLENGE - Find the Commit URL

Using the payload sent by GitHub, see if you can identify the Commit URL that points to the changes you made to the README file.

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

You will build a Scripted REST API in ServiceNow to accept Webhooks sent by GitHub.

Lab 3
Build Scripted REST API

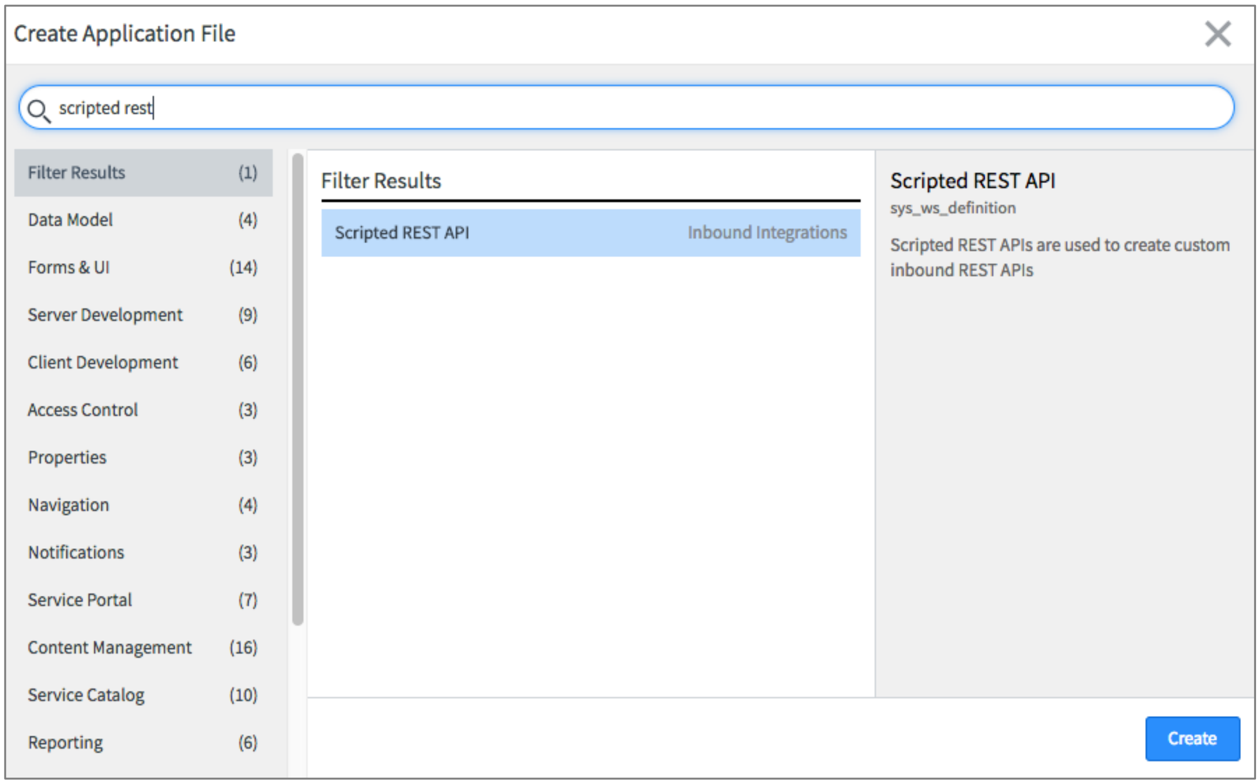
About Scripted REST APIs

ServiceNow gives you the tools to build custom REST APIs (referred to as **Scripted REST APIs**). These APIs can be configured to accept *any* payload and give you full control over how the payload is processed. This gives you, the developer, the flexibility to build an API that is compatible with any Webhook payload.

Create the Scripted REST API

1. Open **Studio** and select the application you imported earlier titled **CC17 Working with Webhooks**.

2. Click the **Create Application File** button, search for **Scripted REST API** and click the **Create** button.


Create Application File

scripted rest

Filter Results	(1)
Data Model	(4)
Forms & UI	(14)
Server Development	(9)
Client Development	(6)
Access Control	(3)
Properties	(3)
Navigation	(4)
Notifications	(3)
Service Portal	(7)
Content Management	(16)
Service Catalog	(10)
Reporting	(6)

Filter Results

Scripted REST API — Inbound Integrations

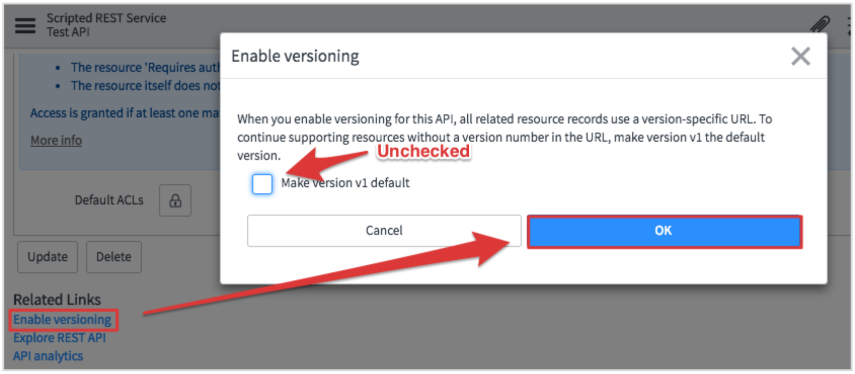
Scripted REST API
sys_ws_definition
Scripted REST APIs are used to create custom inbound REST APIs

© 2017 ServiceNow, Inc. All rights reserved.

3. Complete the Scripted REST API form and **Save** the record.

    - Name: **Webhook Handler**
    - API ID: **webhook_handler**

4. Under Related Links, click **Enable Versioning**, uncheck **Make version v1 default** and click the **OK** button on the resulting dialog.


Scripted REST Service
Test API

- The resource 'Requires auth
- The resource itself does not

Access is granted if at least one ma

More info

Enable versioning

When you enable versioning for this API, all related resource records use a version-specific URL. To continue supporting resources without a version number in the URL, make version v1 the default version.

Make version v1 default

5. Scroll to the **Resources** related list and click **New** to create a new Scripted REST Resource.

    - Name: **GitHub Webhook**
    - API version: **v1**
    - HTTP Method: **POST**
    - Relative path: **/github_webook**
    - Requires authentication: **Uncheck**

6. In the Script field:

    Note: This code can be copied from http://bit.ly/CC17-Webhooks-Snippets in `Lab_3_1_RestResource_Script.js`.

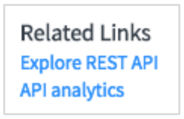
```
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {

    gs.info('GitHub Webhook URL: {0}', request.url);
    gs.info('GitHub Webhook Body: {0}', request.body.dataString);

})(request, response);
```

7. **Submit** the Scripted REST Resource.

© 2017 ServiceNow, Inc. All rights reserved.

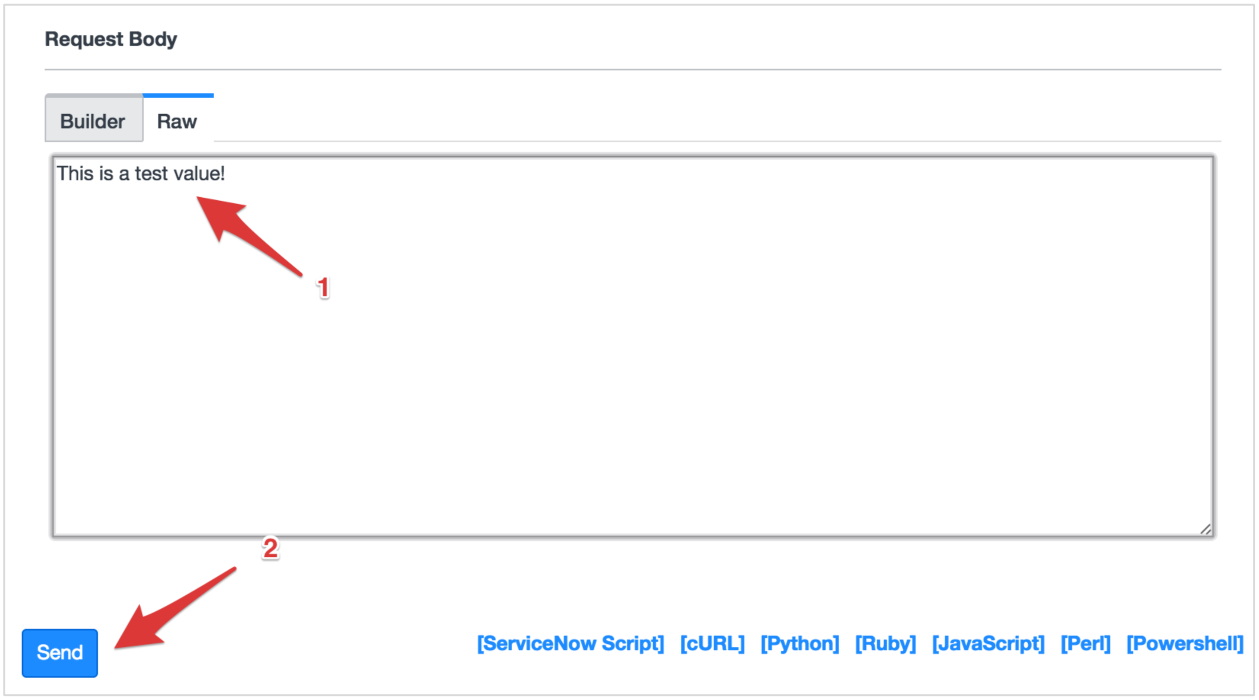
Test the Scripted REST Resource

1. From the **GitHub Webhook** Scripted REST Resource, click the **Explore REST API** Related Link.



2. Populate the Request Body / Raw tab with the value of your choosing.

3. Click the **Send** button.


Request Body
Builder
Raw
1
2
Send
[ServiceNow Script] [cURL] [Python] [Ruby] [JavaScript] [Perl] [Powershell]

This is a test value!

4. Examine the system logs and make sure you see output similar to the following:

```
>> GitHub Webhook URL: https://instance.service-
now.com/api/x_snc_cc17_webhook/v1/webhook_handler/github_webhook
>> GitHub Webhook Body: Test Body
```

Lab Goal

Create a new webhook in the cc17-webhooks repository to send events to the Scripted REST API you just built.

Lab 4
Create GitHub Webhook and Trigger Push

Create a New Webhook

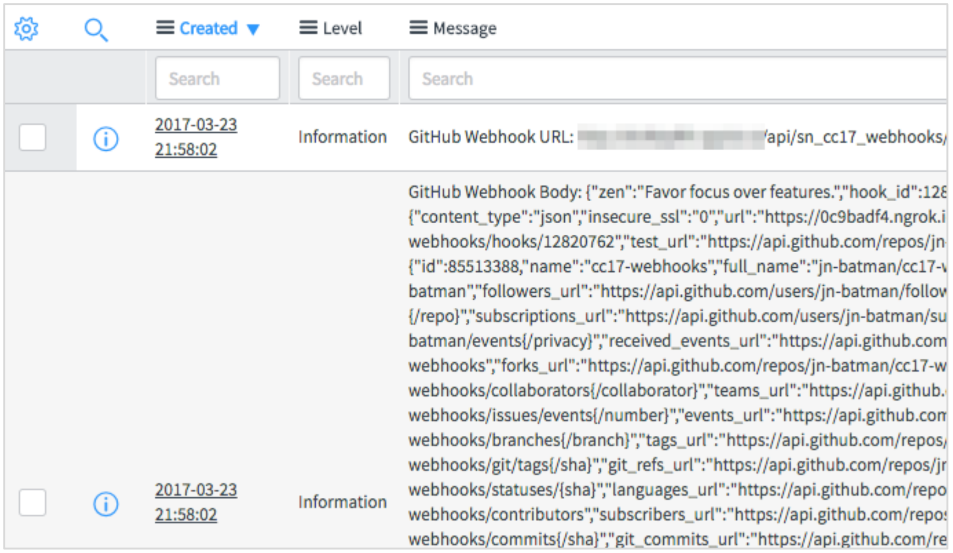
1. Follow the steps from **Lab 1** and **Lab 2** to add a **new** webhook to your GitHub repository.

2. Set the **Payload URL** to the full URL of your Scripted REST API. This will look something like:

   ```
   https://instance.service-now.com/api/x_snc_cc17_webhook/v1/webhook_handler/github_webhook
   ```

   Note: *Leave the existing RequestBin webhook in place. This may be useful for troubleshooting later if you run into issues.*

3. Examine the system logs and make sure you see the "ping" request from GitHub.



Created	Level	Message
Search	Search	Search
2017-03-23 21:58:02	Information	GitHub Webhook URL: [redacted]/api/sn_cc17_webhooks/
2017-03-23 21:58:02	Information	GitHub Webhook Body: {"zen":"Favor focus over features.","hook_id":128 {"content_type":"json","insecure_ssl":"0","url":"https://0c9badf4.ngrok.i webhooks/hooks/12820762","test_url":"https://api.github.com/repos/jn- {"id":85513388,"name":"cc17-webhooks","full_name":"jn-batman/cc17- batman","followers_url":"https://api.github.com/users/jn-batman/follow {/repo}","subscriptions_url":"https://api.github.com/users/jn-batman/su batman/events{/privacy}","received_events_url":"https://api.github.com webhooks","forks_url":"https://api.github.com/repos/jn-batman/cc17-w webhooks/collaborators{/collaborator}","teams_url":"https://api.github. webhooks/issues/events{/number}","events_url":"https://api.github.com webhooks/branches{/branch}","tags_url":"https://api.github.com/repos/ webhooks/git/tags{/sha}","git_refs_url":"https://api.github.com/repos/jr webhooks/statuses/{sha}","languages_url":"https://api.github.com/repo webhooks/contributors","subscribers_url":"https://api.github.com/repos webhooks/commits{/sha}","git_commits_url":"https://api.github.com/re

Modify README.md again

1. Follow the steps from **Lab 2** to modify the **README.md** file again and commit the changes.

2. Examine the system logs to verify the receipt of a webhook from GitHub indicating a push occurred.

Lab Goal

Now that you are receiving data from GitHub, it's time to do something useful with it. In this lab, you will build a "Webhook Stream" table that will store inbound payloads and give you the flexibility to do further processing on the data.

Lab 5 Build the Webhook Stream

The Webhook Stream

You could build business logic directly into the Scripted REST API to handle and process the payload from GitHub, but this is not an ideal solution. There are many event types that GitHub may send, and it would be preferable to avoid embedding too much logic into the Scripted REST API.

Moving this logic into a Script Include would be a big improvement, but still isn't quite as flexible as we'd like because it forces us to do all processing in code. What if we want to use a Workflow or Business Rule?

The **Webhook Stream** is a simple table that will store the payloads of inbound webhooks. Once saved, you can process the data any way you like. Options include:

- Write business rules that react to payloads containing certain values.
- Trigger notifications or workflows off of inbound requests.
- Easily explore the data sent by the external system.

© 2017 ServiceNow, Inc. All rights reserved.

Let's Build It!

Create a new table with the following attributes:

- Label: **Webhook Stream**
- Name: **x_snc_cc17_webhook_stream**
- Add module to menu: **-- Create new --**
- New menu name: **Webhook Stream**

Columns

Column label	**Type**	**Length**
ID	String	40
Source	String	40
Action	String	40
Payload	String	65535

If desired, switch to the main ServiceNow UI and make sure you can see the Webhook Stream menu and table.

© 2017 ServiceNow, Inc. All rights reserved.

Update the Scripted REST API

You will now update the Scripted REST API operation to write incoming payloads to the Webhook Stream.

1. Open the **GitHub Webhook** Scripted REST Resource.

2. Comment out the two gs.info statements.

3. Add the following script:

   Note: This code can be copied from http://bit.ly/CC17-Webhooks-Snippets in `Lab5_1_RestResource_Script.js`.

```
var grWebhook = new GlideRecord('x_snc_cc17_webhook_stream');
grWebhook.newRecord();
grWebhook.source = 'GitHub';

// Extract unique ID associated with this webhook
grWebhook.id = request.getHeader('X-GitHub-Delivery');

// Extract event type associated with this webhook
// See https://developer.github.com/v3/activity/events/types/
grWebhook.action = request.getHeader('X-GitHub-Event');

// Save the payload from the request body
grWebhook.payload = request.body.dataString;
grWebhook.insert();
```

4. **Save** the changes.

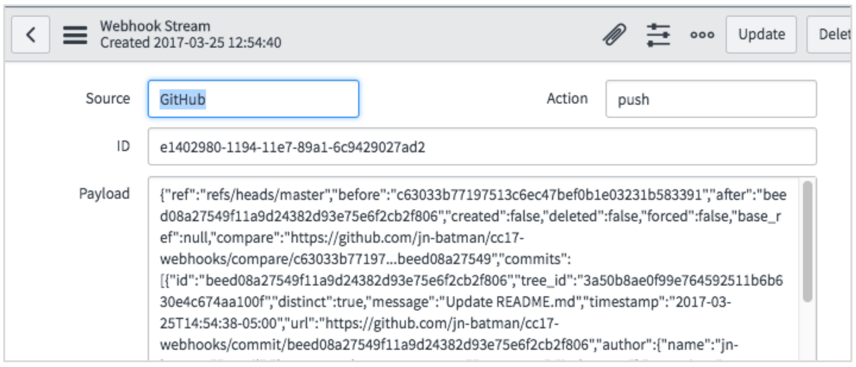
This is the foundation of the Webhook Stream! Now, any webhooks sent to the instance will be stored in this table, and the Source, Unique ID, Event and Payload will be captured.

© 2017 ServiceNow, Inc. All rights reserved.

Verify the Updated /github_webhook operation

1. Modify **README.md** again and commit the change.

2. Navigate to the **Webhook Stream** table and you should see a new record. Open this record and make sure all fields are populated.



Source: GitHub
Action: push
ID: e1402980-1194-11e7-89a1-6c9429027ad2
Payload:

{"ref":"refs/heads/master","before":"c63033b77197513c6ec47bef0b1e03231b583391","after":"beed08a27549f11a9d24382d93e75e6f2cb2f806","created":false,"deleted":false,"forced":false,"base_ref":null,"compare":"https://github.com/jn-batman/cc17-webhooks/compare/c63033b77197...beed08a27549","commits":[{"id":"beed08a27549f11a9d24382d93e75e6f2cb2f806","tree_id":"3a50b8ae0f99e764592511b6b630e4c674aa100f","distinct":true,"message":"Update README.md","timestamp":"2017-03-25T14:54:38-05:00","url":"https://github.com/jn-batman/cc17-webhooks/commit/beed08a27549f11a9d24382d93e75e6f2cb2f806","author":{"name":"jn-

Implement Additional Business Logic

You will now create a workflow on the Webhook Stream table to process "push" events from GitHub.

1. From Studio, create a new Workflow.

   - Name: **GitHub Push Process**
   - Table: **Webhook Stream**
   - Condition: **Source is GitHub AND Action is push**

2. Drag a **Run Script** activity onto the workflow canvas. Name it **Build Message Body** and populate the **Script** field with the following script:

```
var payloadObject = JSON.parse(current.payload);
var message = "";
payloadObject.commits.forEach(function (commit) {
    message += "Message: " + commit.message      + "\n";
    message += "URL: "     + commit.url          + "\n";
    message += "Author: "  + commit.author.email + "\n";
    message += "------------------------------------\n";
});

workflow.scratchpad.message = message;
```

Note: This code can be copied from http://bit.ly/CC17-Webhooks-Snippets in `Lab5_2_Workflow_Script.js`.

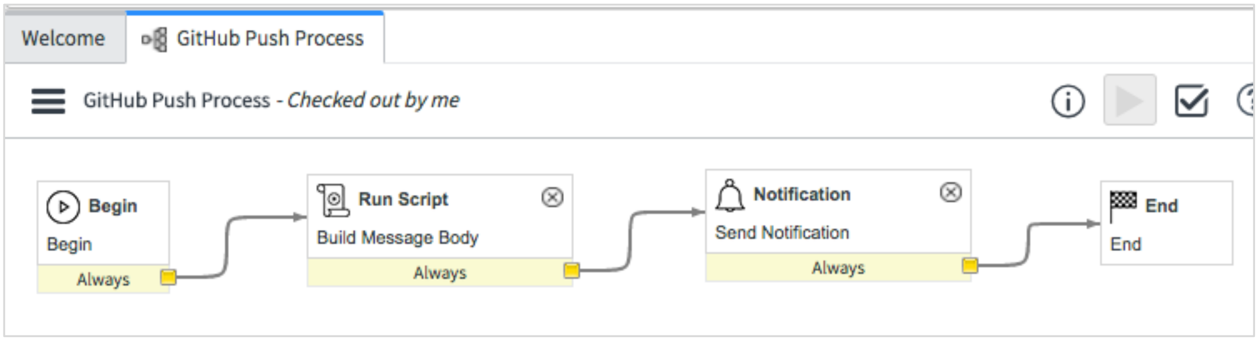
© 2017 ServiceNow, Inc. All rights reserved.

3. Drag a **Notification** activity onto the workflow between the **Run Script** and **End** activities and configure it with the following values:

   - Name: **Send Notification**
   - To: **System Administrator**
   - Subject: **New Push from GitHub**
   - Message: **${workflow.scratchpad.message}**

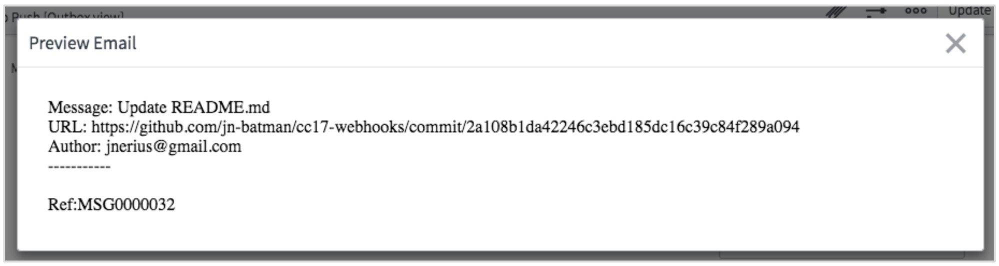
4. **Publish** the workflow.

   The workflow should now look like this:



Test the New Workflow

1. Update the **README.md** file yet again, and commit the changes.
2. Navigate to **System Mailboxes > Outbound > Outbox**.
3. Look for an outbound email with subject **New GitHub Push**. It may take a minute or two for this message to show up.
4. Open the record and click the **Preview HTML Body** Related Link.
5. The email preview should look similar to the following screenshot:



Message: Update README.md
URL: https://github.com/jn-batman/cc17-webhooks/commit/2a108b1da42246c3ebd185dc16c39c84f289a094
Author: jnerius@gmail.com

Ref:MSG0000032

© 2017 ServiceNow, Inc. All rights reserved.

Webhook Stream Summary

In this lab, we chose to build a Workflow to process a specific type of webhook, but there are many more things you could do with the Webhook Stream. Options include:

- Generate **Events**
- Trigger **Notifications**
- Run **Business Rules**

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

Webhook security varies from service to service. GitHub allows you to configure a "secret" value that is used to compute a hash signature that is sent with each webhook payload as a header called X-Hub-Signature. You can then re-compute this hash on the ServiceNow side to ensure the validity of the payload.

Lab 6 Bonus Implement Security: Secret Validation

Implement security check

1. From Studio, create a new System Property with suffix **github_secret**.

   ***Note**: the fully qualified property name should be **x_snc_cc17_webhook.github_secret**.*

2. Set the value of the new property to **ThisIsNotVerySecret**.

© 2017 ServiceNow, Inc. All rights reserved.

3. Add the following code and if/else condition to the **GitHub Webhook** REST API resource.

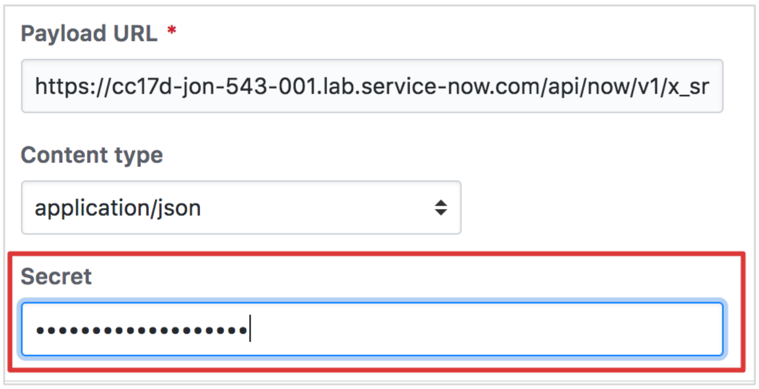
   Note: This code can be copied from http://bit.ly/CC17-Webhooks-Snippets in **`Lab6_1_Updated_RestResource_Script.js`**.

```
// The 'jsSHA' library has been pre-loaded into your lab instance as a Script
// Include. You can retrieve it from https://github.com/Caligatio/jsSHA if you
// wish to use it in your own instances.
var body   = request.body.dataString;
var secret = gs.getProperty('x_snc_cc17_webhook.github_secret');
var sha    = new global.jsSHA('SHA-1', 'TEXT');
sha.setHMACKey(secret, 'TEXT');
sha.update(body);
var hmac = sha.getHMAC('HEX');

// Now that we have a hash, compare it with the hash from the request
if (request.getHeader('X-Hub-Signature') == 'sha1=' + hmac) {

    // Move the rest of the processing logic here
    var grWebhook = new GlideRecord('x_snc_cc17_webhook_stream');
    grWebhook.newRecord();
    grWebhook.source  = 'GitHub';
    grWebhook.id      = request.getHeader('X-GitHub-Delivery');
    grWebhook.action  = request.getHeader('X-GitHub-Event');
    grWebhook.payload = body;
    grWebhook.insert();

} else {
    // The signature did not match, reject the request
    var unauthorized = new sn_ws_err.ServiceError();
    unauthorized.setStatus(401);
    unauthorized.setMessage('Invalid secret');
    response.setError(unauthorized);
}
```

© 2017 ServiceNow, Inc. All rights reserved.

Enable Security on the GitHub Side

From the GitHub repository, edit the webhook you created earlier. Set the **Secret** field to **ThisIsNotVerySecret** and click the **Update Webhook** button.


Payload URL *

https://cc17d-jon-543-001.lab.service-now.com/api/now/v1/x_sr

Content type

application/json

Secret

•••••••••••••••••••

Test the New Security Measures

1. You know the drill by now. Edit and commit **README.md**, and check the Webhook Stream table. If the webhook shows up there, everything worked as expected!

2. For bonus points, try changing the github_secret property to something that does not match the value you set on the GitHub side. Trigger a webhook and make sure the value does **not** make it into the Webhook Stream table.

© 2017 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 7
Bonus
Implement your own

In this optional lab, you will configure and process a webhook sent from a system of your choosing into ServiceNow. Suggestions:

- Twilio
- Dropbox
- Box.com
- Find one on your own

Hints

At a high level, follow these steps:

1. Identify the webhook you wish to implement.
2. Use RequestBin to send test events and to explore the data format, headers, HTTP method, etc.
3. Create a new Scripted REST Resource specific to that service, e.g. /dropbox_webhook.
4. Parse the incoming payload and store it in the Webhook Stream table.
5. If appropriate, apply security checks to validate incoming webhooks.
6. Create business logic in the form of a Workflow, Business Rule, or Notification to process the incoming data.

Show off what you've built! We'd love to see it.

© 2017 ServiceNow, Inc. All rights reserved.