CyberScienceLab

University of Guelph | Canada Cyber Foundry

LLM Gateway Routing For Shuffler UI Apps

Introduction

The team at Cyber Science Lab added an option for routing requests through the LLM Gateway during app creation in Shuffler[1]. The "Use Gateway" toggle has been added to the app-building form, which is disabled by default. When the toggle is enabled, everything about your app remains the same, but during execution your requests will first be routed to the LLM Gateway for processing. The gateway will then relay your request to the specified LLM, receive and process the response and send it back to your app. If the toggle is turned off your app will function as it always has, without any differences in behavior.
Note: You can toggle this option on or off at any time through the Shuffler UI app editor.

Example

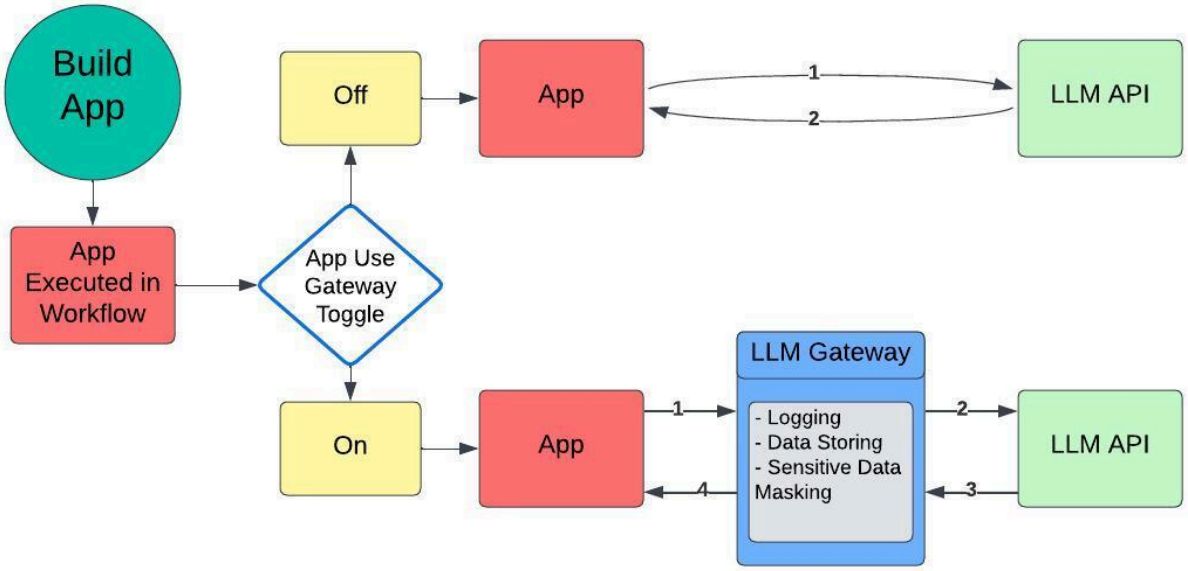


Diagram for Gateway Routing

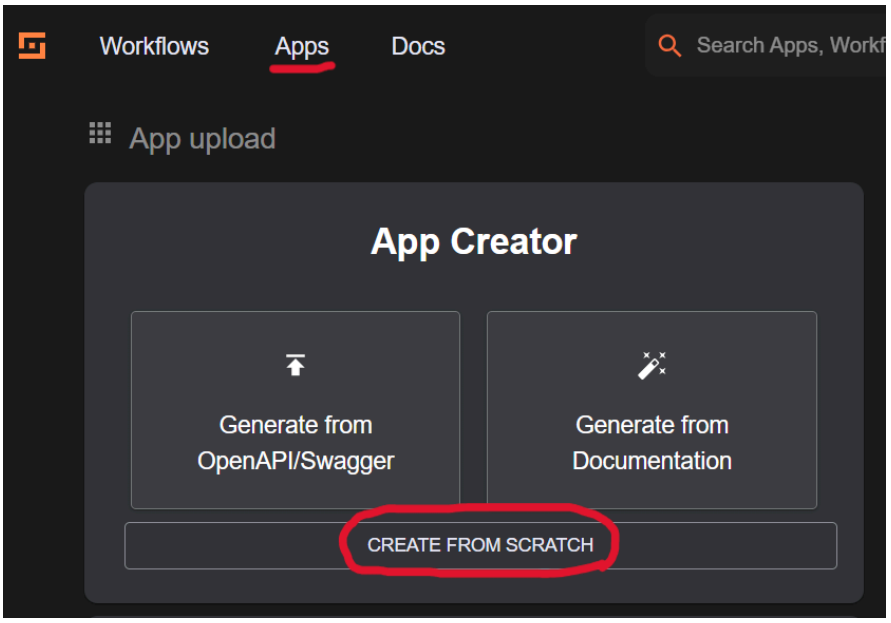
The app is built through the Shuffler custom UI. Then once your app is executed in a workflow, the behavior depends on whether the "Use Gateway" toggle was enabled during app creation. If the toggle was disabled, the app directly makes an HTTP request to the LLM API and receives the response. If enabled, the app makes an HTTP request to the LLM Gateway, which processes the data and forwards the request to the LLM API. The Gateway then processes the response from the LLM and sends it back to the app.

---

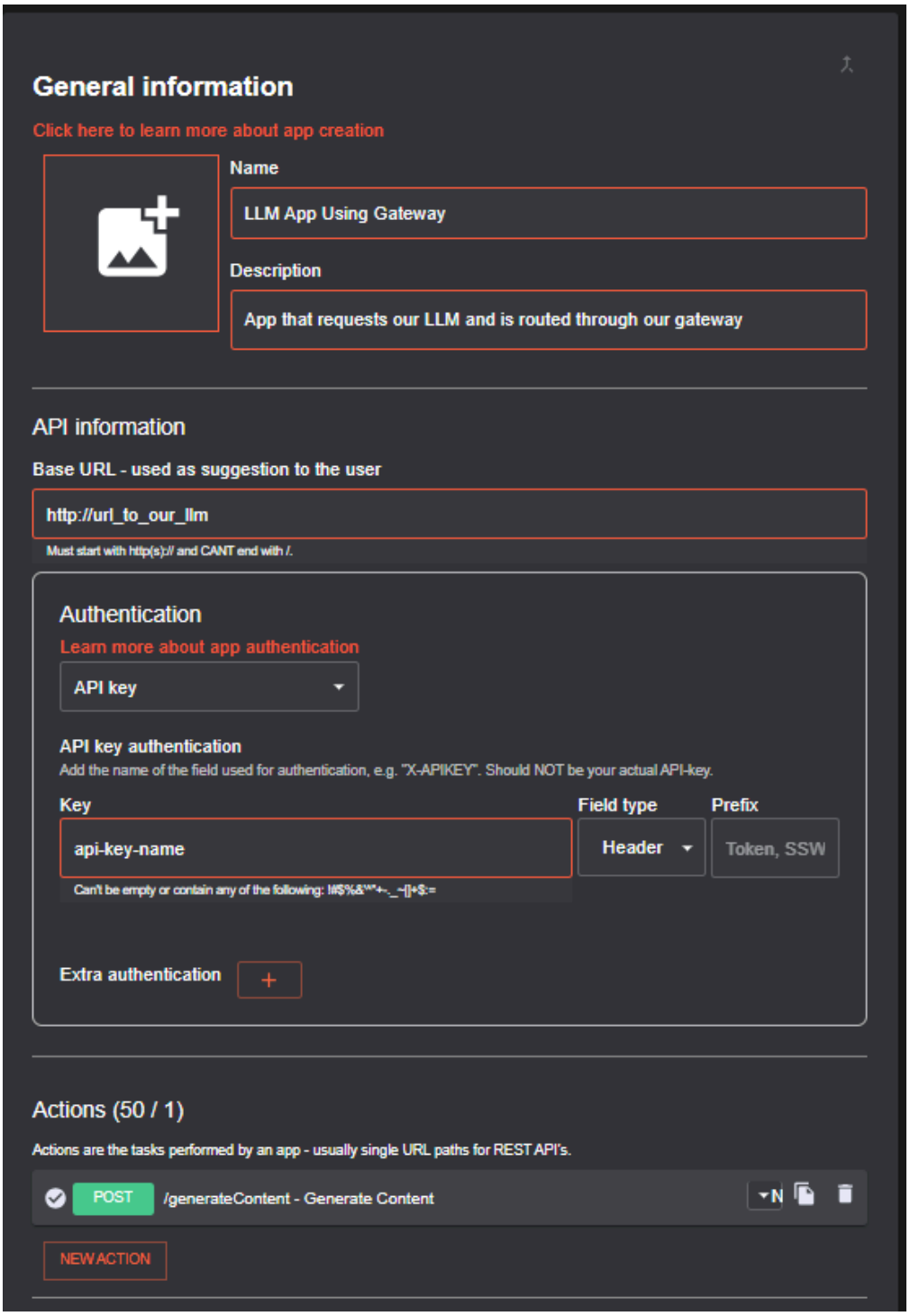
[1] This work was supported by the contributions of Mark Schmid (Undergraduate Researcher), Hadiseh Moradisani (PhD Researcher), Dr. Abbas Yazdinejad (Postdoctoral Research Fellow), and Dr. Ali Dehghantanha (Director).

Steps to Use Our Gateway

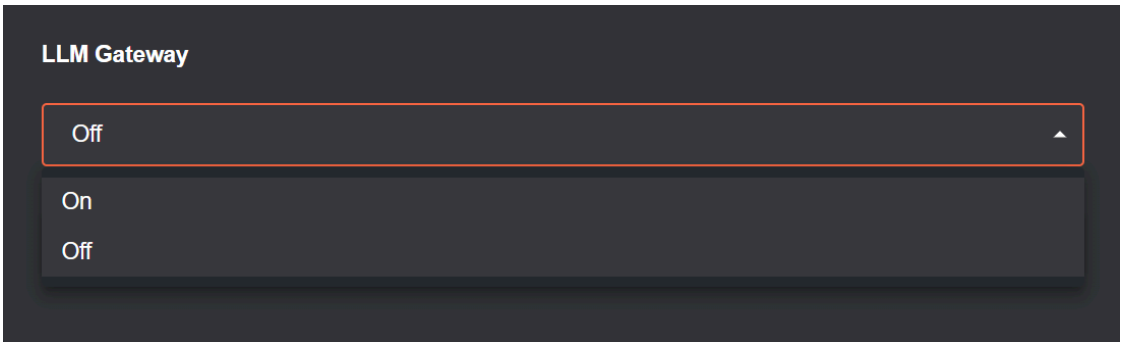
1. Create an app using the Shuffler UI app creator



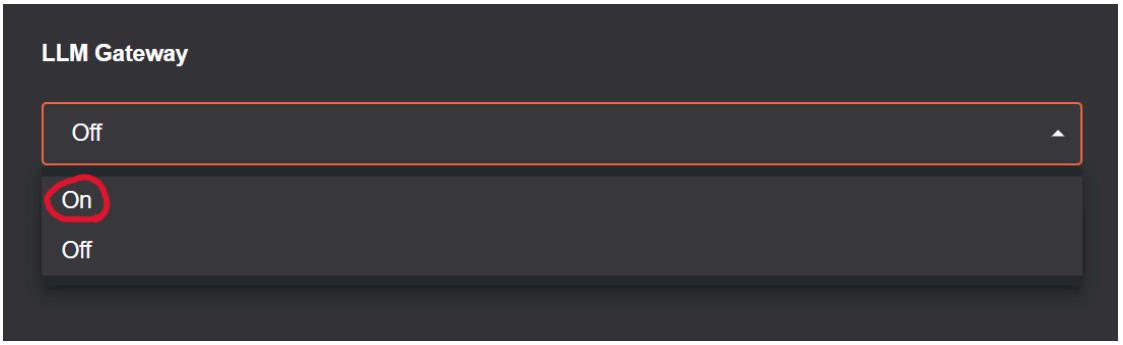
2. Fill in all fields like creating a normal app to use your LLM



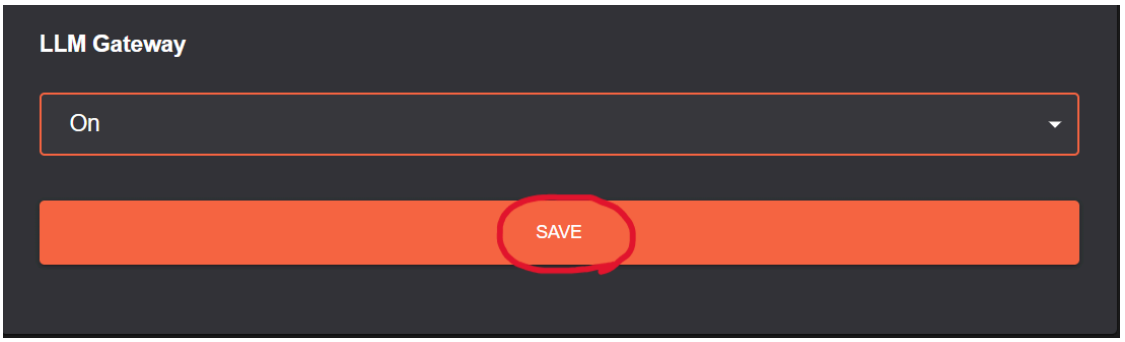
3. Select the LLM Gateway toggle



4. Set toggle to 'On'



5. Save Application



6. You can now use your app in a workflow as you would a normal app, but when it's executed requests will first be routed to LLM Gateway for processing before being sent to the LLM.
   Note: toggle can be turned on and off at any time through the app editor.

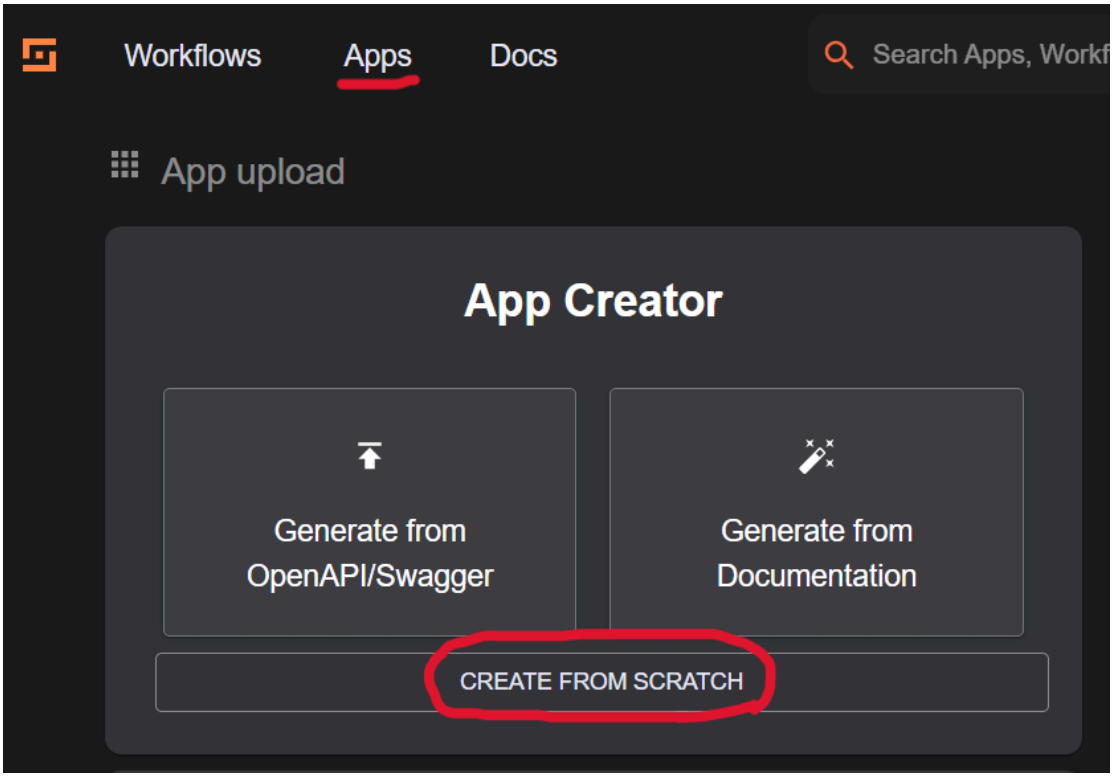
Simple Gemini Workflow

This workflow below will require two steps:

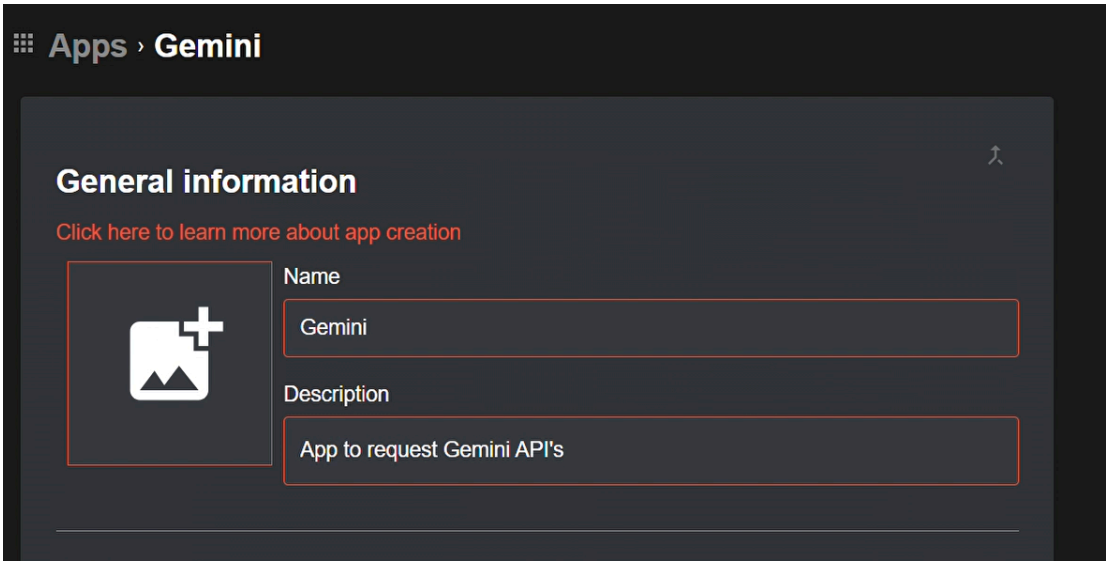
- Step 1 - Build Gemini app
- Step 2 - Build simple workflow using our Gemini app

Step 1 - Build Gemini app

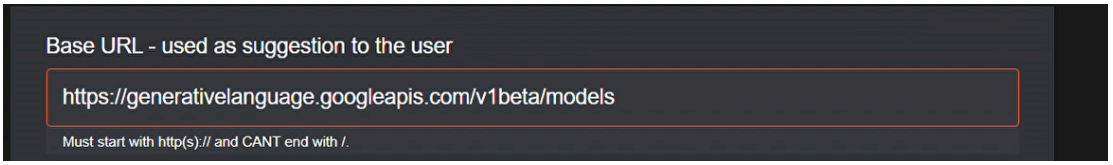
1. Create an app using the Shuffler UI app creator



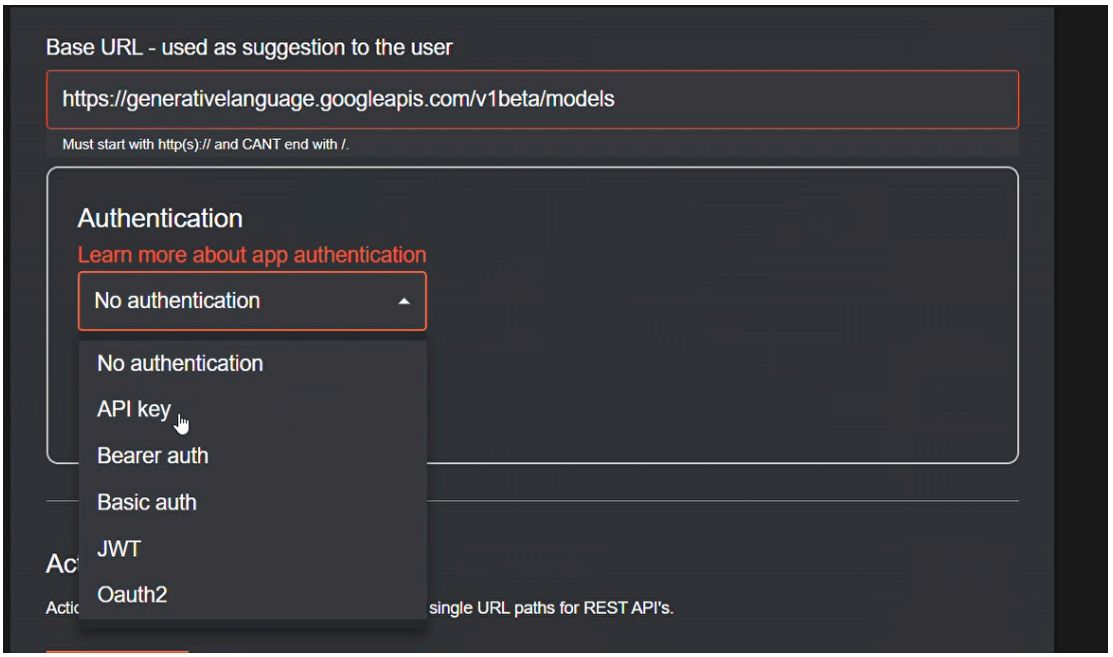
2. Give your app a name and a description



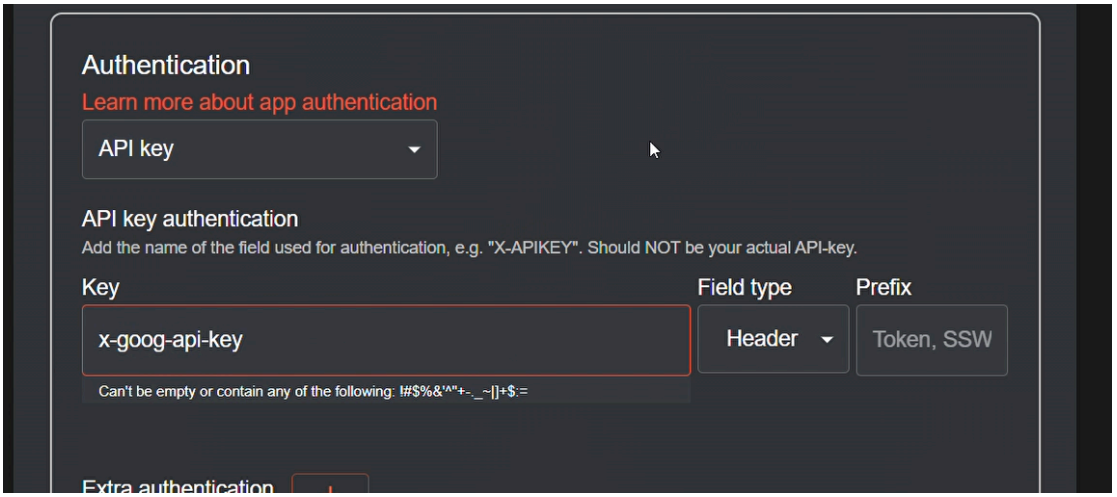
3. Add the base URL to Google's generated APIs



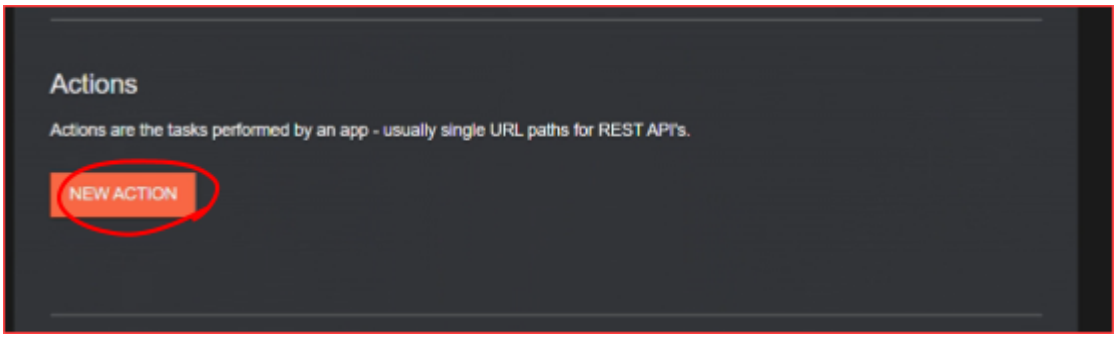
4. Select your app authentication, for Gemini we'll use an API key



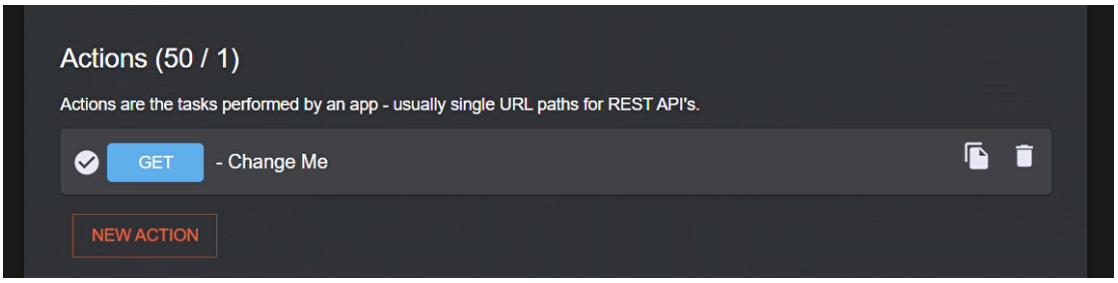
5. Add the API key name, google expects 'x-goog-api-key'



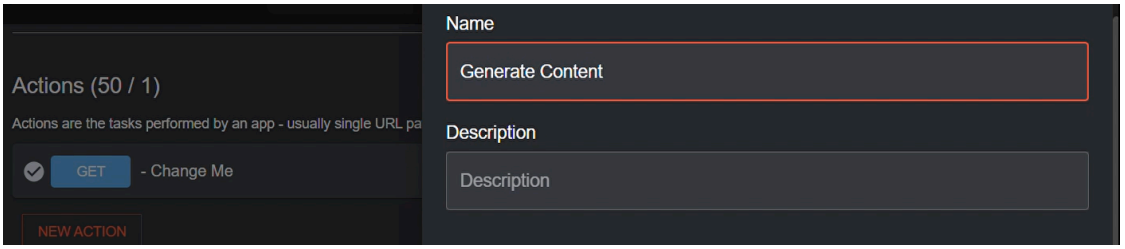
6. Now we'll create our action which will be the requests we'll send to the Gemini API to prompt it and get the response



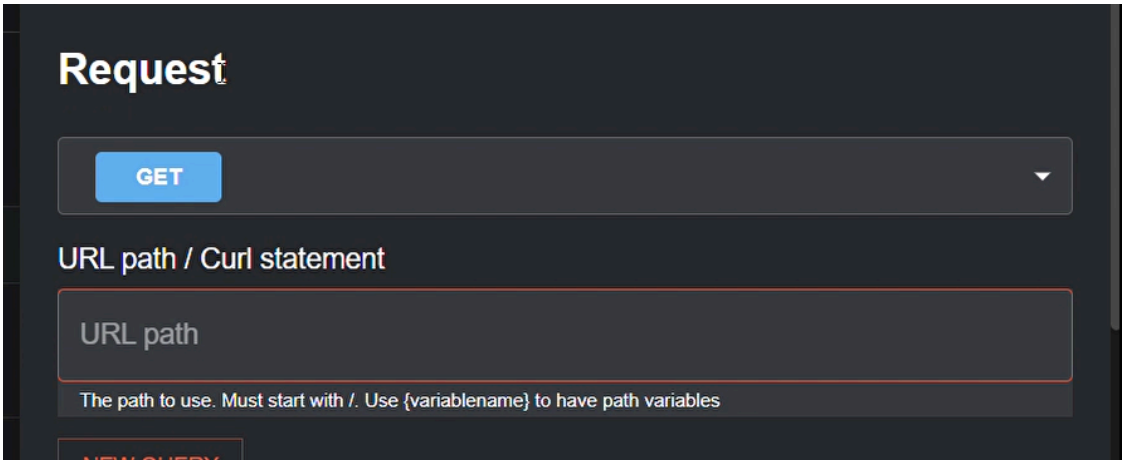
7. Click on the action to edit it



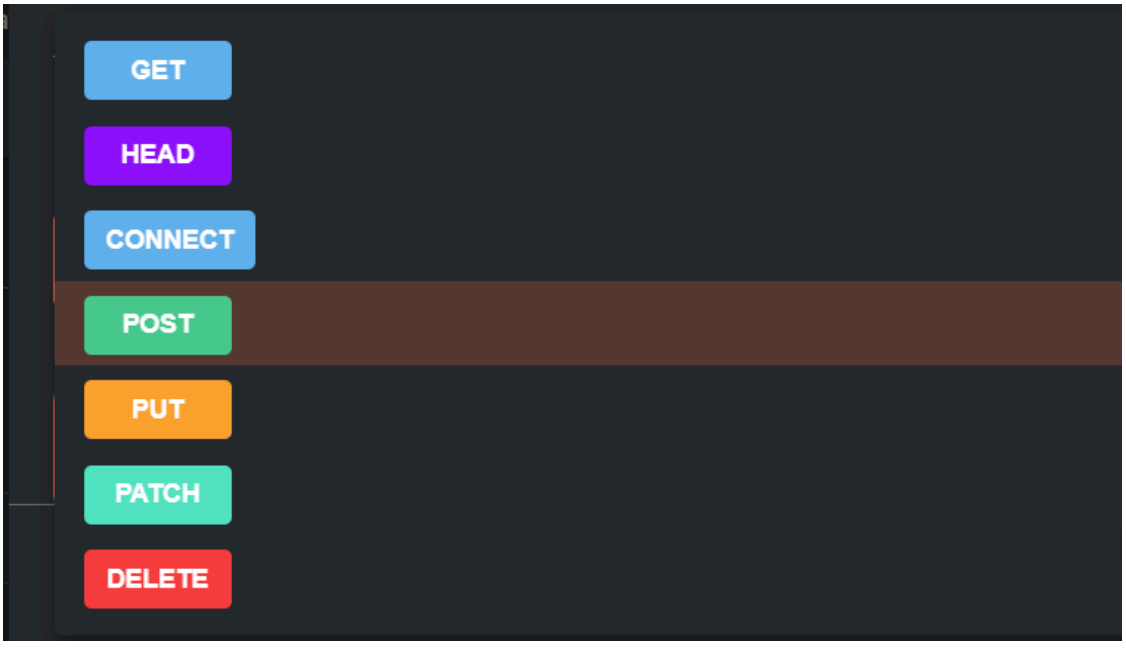
8. In the window that opens on the side, input all the information about our action. Start by filling in the 'Name' field with the action name



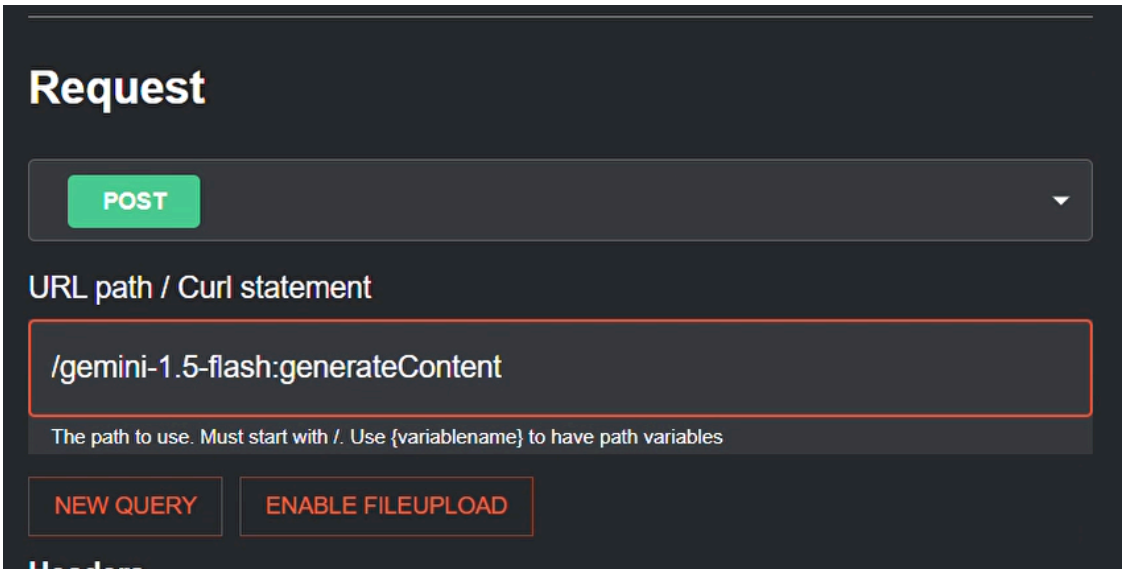
9. Select the HTTP method for your request by clicking the Request dropdown



10. Select 'POST'

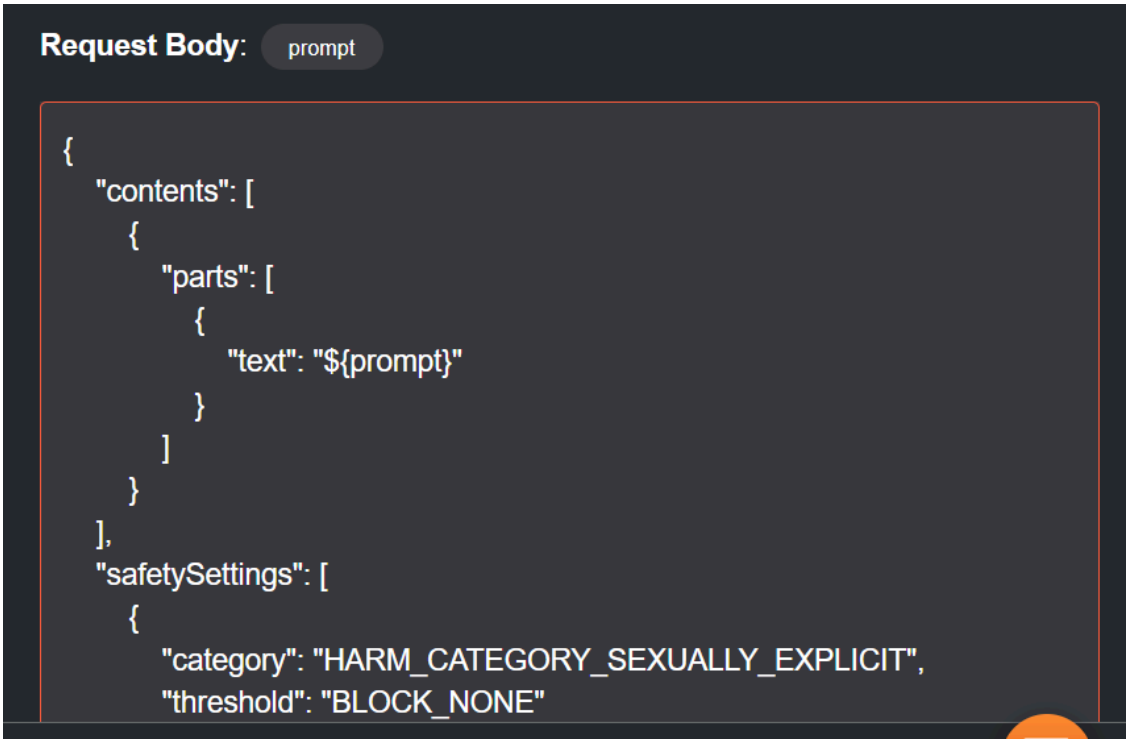


11. Next we'll specify the path for the endpoint we're using

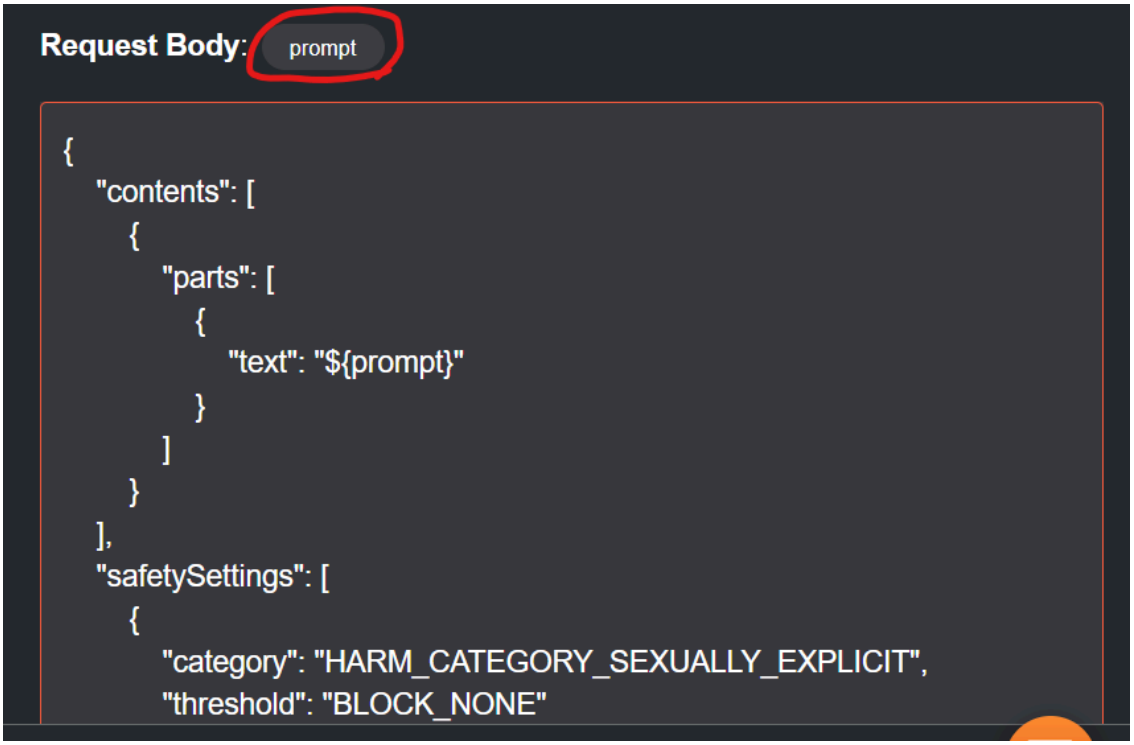


12. Now we'll provide the action's request body, which includes the text prompt for Gemini LLM as well as safety settings. Safety settings aren't required but are used to handle cases where questions regarding cybersecurity may be flagged. The request body can be copied below
13.

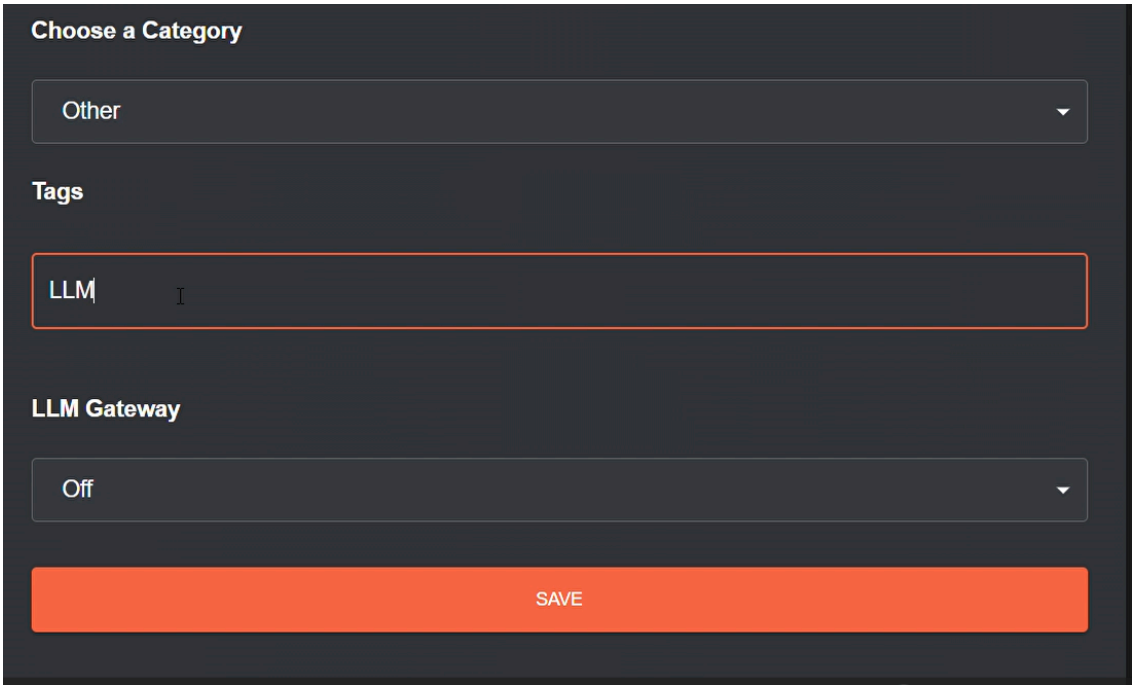
```
{
   "contents": [
      {
         "parts": [
            {
               "text": "${prompt}"
            }
         ]
      }
   ],
   "safetySettings": [
      {
         "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
         "threshold": "BLOCK_NONE"
      },
      {
         "category": "HARM_CATEGORY_HATE_SPEECH",
         "threshold": "BLOCK_NONE"
      },
      {
         "category": "HARM_CATEGORY_HARASSMENT",
         "threshold": "BLOCK_NONE"
      },
      {
         "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
         "threshold": "BLOCK_NONE"
      }
   ]
}
```



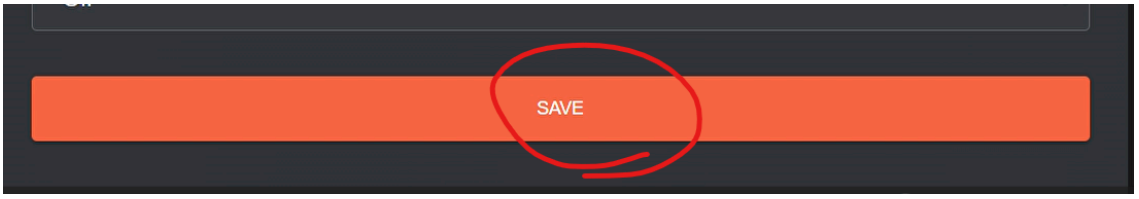
14. In the request body, the content.parts.text field is set to "${prompt}". This format with ${variableName} means that this value will be defined when the app is actually used. Since this is where Gemini gets its prompt response, ensure this field is required and always has a value. Click the prompt button



15. Finally, (optional step) select a category and tags to help sort the app and turn on LLM Gateway routing if desired

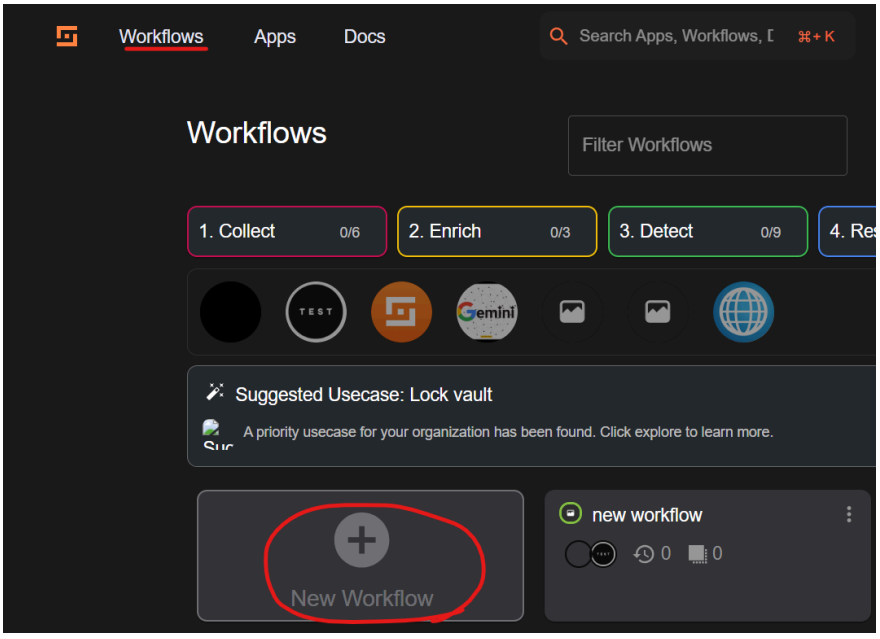


16. Save your app to complete the build

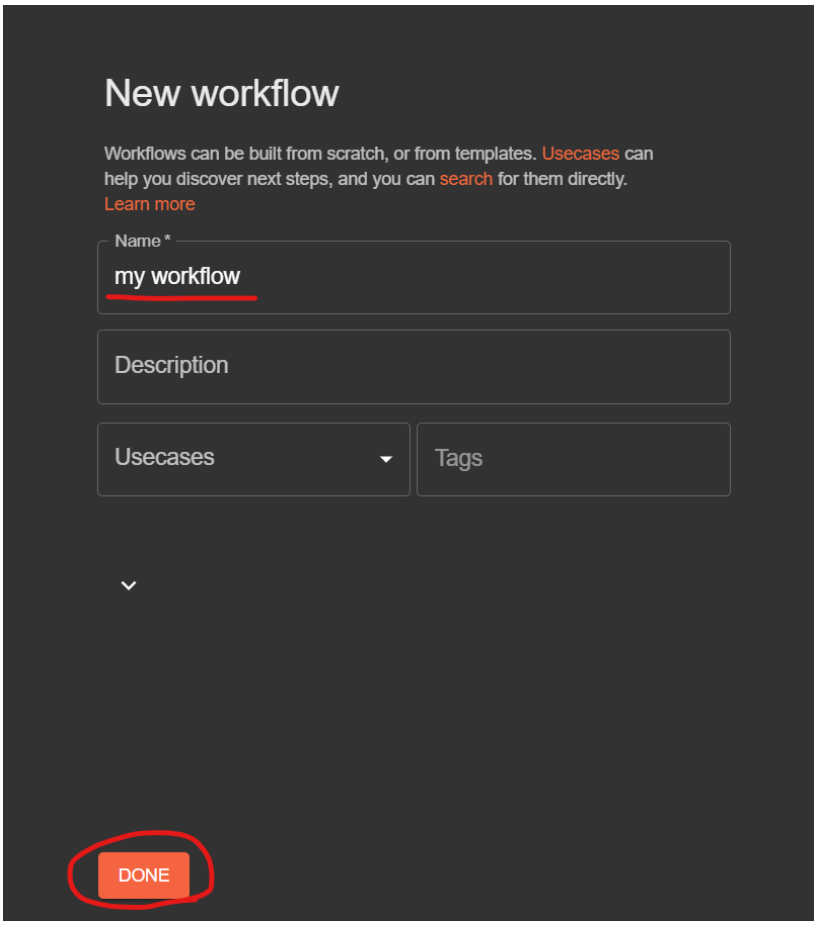


Step 2 - Build simple workflow using our Gemini app

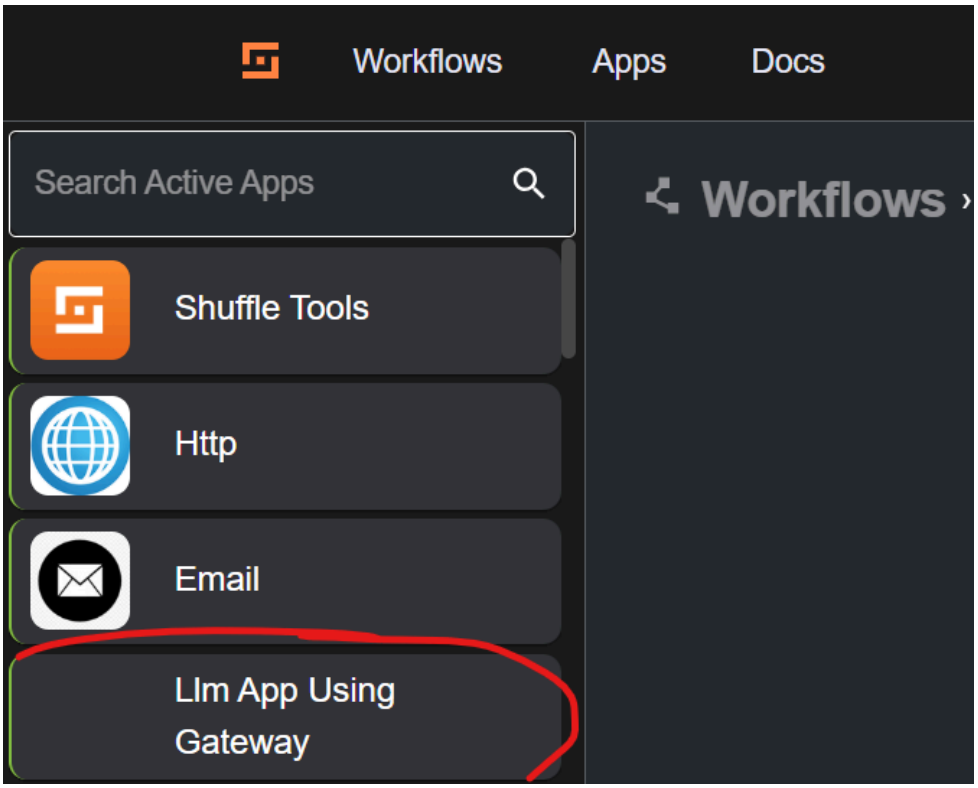
1. Navigate to the workflows page and click on 'New Workflow'



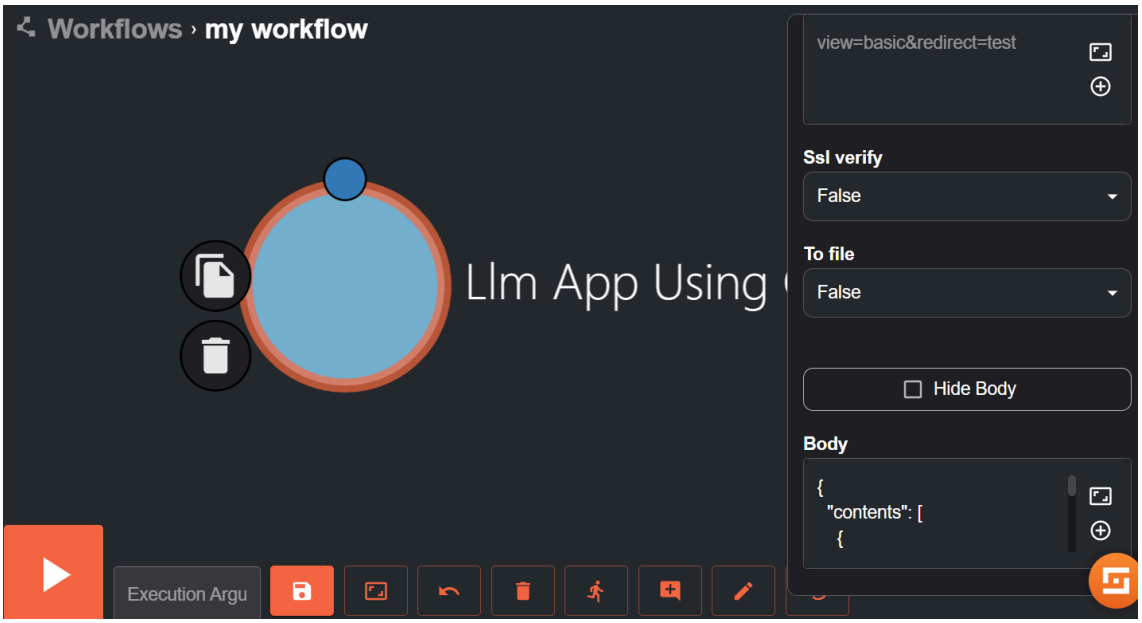
2. Give your workflow a name and click 'Done'



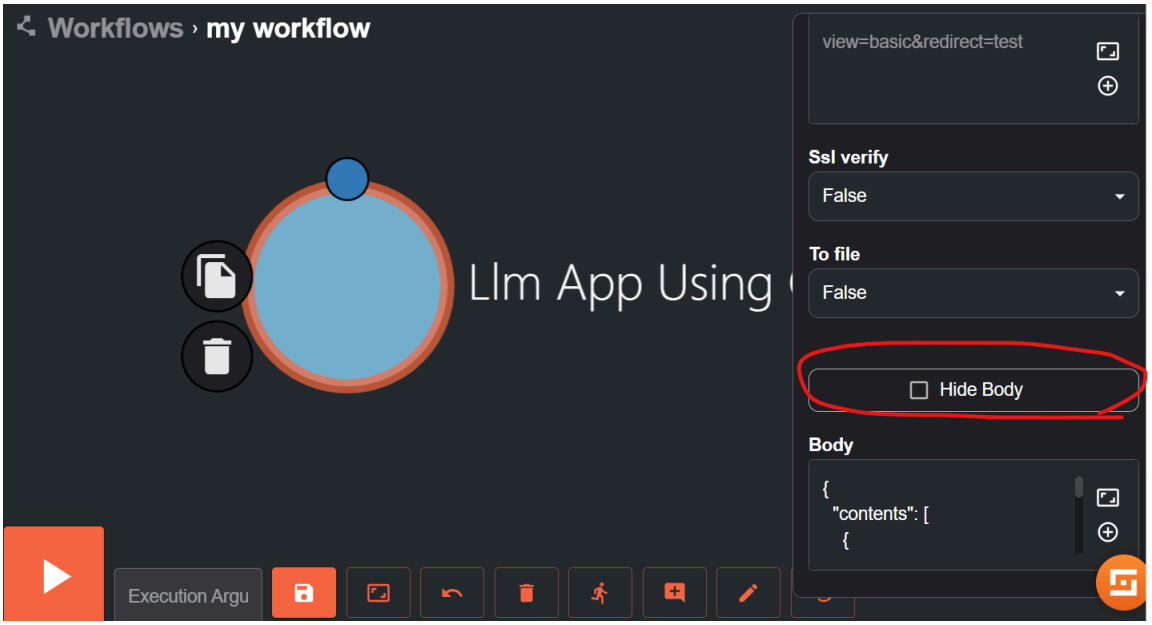
3. Find your app in the left apps sidebar. It should be visible. If not, enter your app name into the 'Search Active Apps' box



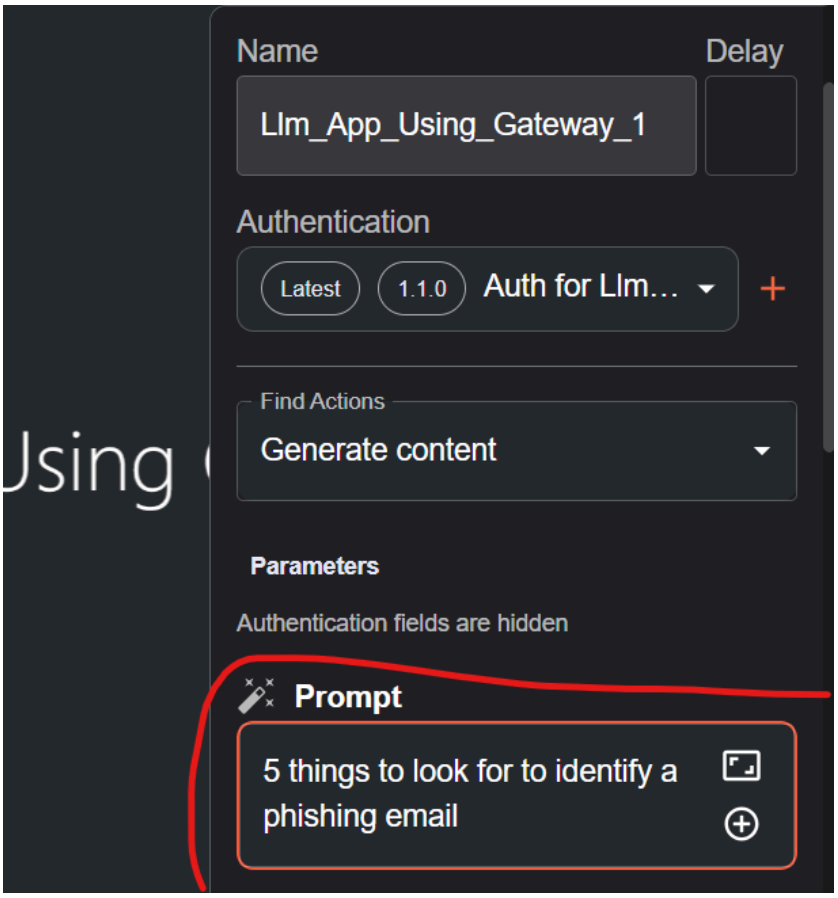
4. Click and drag your app onto the workflow board



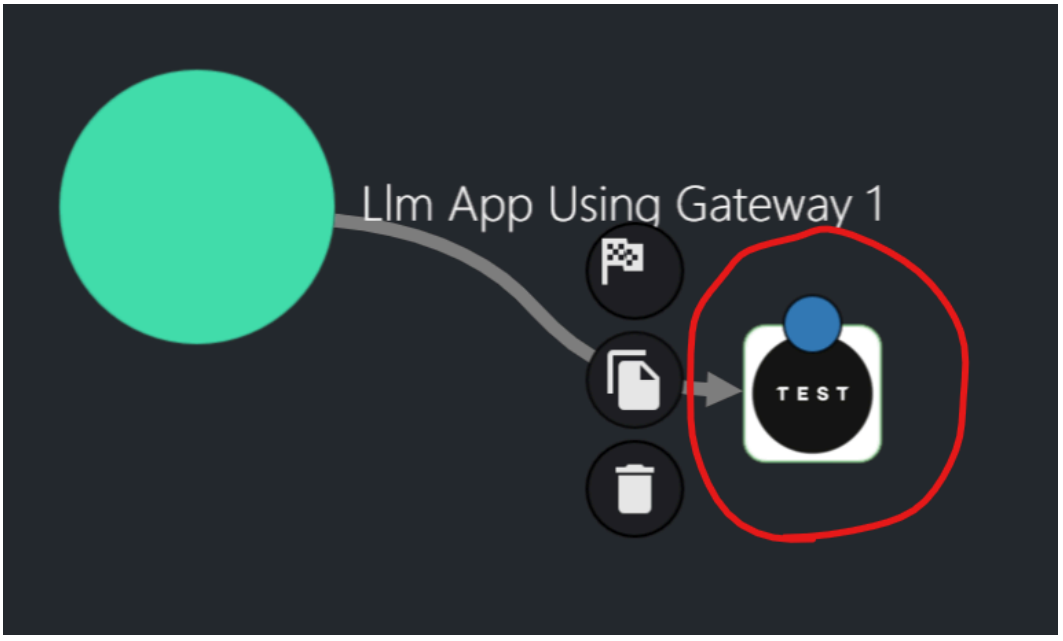
5. Scroll down to see the body text box. If you added variables to your action body, click 'Hide Body' to simplify the input process. If not, skip steps 5 and 6 and manually update the fields you want in the 'Body' text box



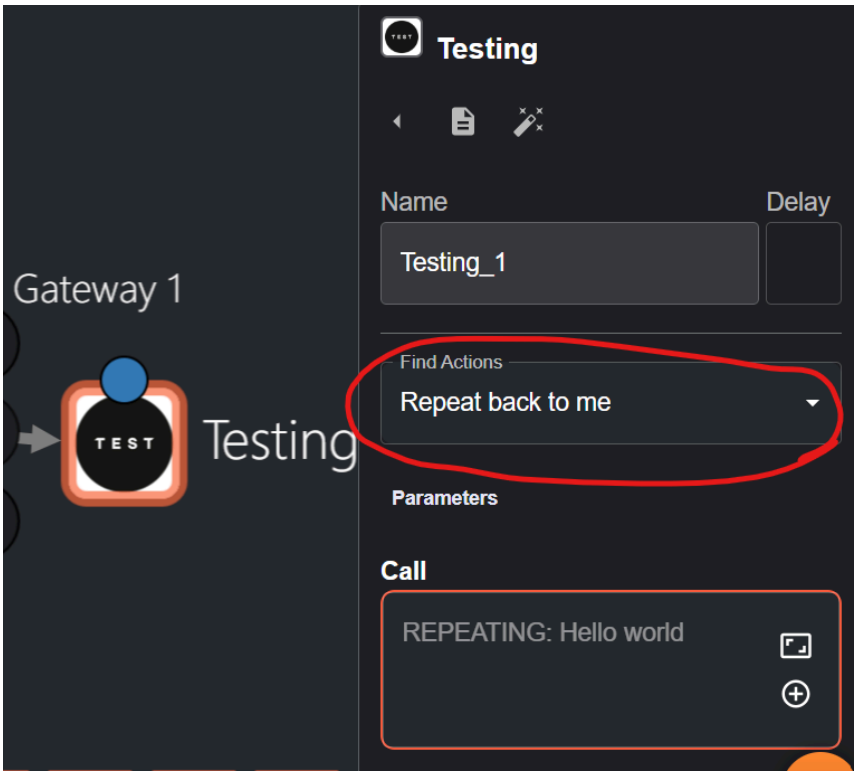
6. Enter the prompt for the LLM



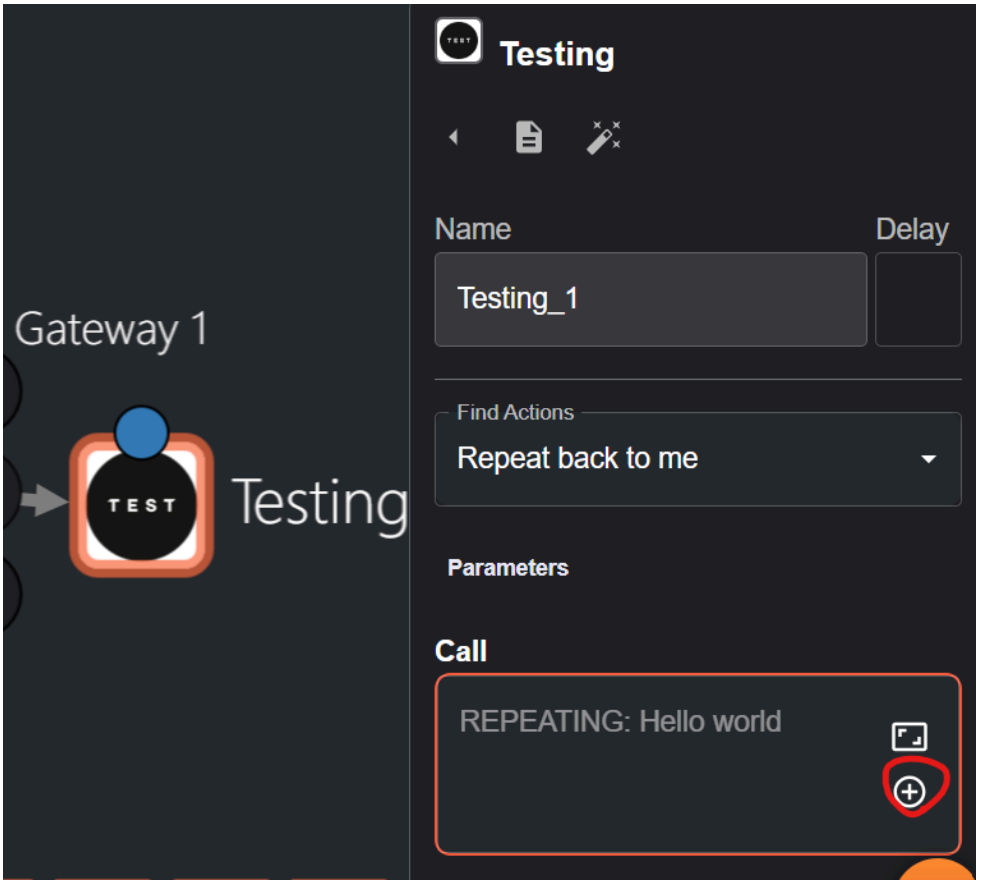
7. Steps below are for nicer output, if you just want to see the raw JSON response from the LLM skip to step 12. For a nicer output, find the 'Testing' app and drag it to the workflow board



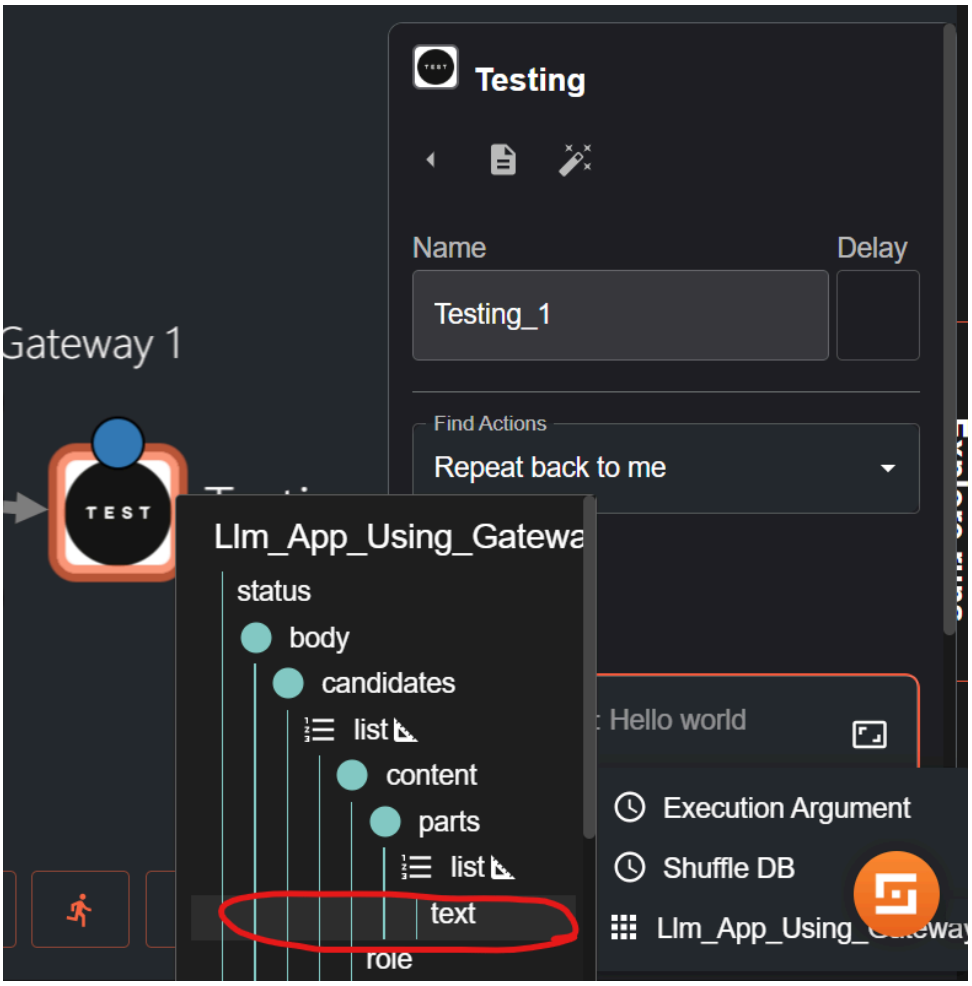
8. Click on 'Find Actions' and select the 'Repeat Back to Me' option



9. You can either manually list the path to the JSON node you want or use the Shuffler autocomplete. Use the autocomplete to select the field your response is stored in

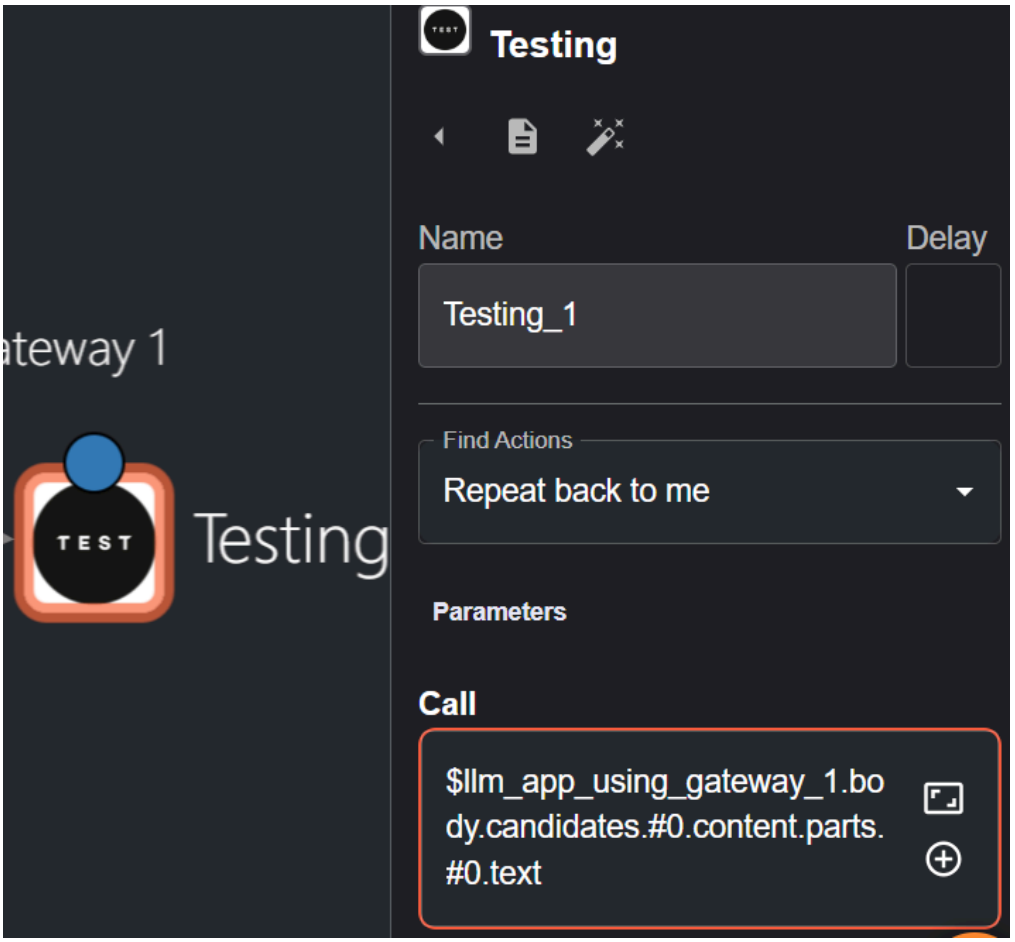


10. Select the field your response is stored in

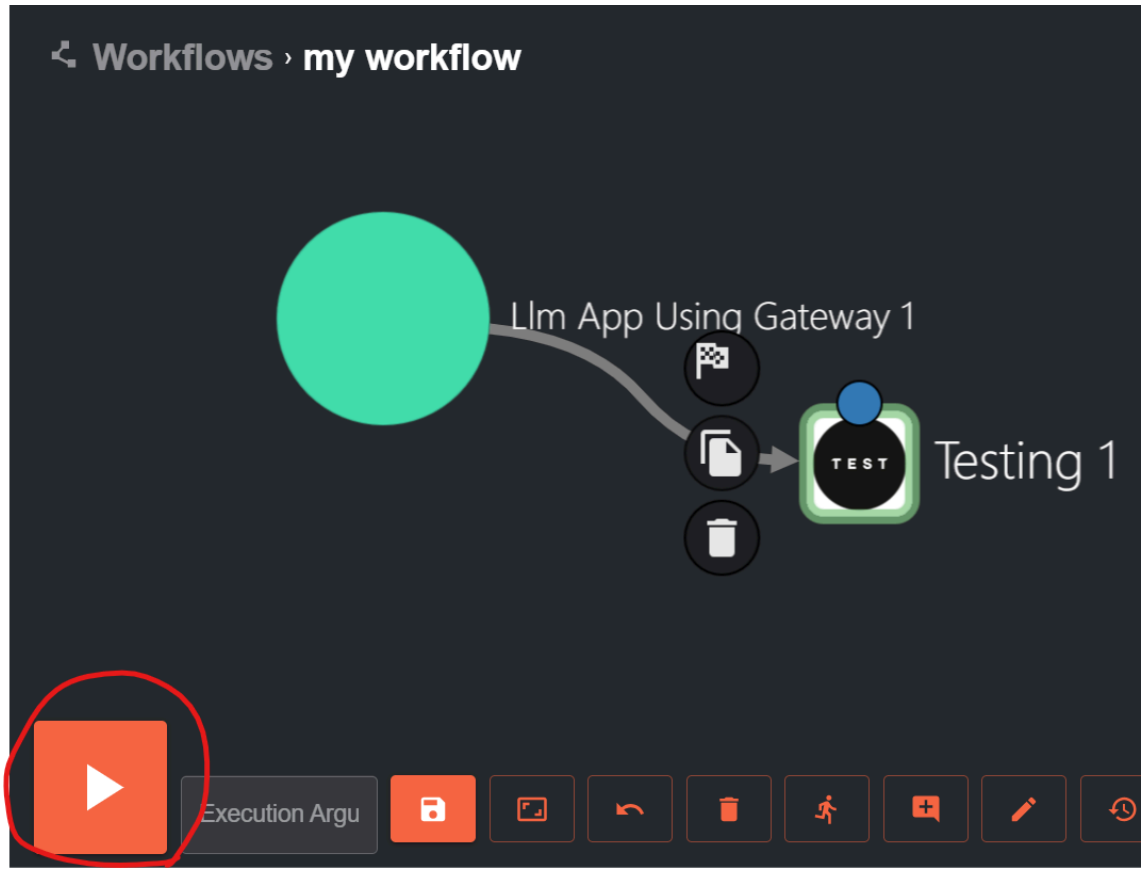


My response has array's inside of it and Shuffler defaults to visiting every array index, but for our case we want to use the first element for all arrays (that's the way Gemini works). So for our case we're going to use the first element for all arrays by adding '0's after the '#'

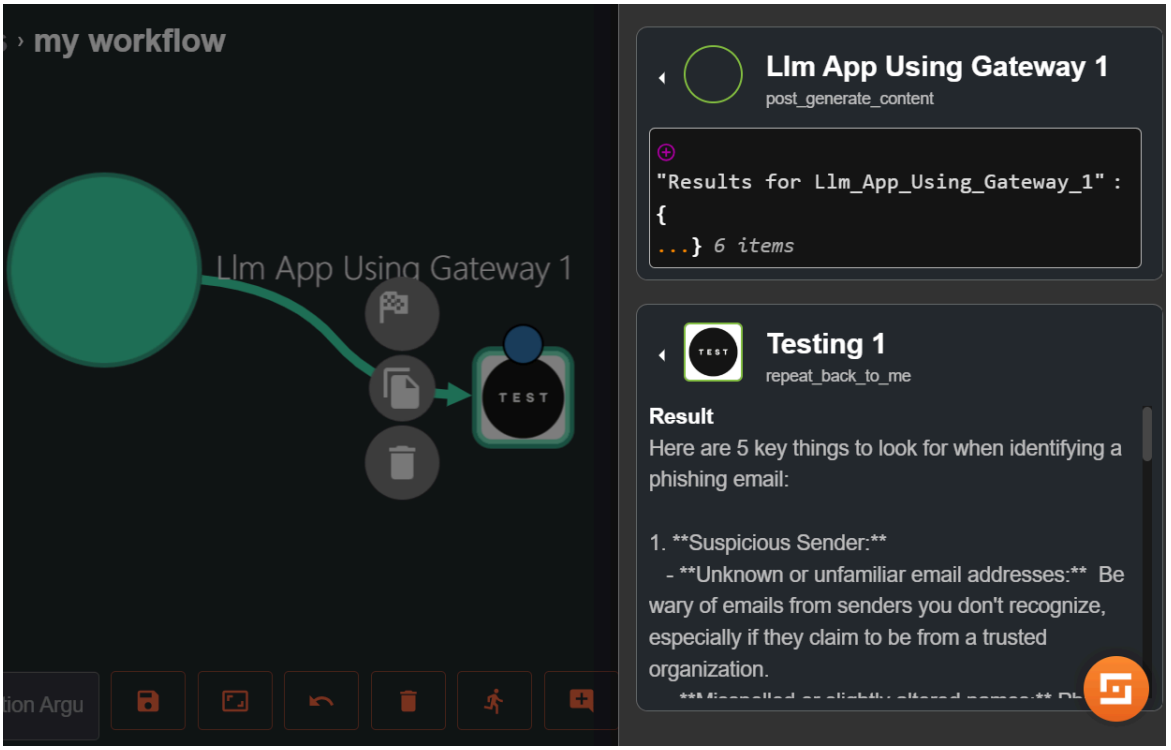

teway 1
TEST
Testing
Testing
Name
Delay
Testing_1
Find Actions
Repeat back to me
Parameters
Call
$llm_app_using_gateway_1.body.candidates.#.content.parts.#.text


teway 1
TEST
Testing
Testing
Name
Delay
Testing_1
Find Actions
Repeat back to me
Parameters
Call
$llm_app_using_gateway_1.body.candidates.#0.content.parts.#0.text

11. Click the play button in the bottom left to execute your workflow

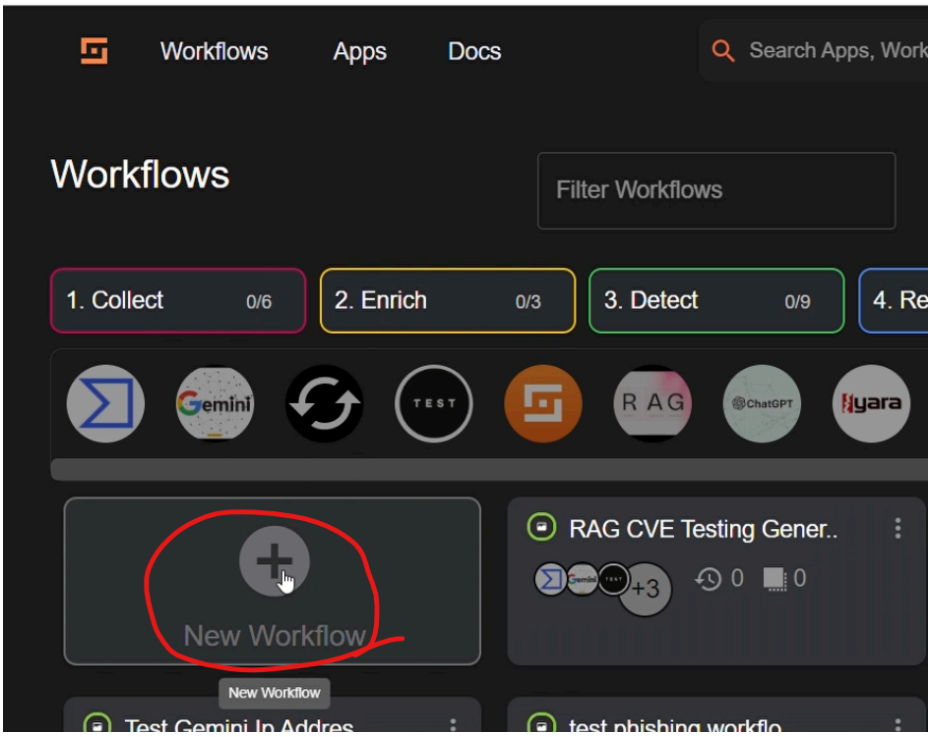


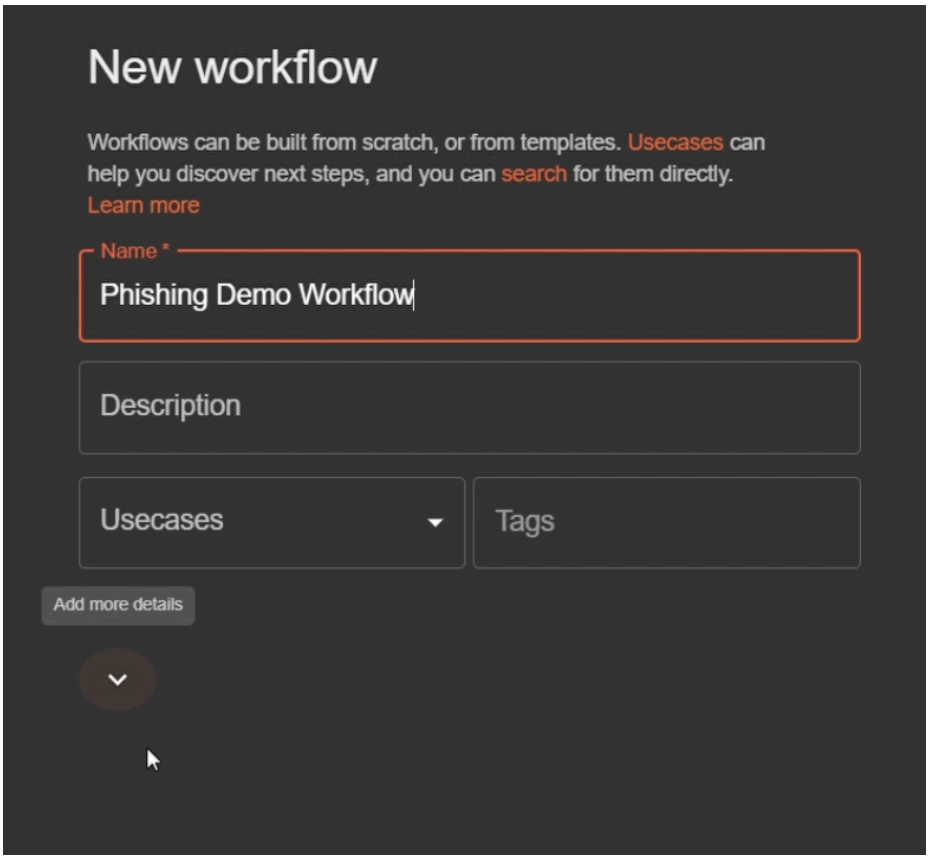
12. The window on the right will show the response from each app in the workflow. The first response is from the app you created, and the second response is from the 'Testing' app



Phishing Workflow

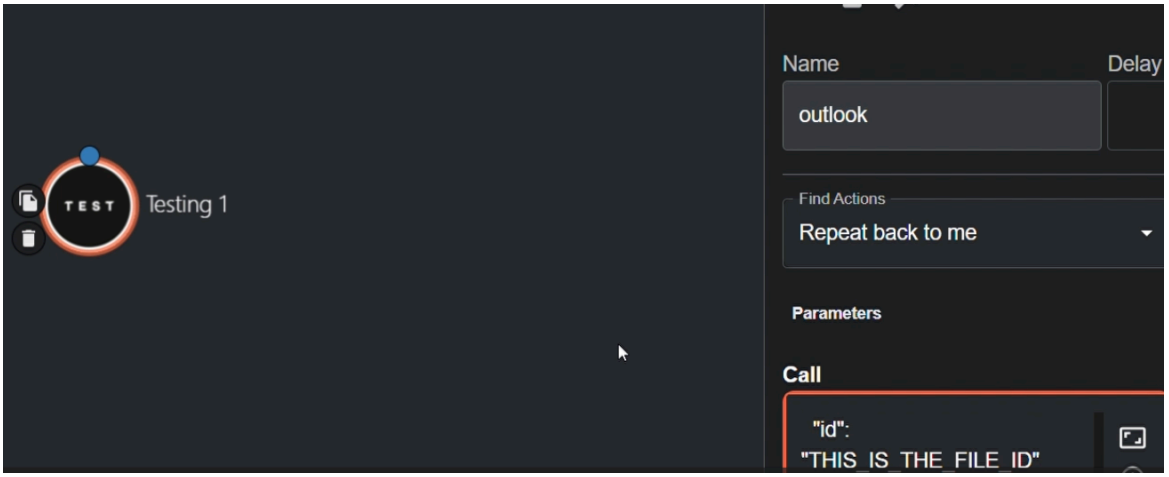
Phishing workflow can easily be updated to use Outlook by minor modifications in path or name of fields.





Create workflow

1. Add a mock Outlook app, as we haven't set up the required credentials to use Outlook with our organization yet. You can build this workflow with the actual Outlook app, making minor modifications to path names. Your test app should look like this (Body to copy is below)

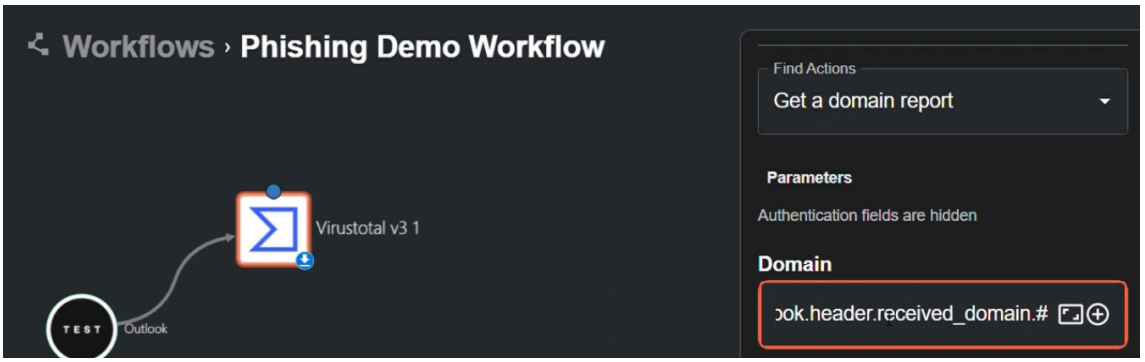



```
{
  "header": {
    "from": "jane.doe@example.com",
    "to": [
      "john.smith@example.com",
      "alice.jones@example.com"
    ],
    "subject": "Meeting Reminder",
    "date": "2024-07-25T09:15:00Z",
    "received_ip": [
      "192.168.1.1"
    ],
    "received_domain": [
      "example.com"
    ]
  },
  "body": {
    "content": "This is a dummy email content with a URL: https://www.exampl1e.com/resource",
    "domain": [
      "exampl1e.com"
    ],
    "uri": [
      "https://www.exampl1e.com/resource"
    ]
  },
  "attachments": [
    ""
  ],
  "id": "THIS_IS_THE_FILE_ID"
}
```

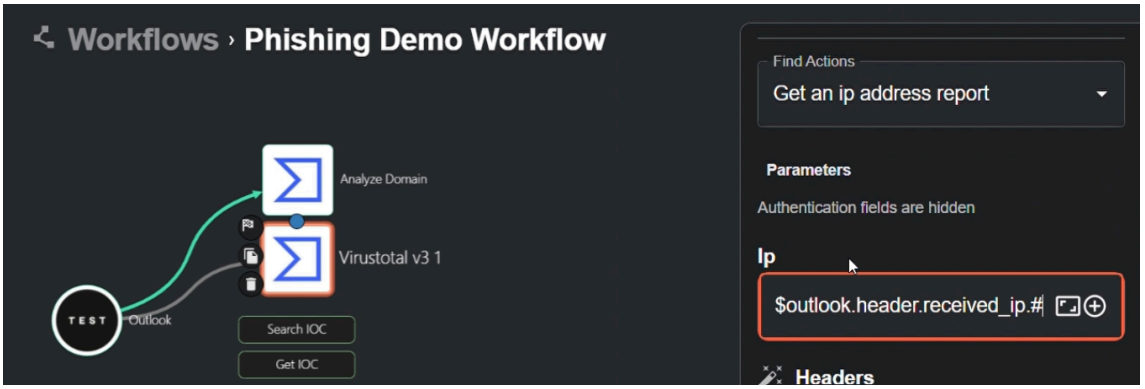
2. Next we'll use VirusTotal to retrieve information regarding the sender IP address and email domain and any urls provided in the email body. First we'll need a VirusTotal

API key from https://www.virustotal.com/gui/sign-in just make a free account to get one.

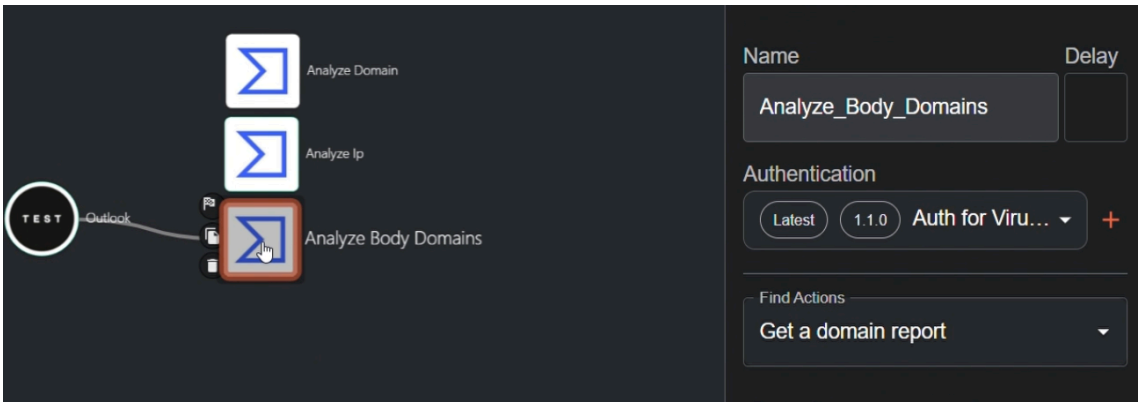
3. Connect VirusTotal to get a report on the sender's domain

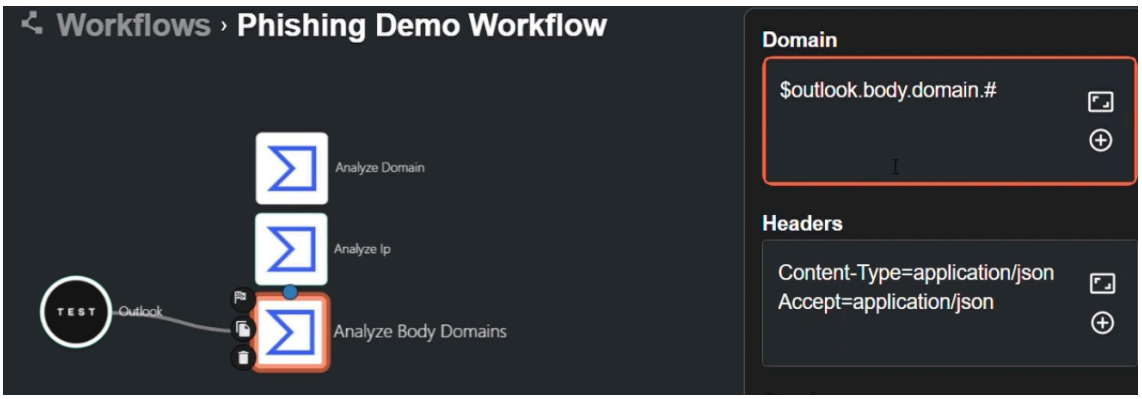


4. Connect another VirusTotal app to get a report on the sender's IP address

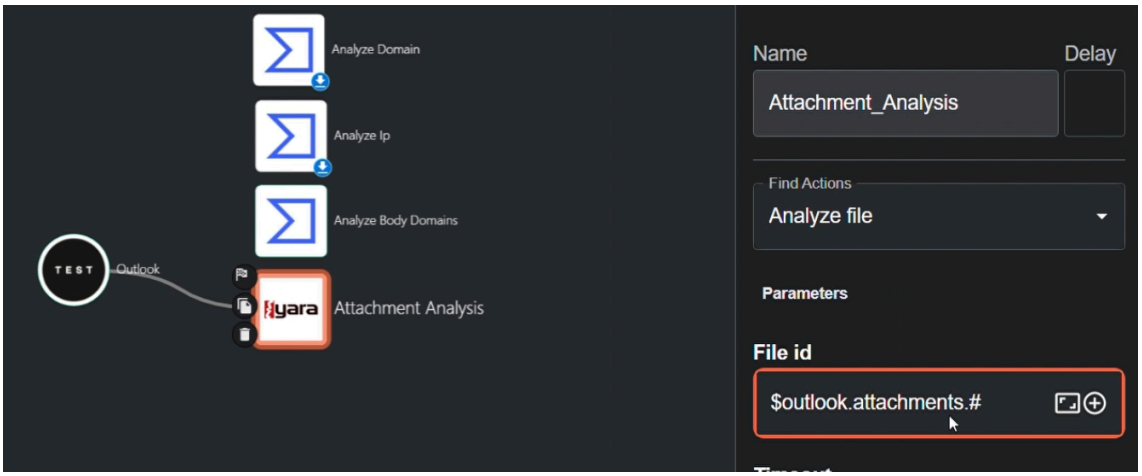


5. Connect a final VirusTotal app to get a report on any URLs in the email body

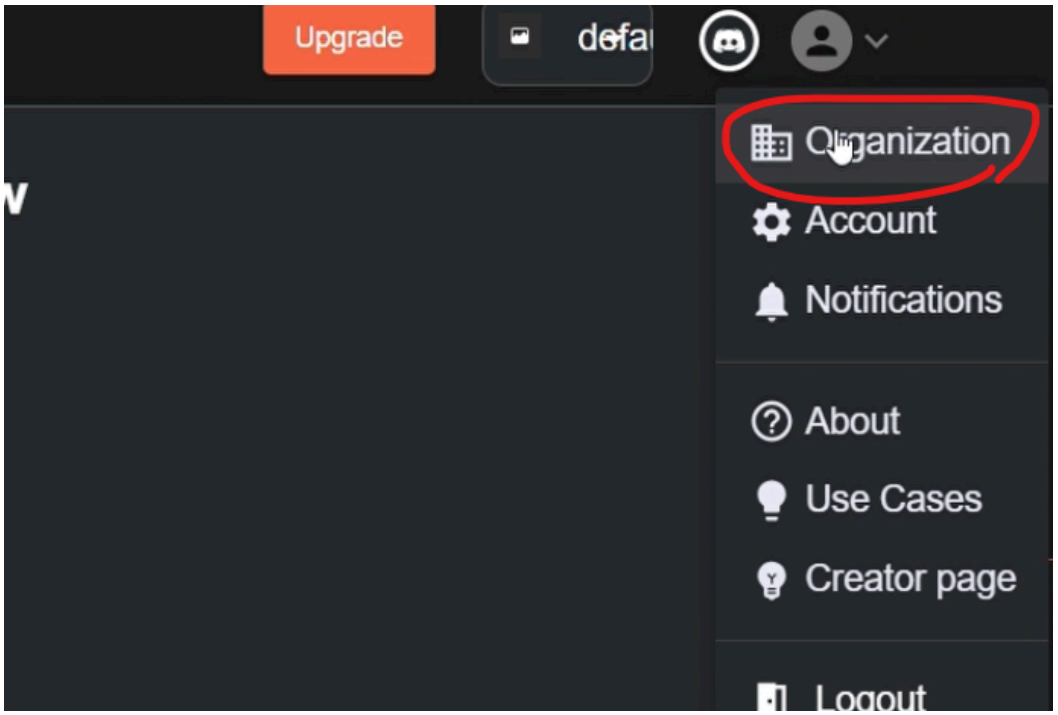




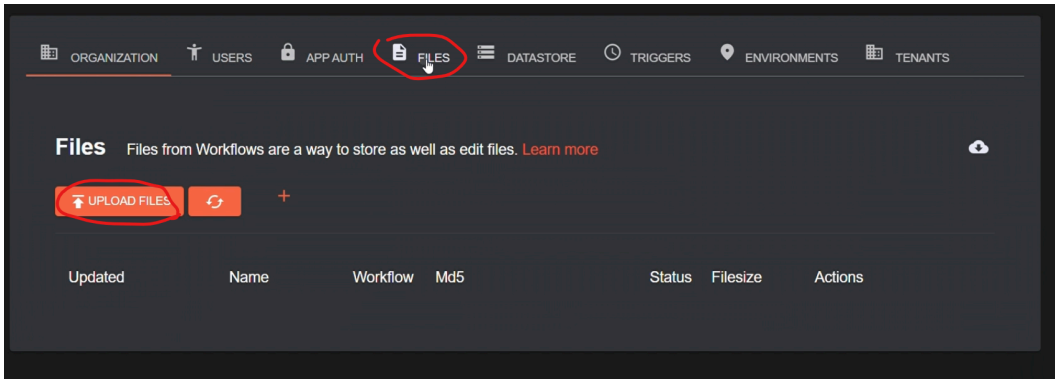
6. Connect Yara app to analyze any files referenced in the outlooks.attachment field



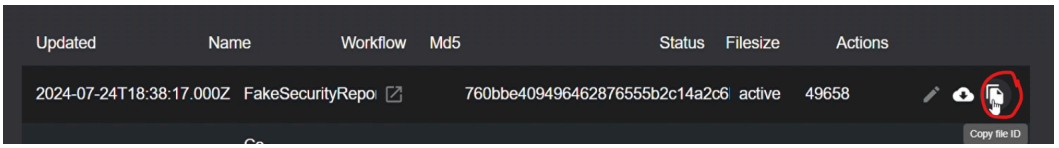
7. To add files to test Yara attachment analysis:
    a. Navigate to the 'Organization' tab



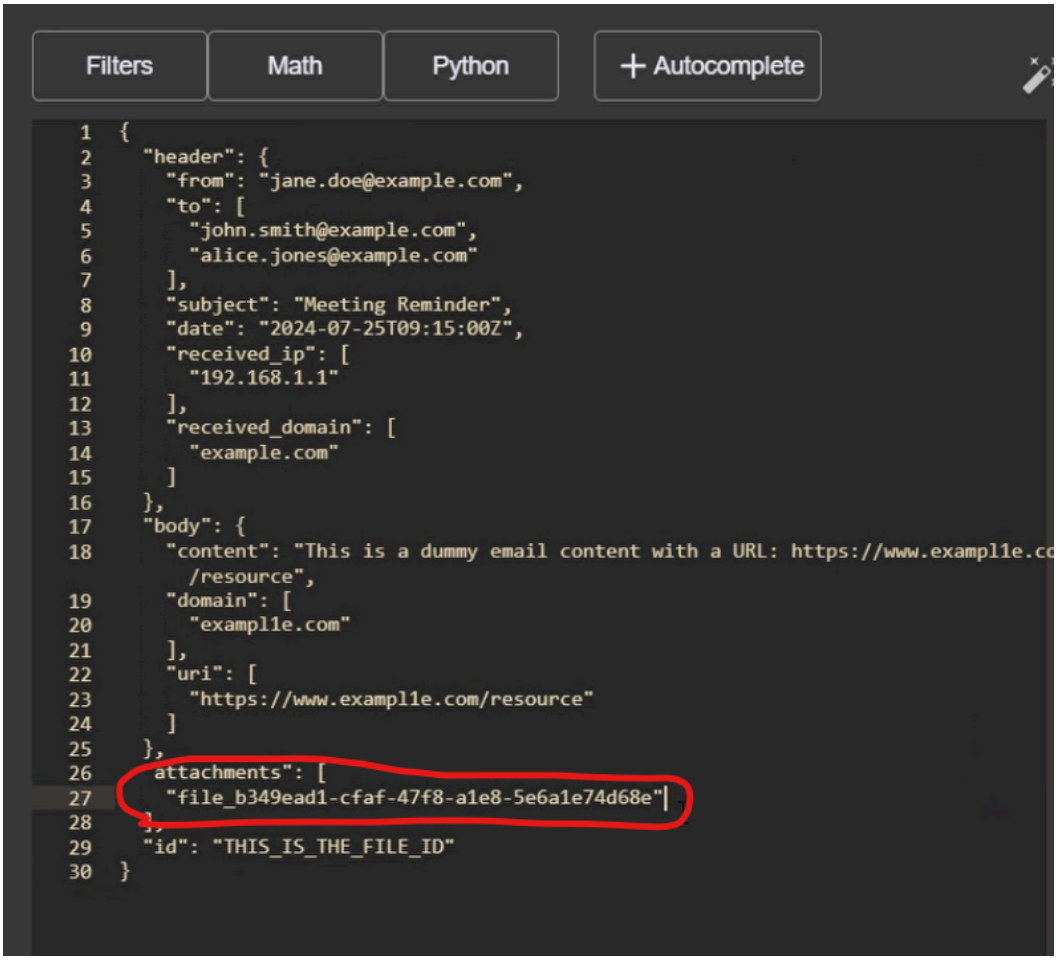
    b. Navigate to the 'Files' tab and select 'Upload Files'



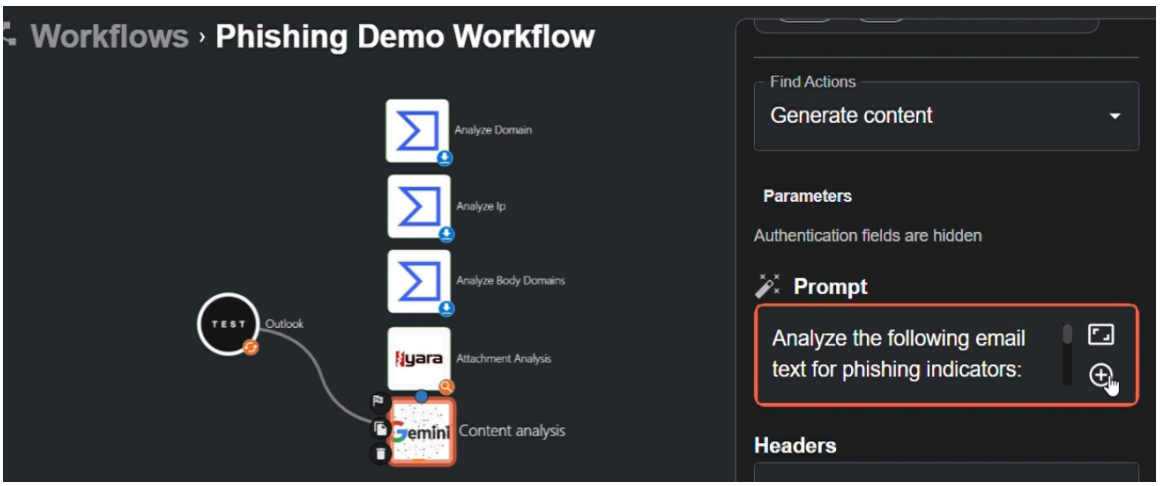
c. Copy the file ID



d. Go back to your workflow and update the mock response to use your file id



8. Connect the Gemini app to analyze the email header and body for phishing indicators. Return an enum indicating the safety level (SAFE, SUSPICIOUS, MALICIOUS)



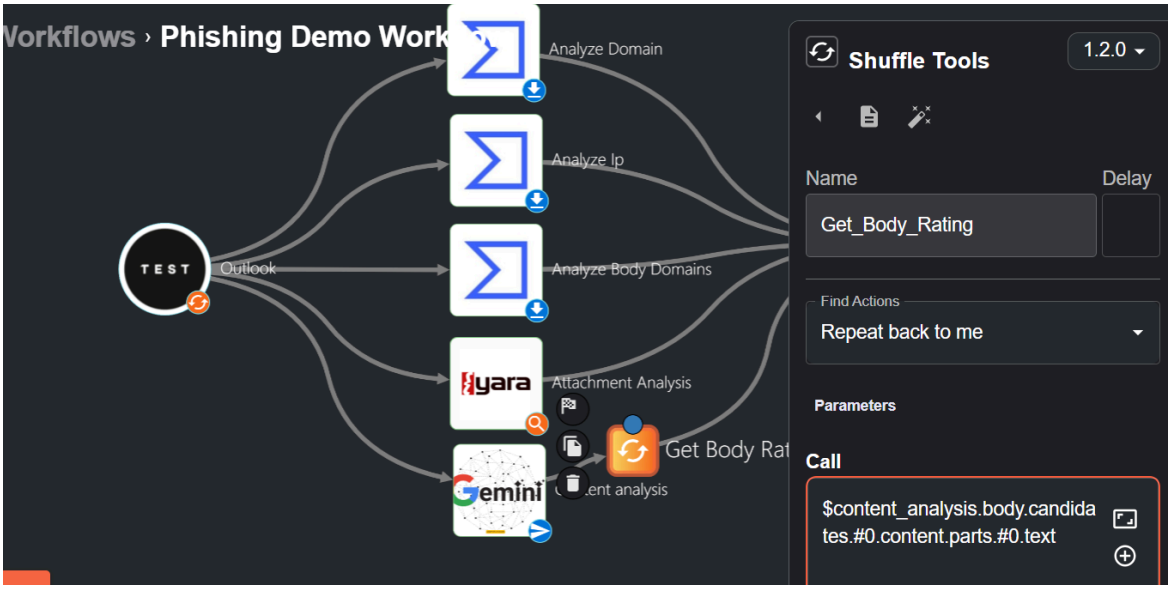
Connect a 'Shuffle Tools' app to the Gemini app to strip any spacing from the LLM

Analyze the following email text for phishing indicators:

$outlook.header.subject
$outlook.body.content

Analyze the provided email text for phishing indicators. Identify potential red flags such as urgent calls to action, generic greetings, requests for personal information and unusual language or grammar. As your response give one of the following enums and nothing else, you can tell what type of rating it is by the word: SAFE, SUSPICIOUS, MALICIOUS

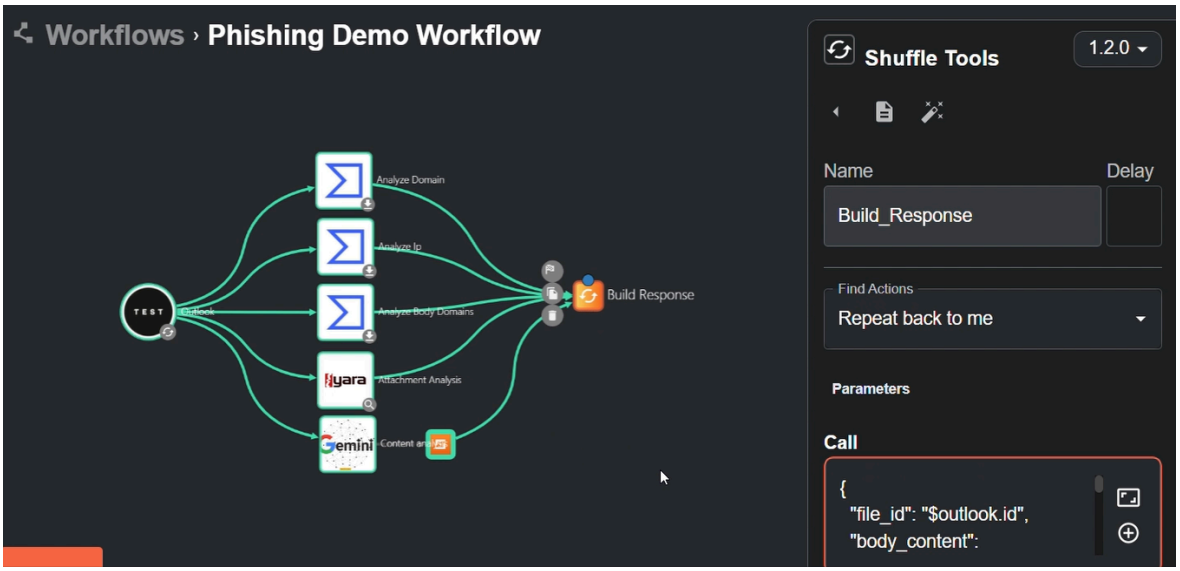
9. response



"{{ "$analyze_body_content.body.candidates.#0.content.parts.#0.text" | strip }}"

10. Now we've created all the apps that we're going to use to help detect if the email is a Phishing email. Now with all the data we're receiving from the current apps, we're going to build an object to store the information we find valuable for Phishing detection. So we'll connect all the apps we just made to the 'Shuffle Tools' app to

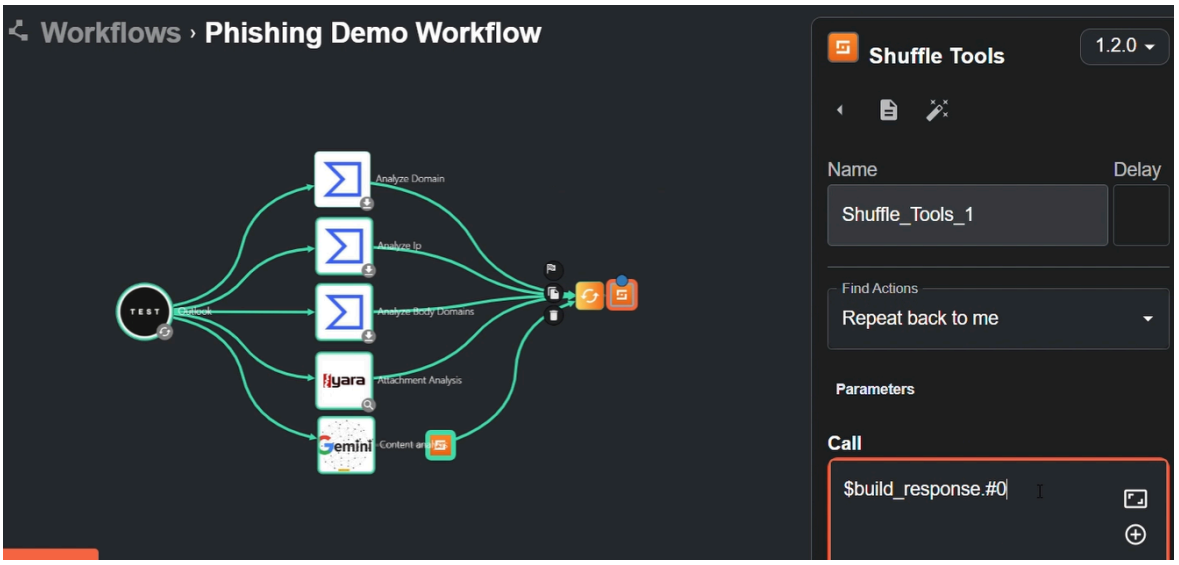
merge the fields we'll use into one json object




```
{
  "file_id": "$outlook.id",
  "body_content": $get_body_rating,
  "attachments": [
    {
      "failed_rules": $attachment_analysis.#.failed_rules,
      "total_failed_rules": $attachment_analysis.#.total_rule_files,
      "total_string_matches": $attachment_analysis.#.total_string_matches
    }
  ],
  "domains": [
    {
      "votes": {
        "harmless": $doman_report.#.body.data.attributes.total_votes.harmless,
        "malicious": $doman_report.#.body.data.attributes.total_votes.malicious
      },
      "analysis_stats": {
        "malicious": $doman_report.#.body.data.attributes.last_analysis_stats.malicious,
        "suspicious": $doman_report.#.body.data.attributes.last_analysis_stats.suspicious,
        "harmless": $doman_report.#.body.data.attributes.last_analysis_stats.harmless
      }
    },
    {
      "votes": {
        "harmless": $body_domain_report.#.body.data.attributes.total_votes.harmless,
        "malicious": $body_domain_report.#.body.data.attributes.total_votes.malicious
      },
      "analysis_stats": {
        "malicious": $body_domain_report.#.body.data.attributes.last_analysis_stats.malicious,
        "suspicious": $body_domain_report.#.body.data.attributes.last_analysis_stats.suspicious,
        "harmless": $body_domain_report.#.body.data.attributes.last_analysis_stats.harmless
      }
    }
  ],
  "sender_ip": {
    "votes": {
```

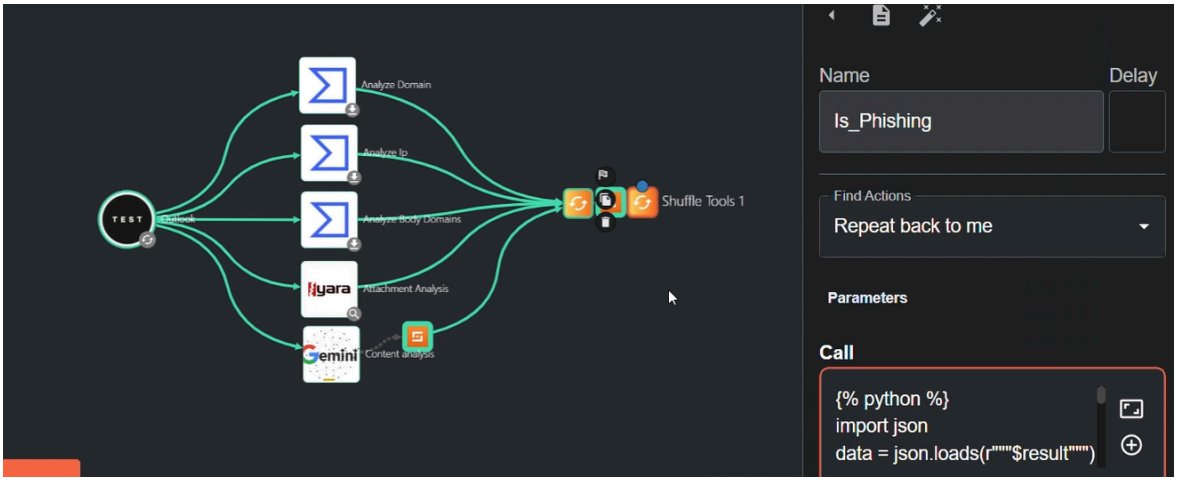
```
    "harmless": $ip_address_report.#0.body.data.attributes.total_votes.harmless,
    "malicious": $ip_address_report.#0.body.data.attributes.total_votes.malicious
   },
   "analysis_stats": {
    "malicious": $ip_address_report.#0.body.data.attributes.last_analysis_stats.malicious,
    "suspicious": $ip_address_report.#0.body.data.attributes.last_analysis_stats.suspicious,
    "harmless": $ip_address_report.#0.body.data.attributes.last_analysis_stats.harmless
   }
  }
}
```

11. Connect a new 'Shuffle Tools' app to choose the first index of the built response since it gets built as an array with only one entry



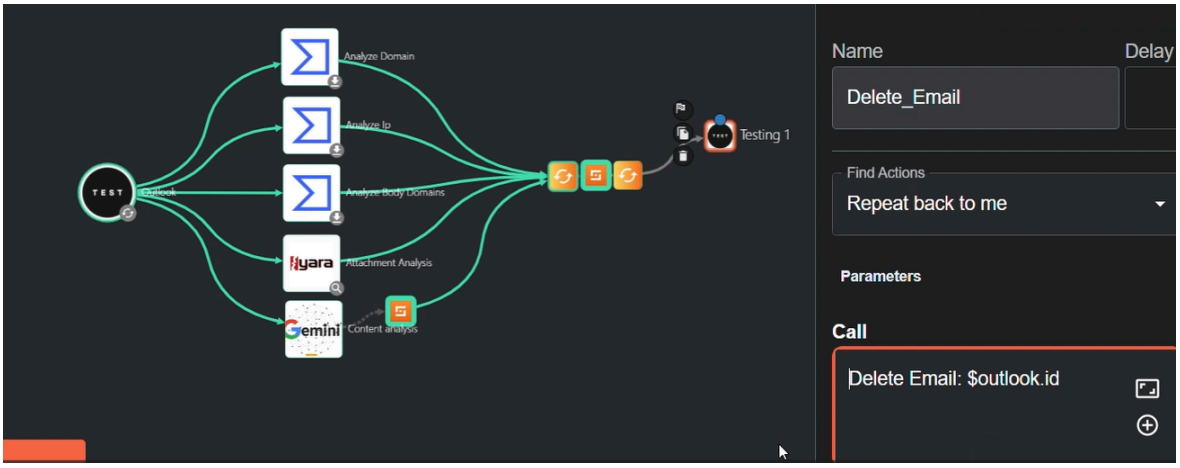

```
$build_result.#0
```

12. Connect 'Shuffle Tools' app again to execute some Python and check the saved fields to see whether the app has any indicators of a Phishing email

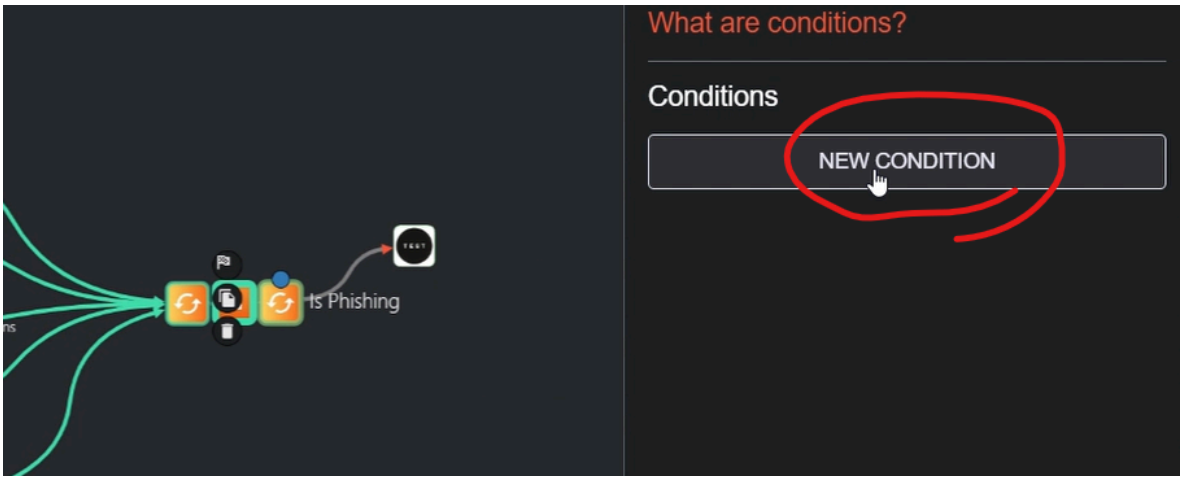


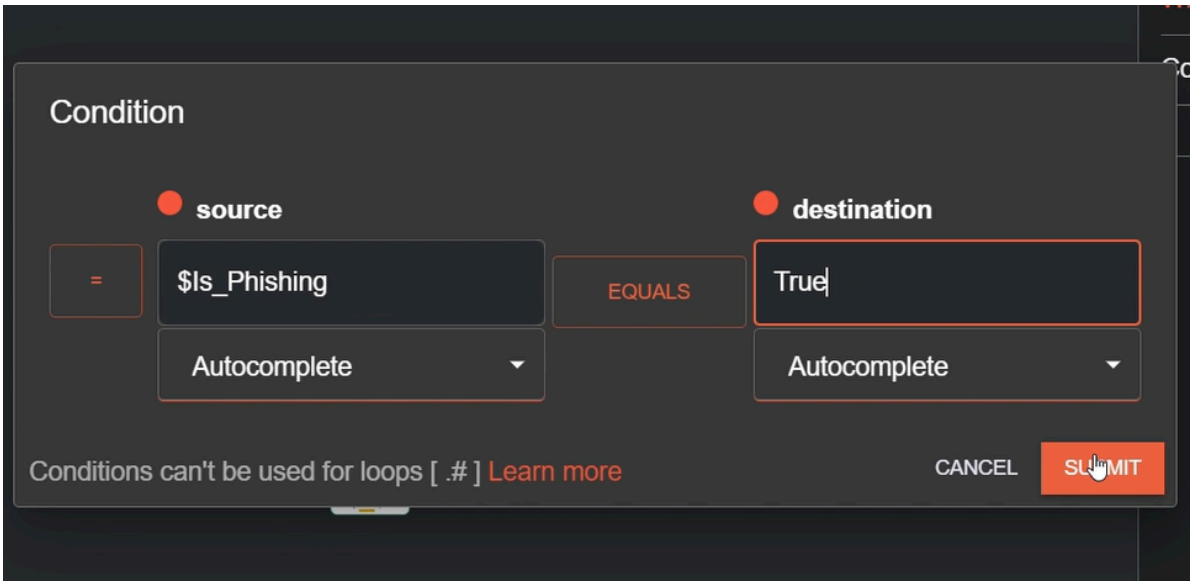
```
{% python %}
import json
data = json.loads(r"""$result""")

is_phishing_email = False

# below checks are examples, they can be adjusted for your use case
if data["body_content"] == "MALICIOUS":
    is_phishing_email = True

for attachment in data["attachments"]:
    if isinstance(attachment.get("failed_rules"), int) and attachment["failed_rules"] > 50:
        is_phishing_email = True
        break

if len(data["domains"]) > 0:
  for domain in data["domains"]:
      if isinstance(domain["votes"].get("malicious"), int) and domain["votes"]["malicious"] > 15:
          is_phishing_email = True
          break
      if isinstance(domain["analysis_stats"].get("malicious"), int) and
domain["analysis_stats"]["malicious"] > 0:
          is_phishing_email = True
          break

sender_ip = data["sender_ip"]
if sender_ip["votes"]["malicious"] > 0:
    is_phishing_email = True
if sender_ip["analysis_stats"]["malicious"] > 0:
    is_phishing_email = True


print(is_phishing_email)

{% endpython %}
```

13. Once Phishing analysis is complete, we can now add our app to delete the email if it was detected as Phishing. In our case since we don't have the Outlook app authenticated yet we're just going to use a testing app to mock deleting the email



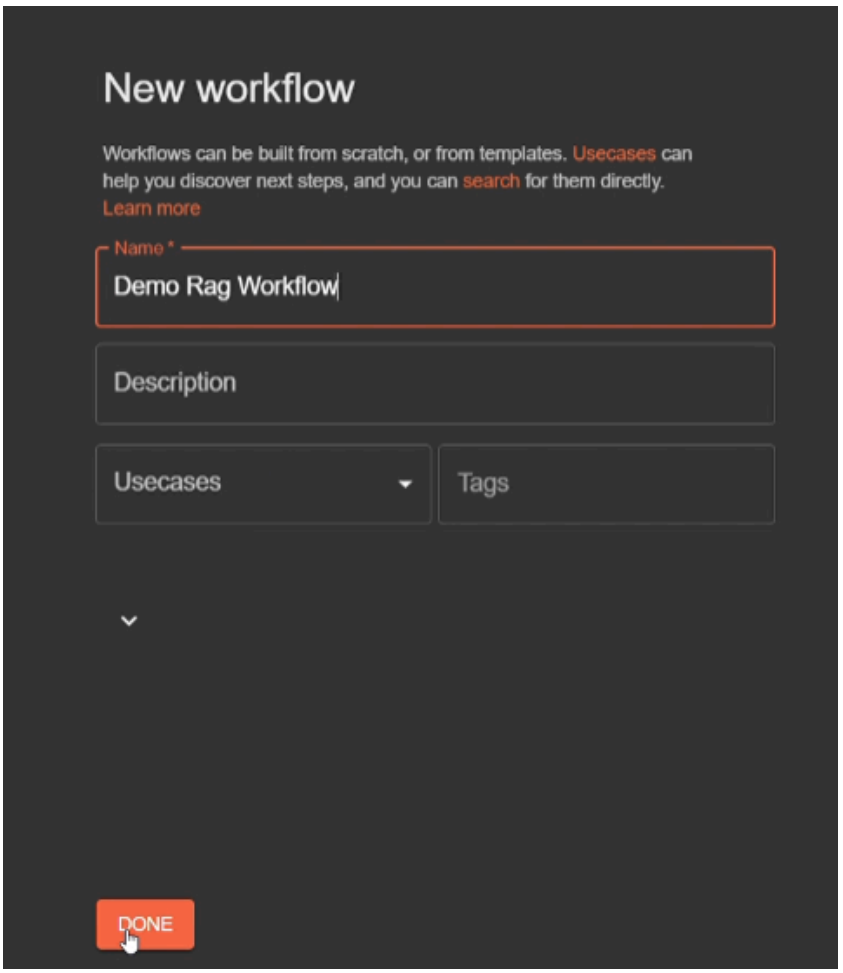
14. Finally we'll add a conditional branch to the 'Delete Email' app to check that $Is_Phishing = true. This condition will allow us to only delete the email if it looks malicious, otherwise we'll do nothing





RAG CVE Workflow

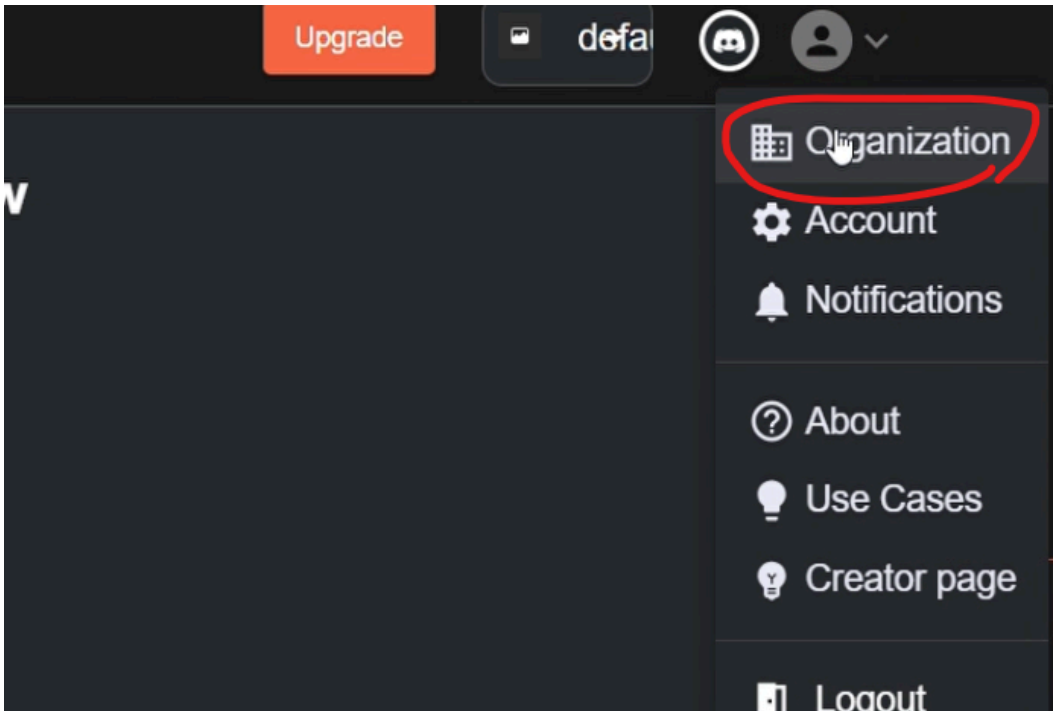
1. Create workflow



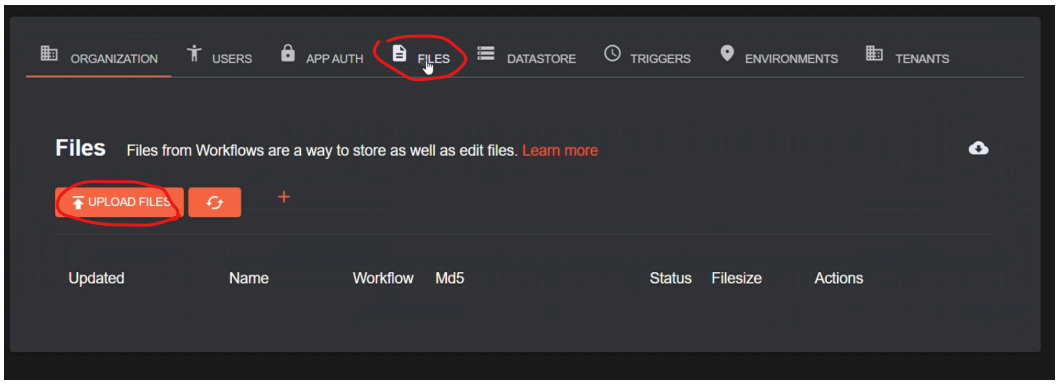
2. We will start with the 'Shuffle AI' app since it has the 'Extract text from PDF' action, which will read from the pdf

3. If you don’t already have a pdf file uploaded to scan for CVEs, follow these steps to add one. If you already have a file skip to step 4.
    a. Create a file using Google Docs and include the CVEs listed below. You can write your own content or use ChatGPT to generate it
    [CVE-2012-2122, CVE-2024-6387p, CVE-2016-2851, CVE-2022-37706]

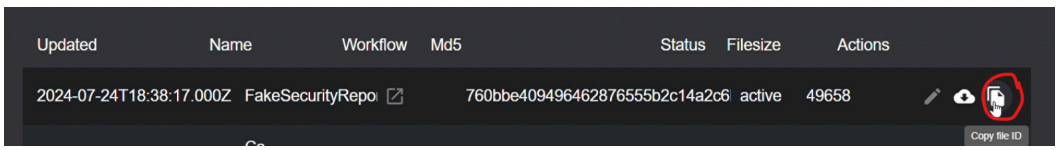
b. Navigate to the 'Organization' tab



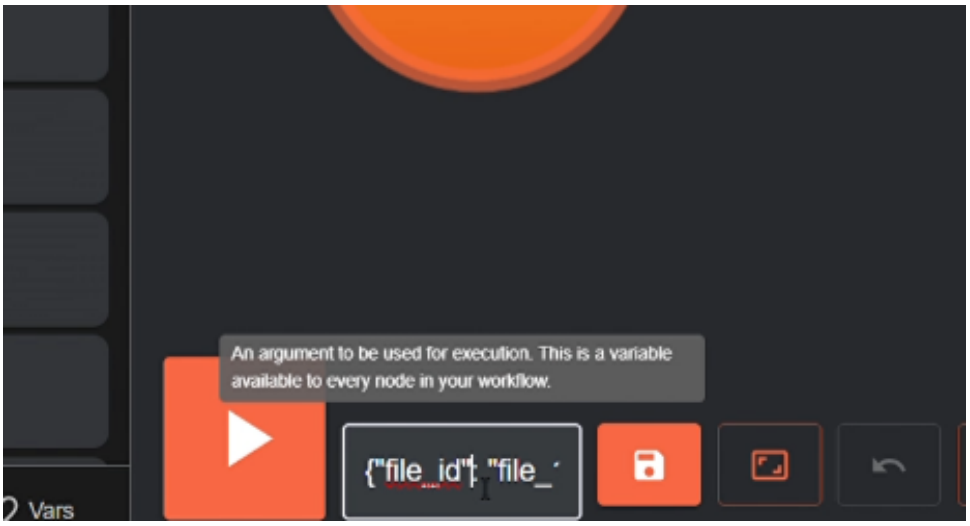
c. Navigate to the 'Files' tab and then select 'Upload Files'



d. Copy the file ID

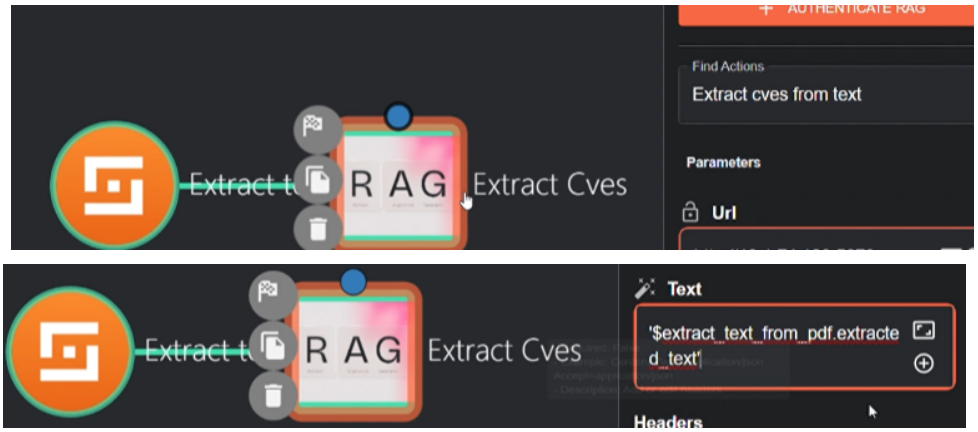


4. With the uploaded file ID, add it to your execution argument

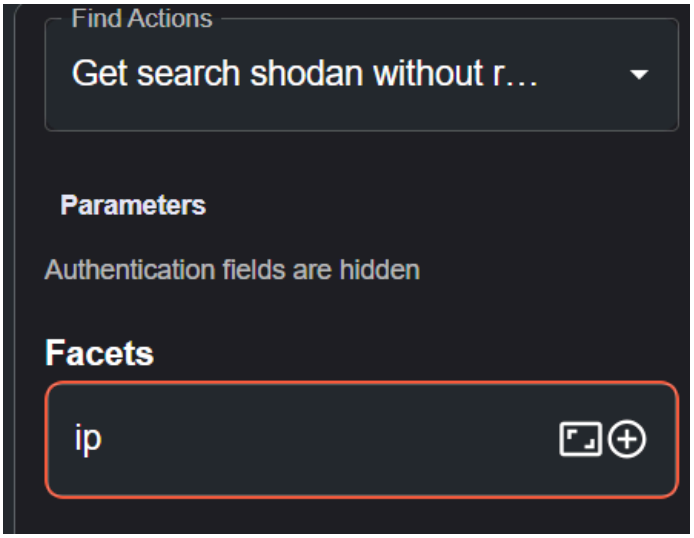


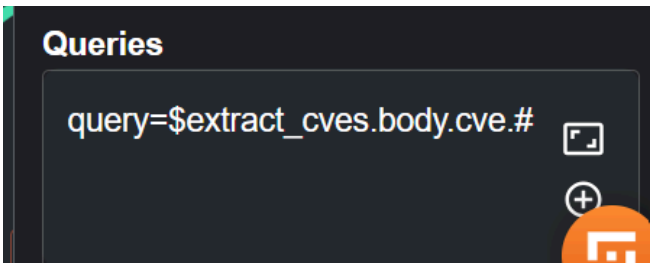

```
{"file_id": "YOUR_FILE_ID_HERE"}
```

5. Add the RAG (Retrieval Augmented Generation) LLM app to extract the CVEs from the text read from the PDF

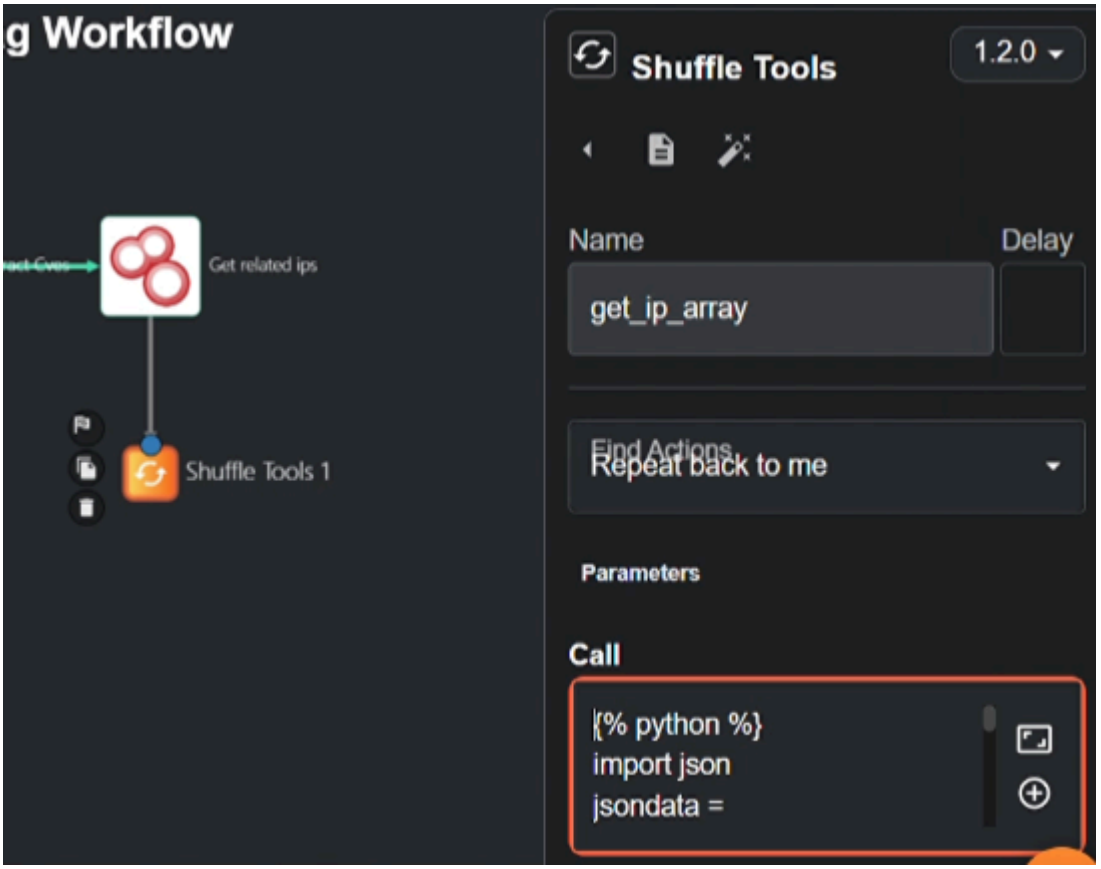


6. Connect the Shodan app to the RAG app to retrieve the IP addresses related to the extracted CVEs



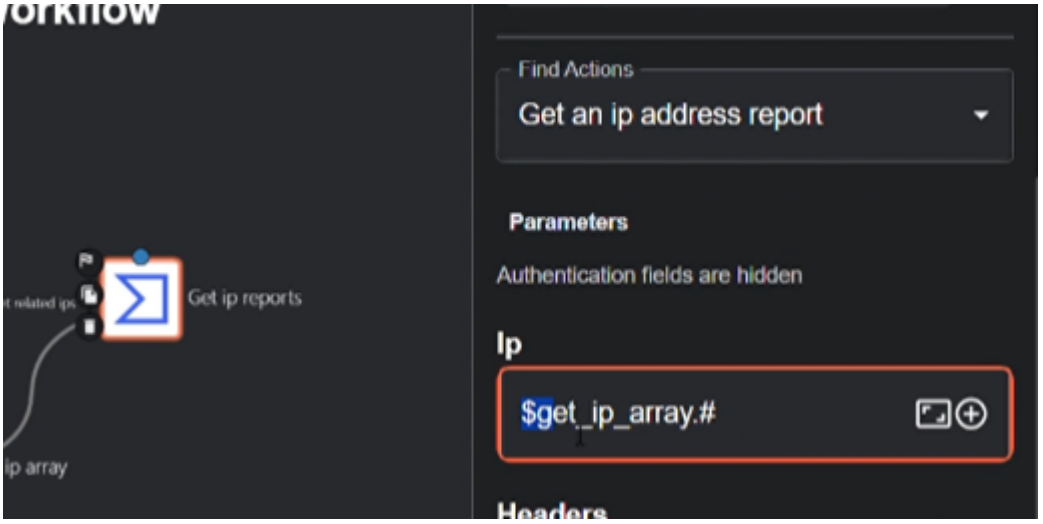


7. Use the 'Shuffle Tools' app to execute a Python script that loops through the responses from Shodan and creates an array of IP addresses found

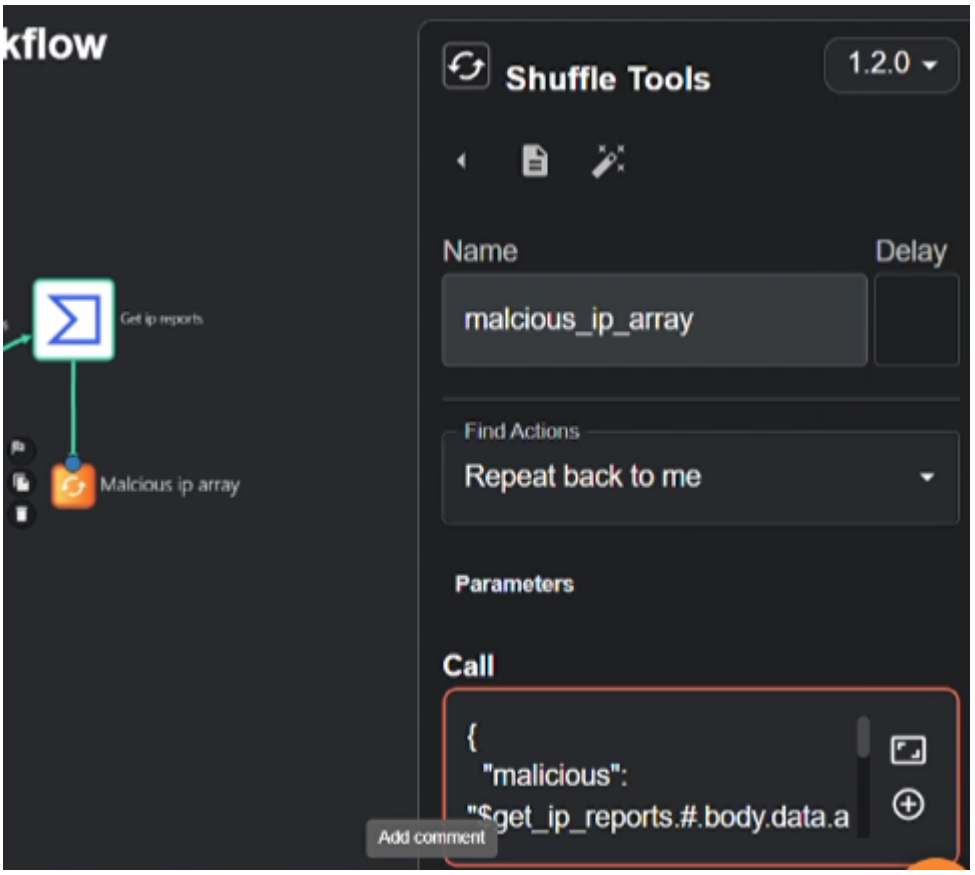



```
{% python %}
import json
jsondata = json.loads(r"""$get_related_ips""")

res_ips = set()
for response in jsondata:
  ips = response['body']['facets']['ip']
  for ip in ips:
    res_ips.add(ip['value'])

print(json.dumps(list(res_ips)))

{% endpython %}
```

8. Connect VirusTotal to retrieve reports on the IP address array to check if any addresses are considered malicious

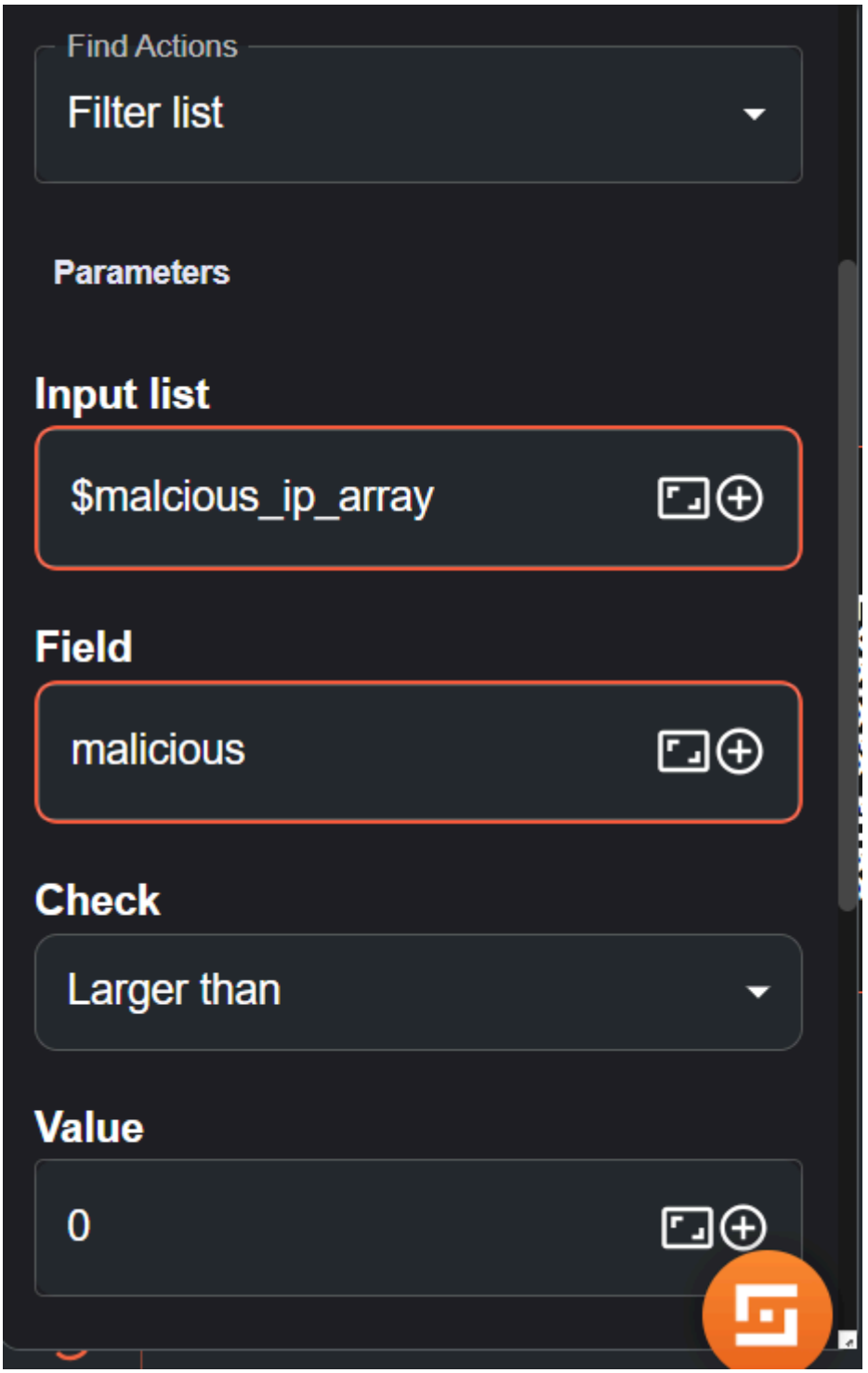


9. Convert each VirusTotal response object to a smaller object containing only the IP and the last analysis malicious count

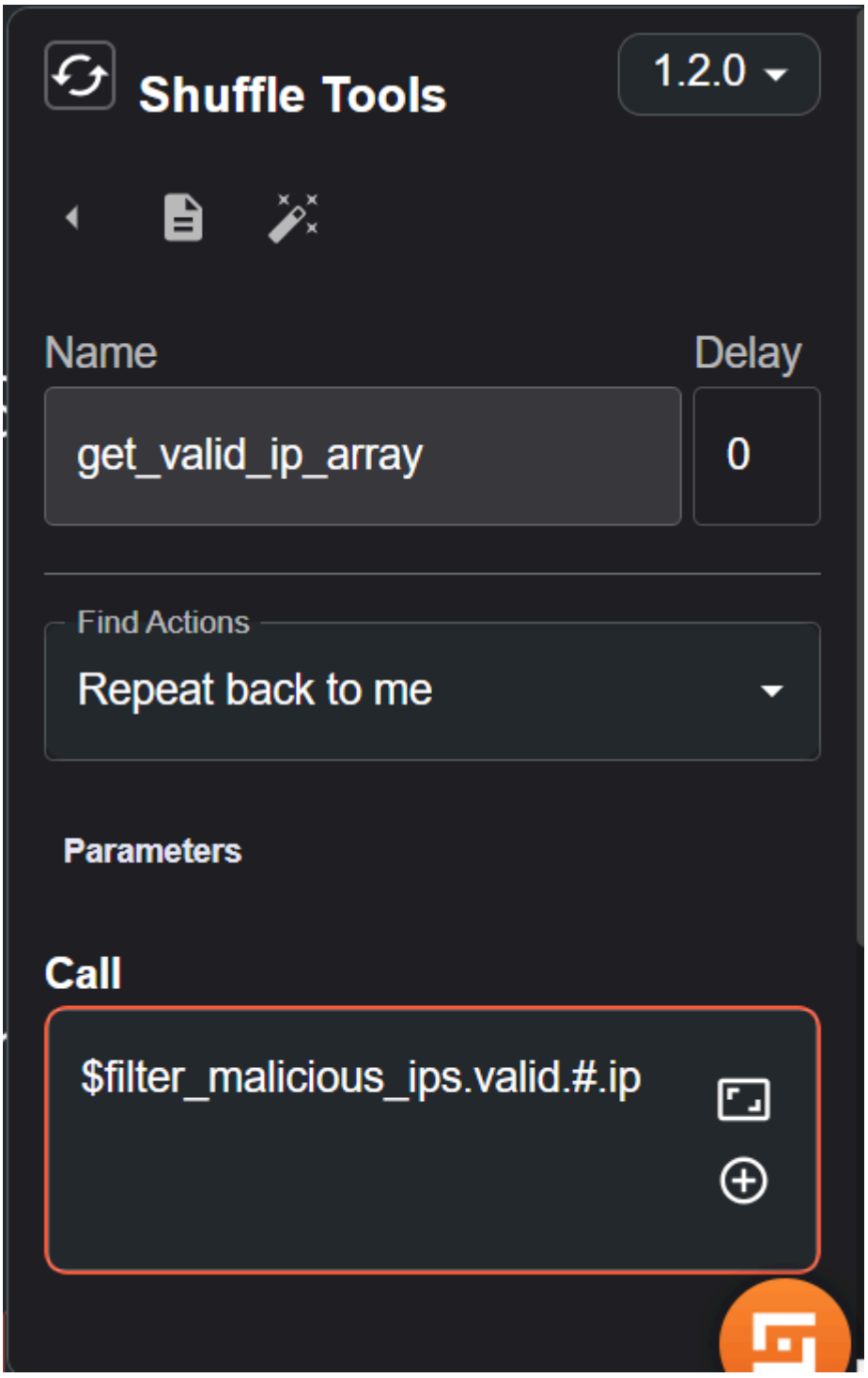



```
{
  "malicious": "$get_ip_reports.#.body.data.attributes.last_analysis_stats.malicious",
  "ip": "$get_ip_reports.#.body.data.id"
}
```

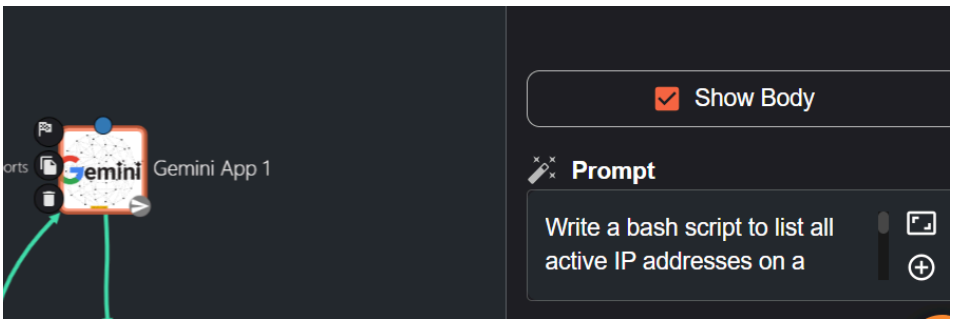
10. Filter the objects to keep only those with a malicious count greater than 0



11. Create an array of the remaining IP addresses that were found to be malicious

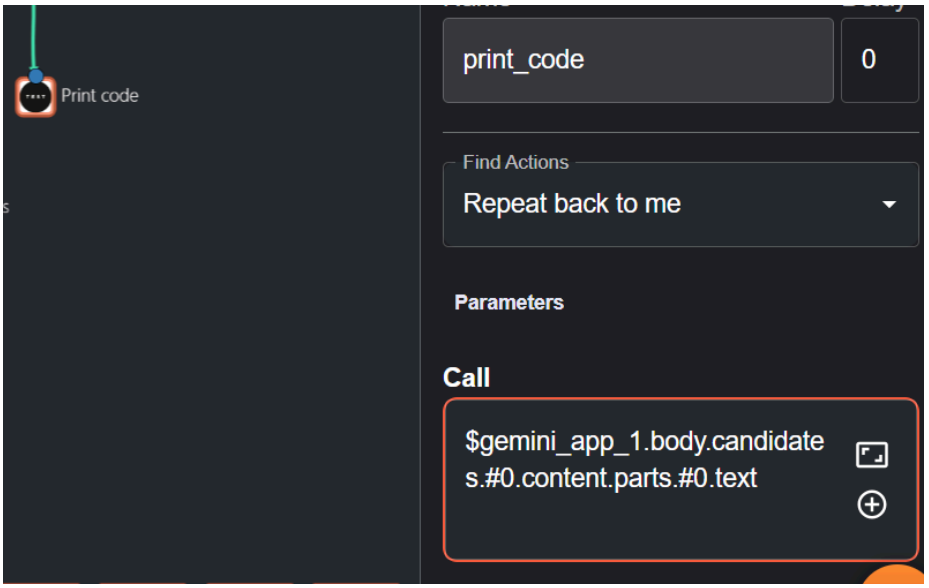


12. Use the Gemini app from our previous tutorial to prompt the LLM to generate code to scan your network for the related IP addresses

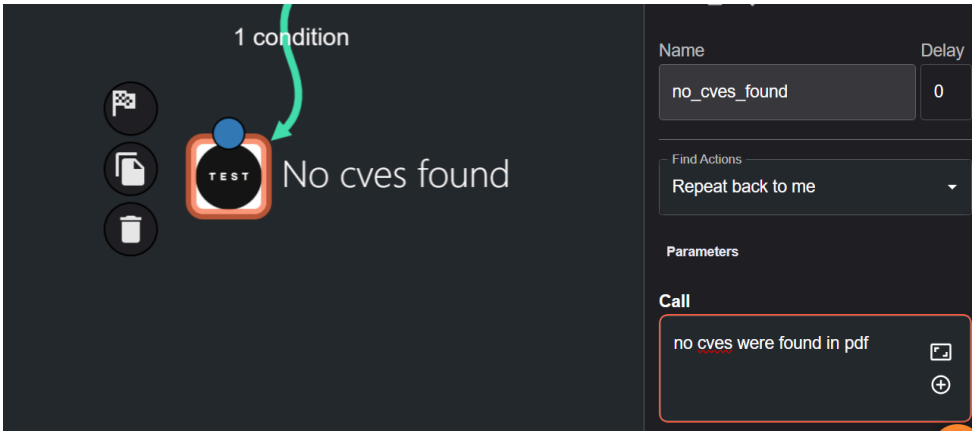


13. Update the prompt body to use single quotes around the ${prompt} variable in the raw body to prevent issues with formatting

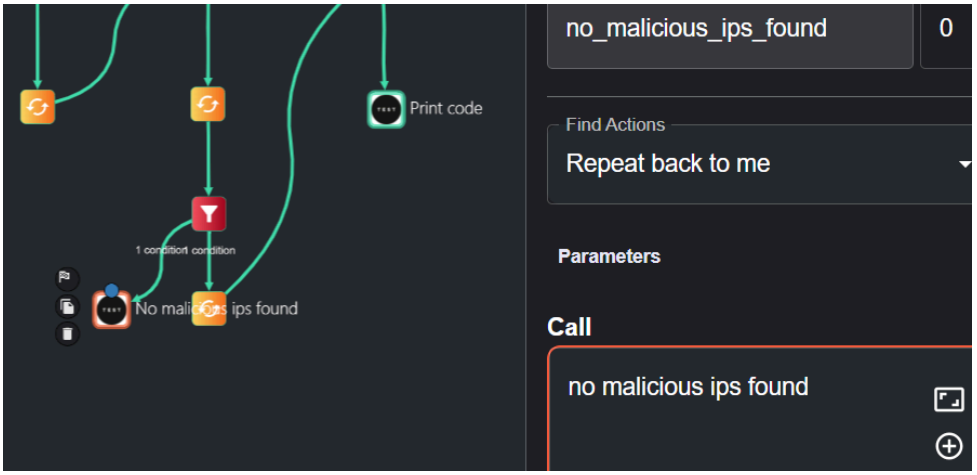
14. Add a test app to print out the response from Gemini



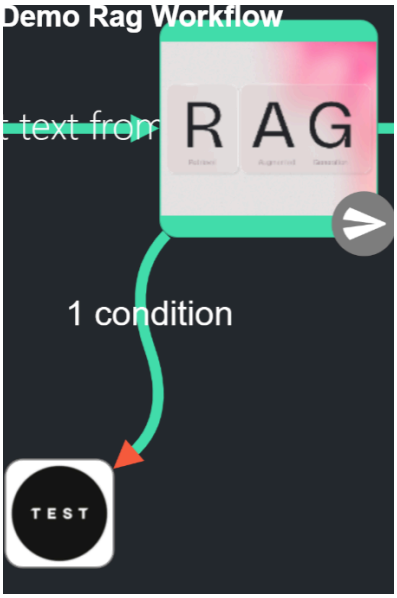
15. Add apps that will be called if execution is executed early. Add a testing app that executes if no CVEs are found in the text

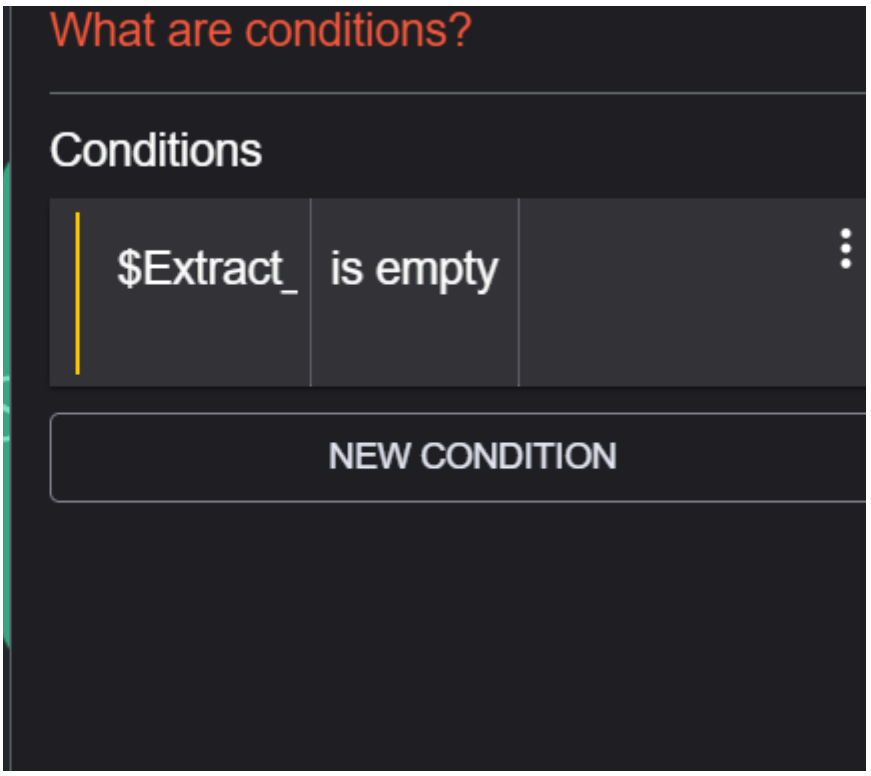


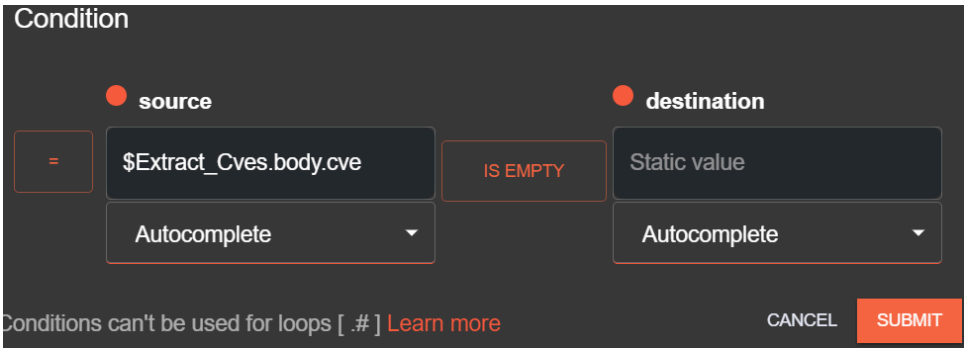
16. Add another testing app for cases where there are no malicious IP addresses related to the CVEs



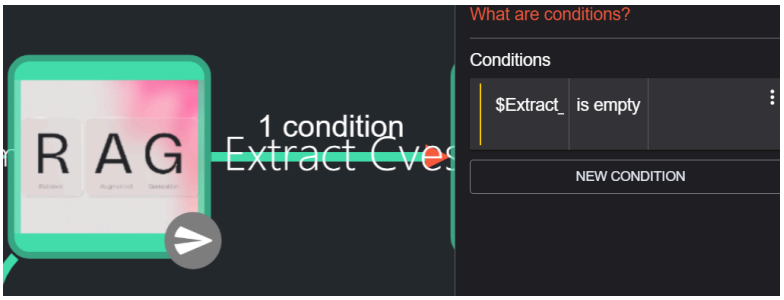
Finally we'll add some conditional branches (like if statements in programming). These will allow the workflow to change execution paths depending on different conditions.. So the first one we'll add is to the branch connecting our RAG to the 'no cves found' app



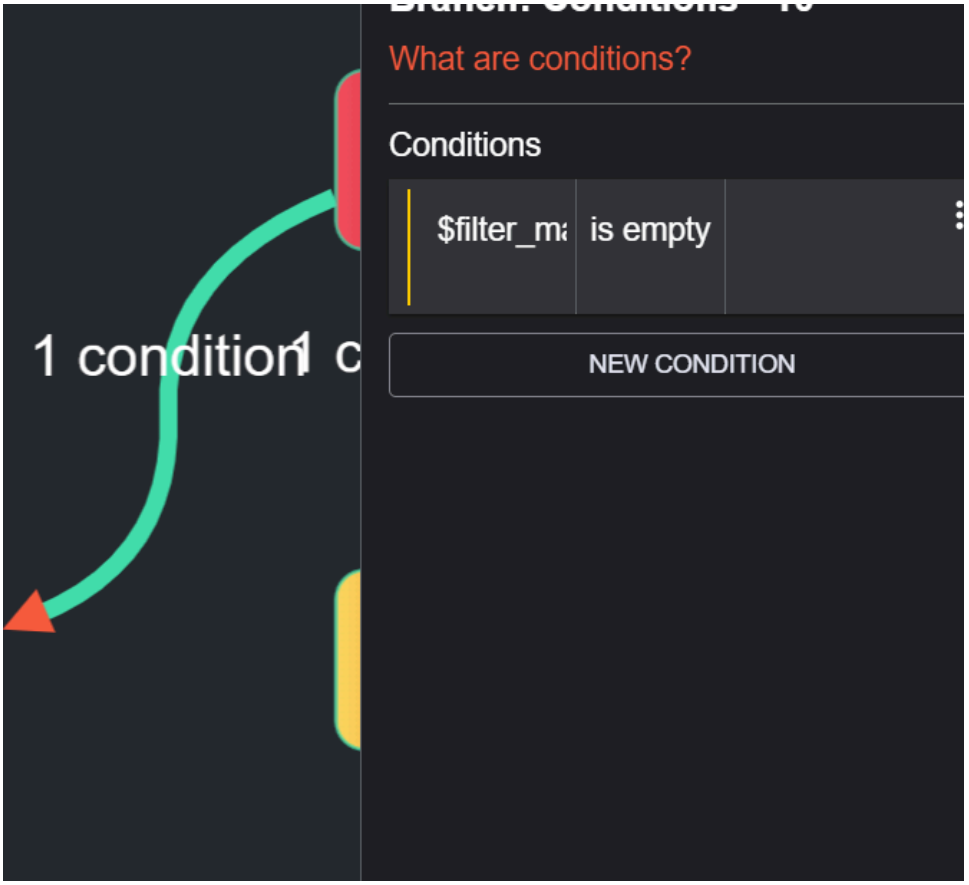


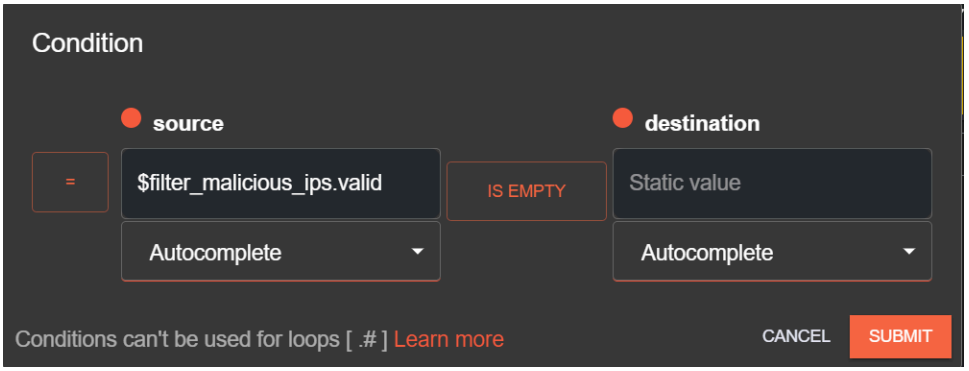


17. Add a condition to the branch connecting the RAG to Shodan to continue execution only if the list of CVEs from the extracted text is not empty

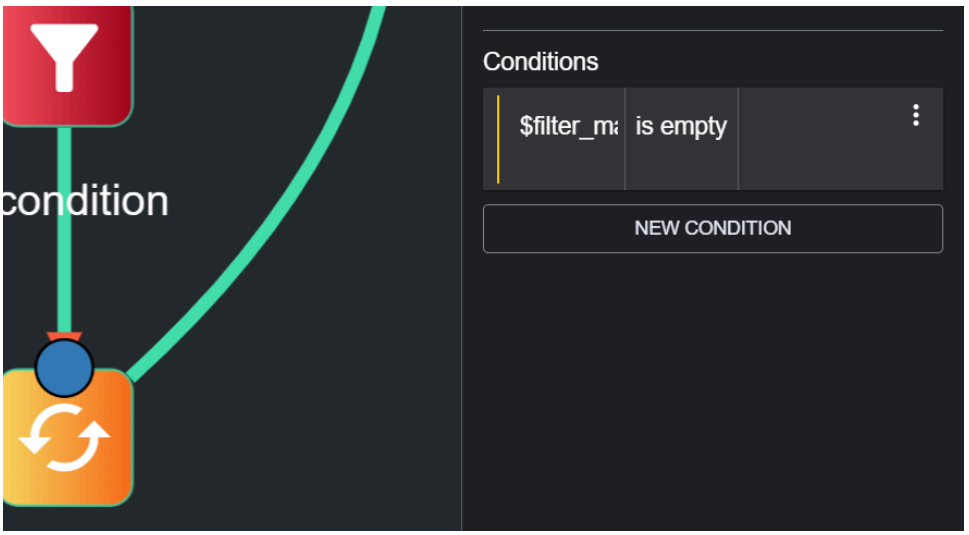


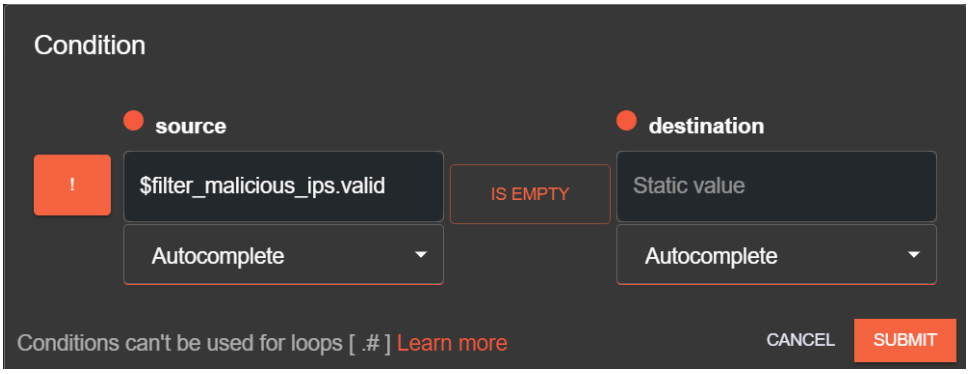
18. Add a condition to continue execution from filter app to the 'no malicious IPs found' app if the array of malicious IP addresses is empty



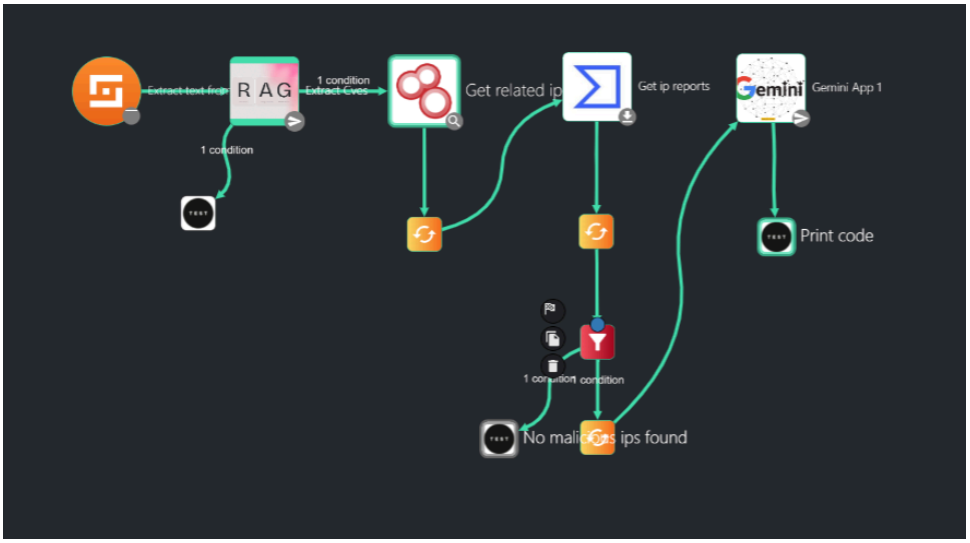


19. Add a condition to check if the array of malicious IP addresses is not empty and to continue the execution path to the 'get_valid_ip_array' app





20. Our workflow is now finished and should look something like this



University of Guelph

Canada Cyber Foundry

CyberScienceLab

Secure the future with us

cybersciencelab.com