DEFENDER MASTERCLASS LAB

Power Virtual Agents Integration with Microsoft Defender for Endpoint

ABSTRACT

This lab document contains a step-by-step guide to create a Power Virtual Agent that can get the alert and secure score status across multiple customers, and then respond with these scores into Teams.

Jack Lewis

Written for Microsoft Defender Masterclass Series – a series of events for Microsoft Partners created by James Graham

Getting Started

Please ensure you have completed the lab prerequisites:

aka.ms/defendermasterclass-repo

The getting started guide provides step by step instructions to create a demo tenant. You can then use this demo tenant to complete this lab – this tenant will be used as the "Partner" tenant for the remainder of the lab.

Create a Multi-Tenant Defender for Endpoint Bot, using Power Virtual Agents for Teams

Overview:

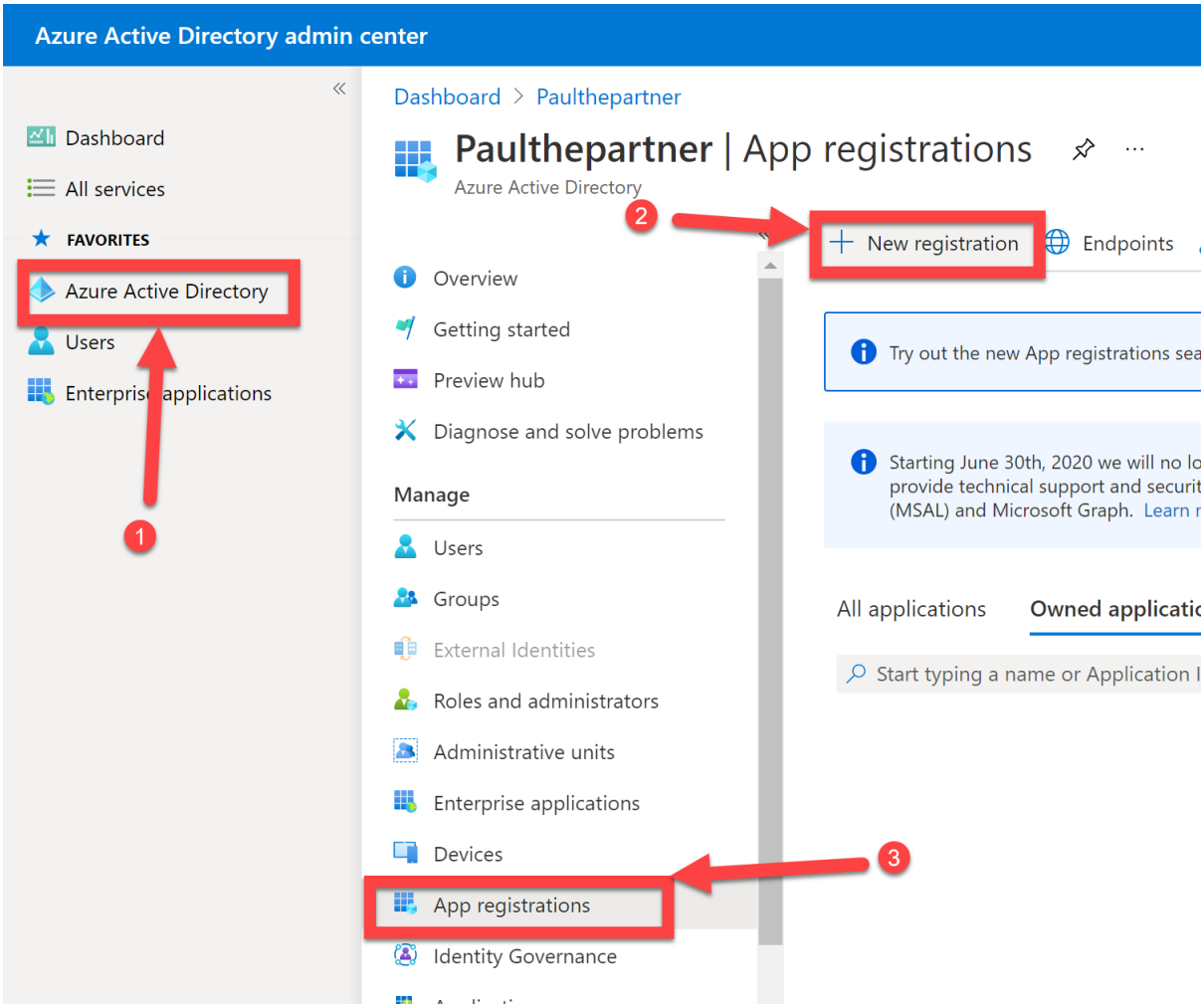
This lab will walk you through the process of creating a Power Virtual Agents Bot for Teams Bot, that can be used to discover the current High Severity Alert status and Secure Score status across multiple customer tenants.

Lab Steps:

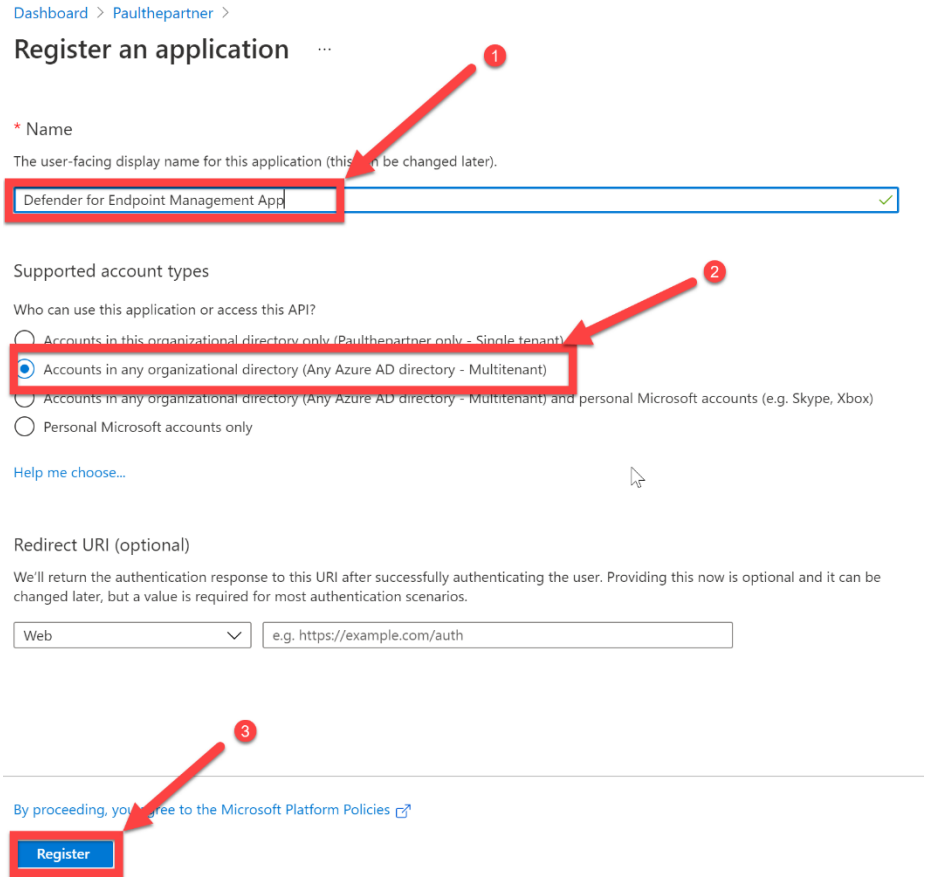
1) Create Azure AD App Registration

In this initial step we will create an Azure AD App Registration that contains the permissions required for you to interrogate the Defender for Endpoint APIs for customer information.

1. Navigate to the Azure Active Directory admin center at https://aad.portal.azure.com/ and login using your **Partner** tenant administrator credentials
2. Select **Azure Active Directory**, select **App Registrations on the right hand side**, click **New Registration**



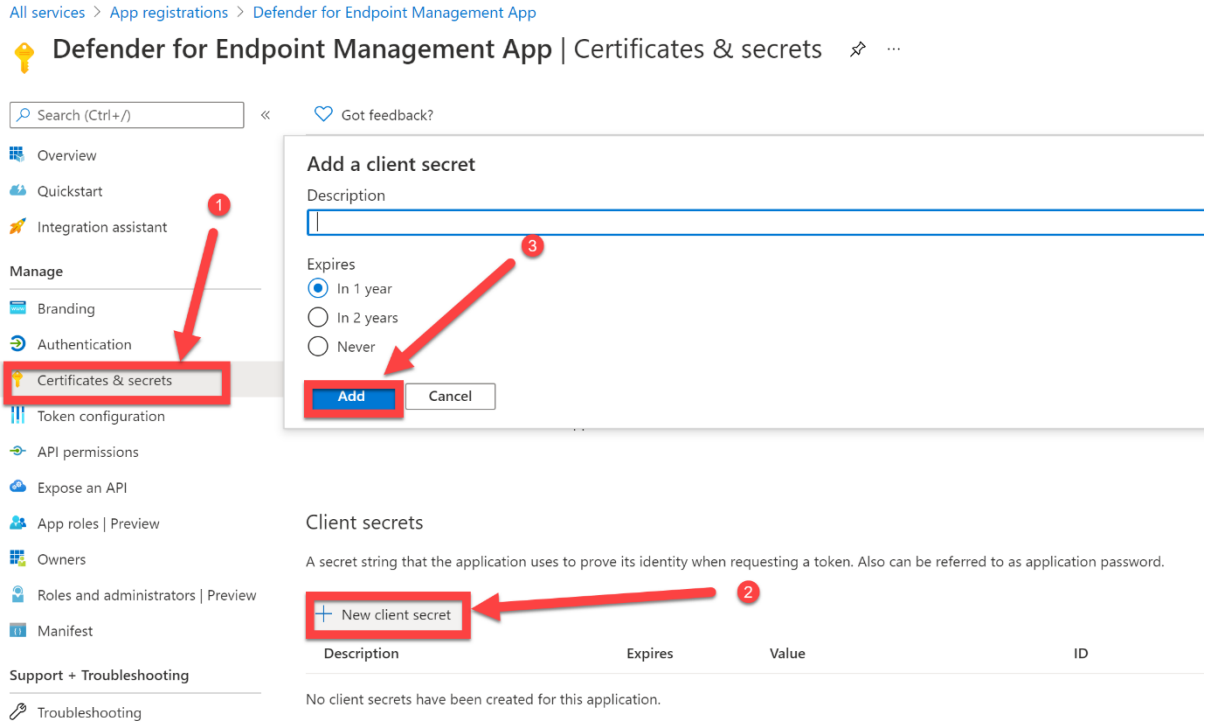
3. Enter a suitable **Name** for your App Registration, e.g. “MDE App”, select **Accounts in any organizational directory** *(note: this enables multi-tenancy for your app registration, which means that once consented by your customers, they will see the application as an Enterprise Application in their own Azure Active Directory tenant)*, click **Register**



4. Capture the Application (client) ID, displayed in GUID format, and store it inside a notepad, you will need to use this later in the lab. It should look similar to:

   Application (client) ID
   b8efd4ec-[redacted]49175c3e9b2b

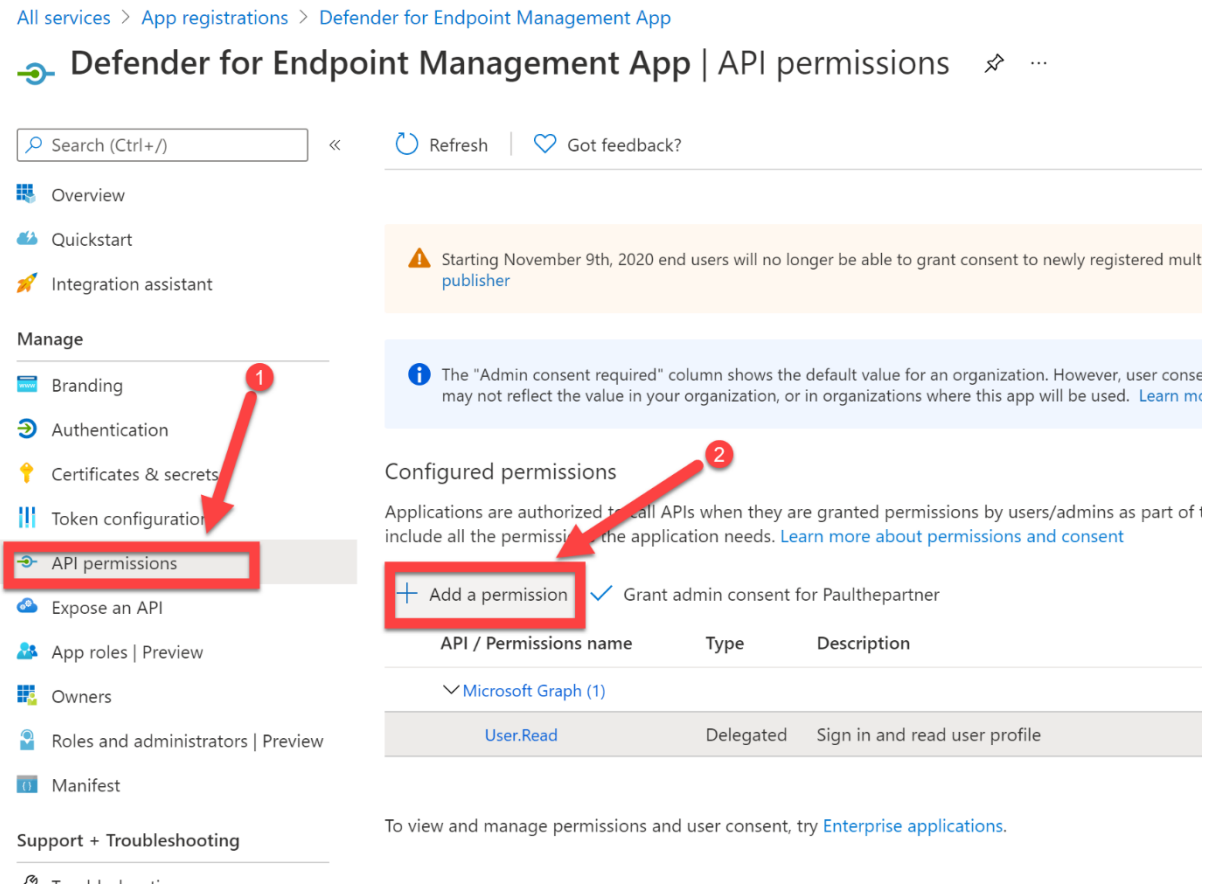
5. We now need to create a secret for our application, to allow us to authenticate as the application, when accessing the Defender for Endpoint APIs to retrieve customer information. Click **Certificates & secrets**, select **New client secret**, click **Add**



6. The Client secret will then be generated and will looks like a random string of letters and numbers. Capture the **Value** displayed and store it alongside the Application (client) Id you captured.

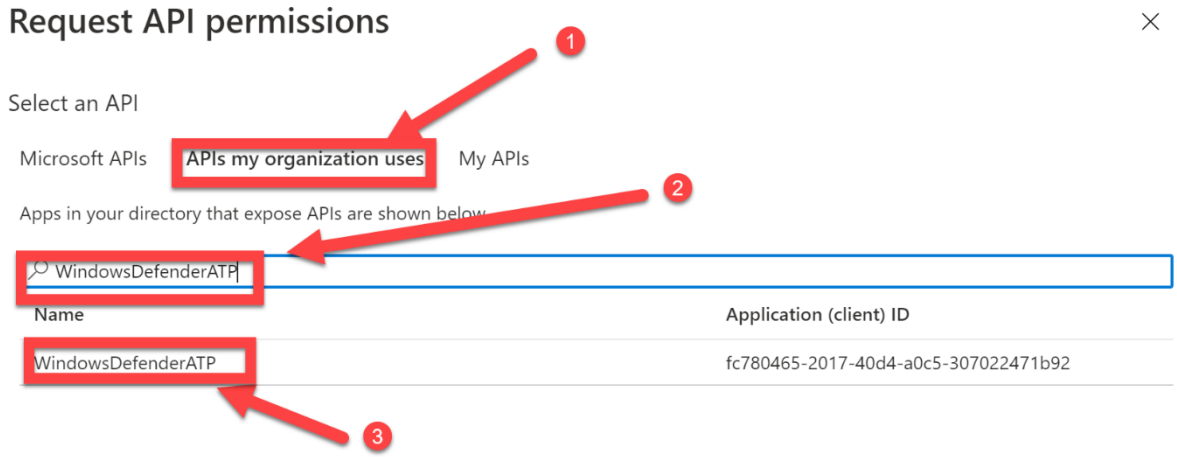
This will then be used a pair of attributes to authenticate your application and receive an access token from Azure Active Directory

7. Select **API permissions**, click **Add a permission**

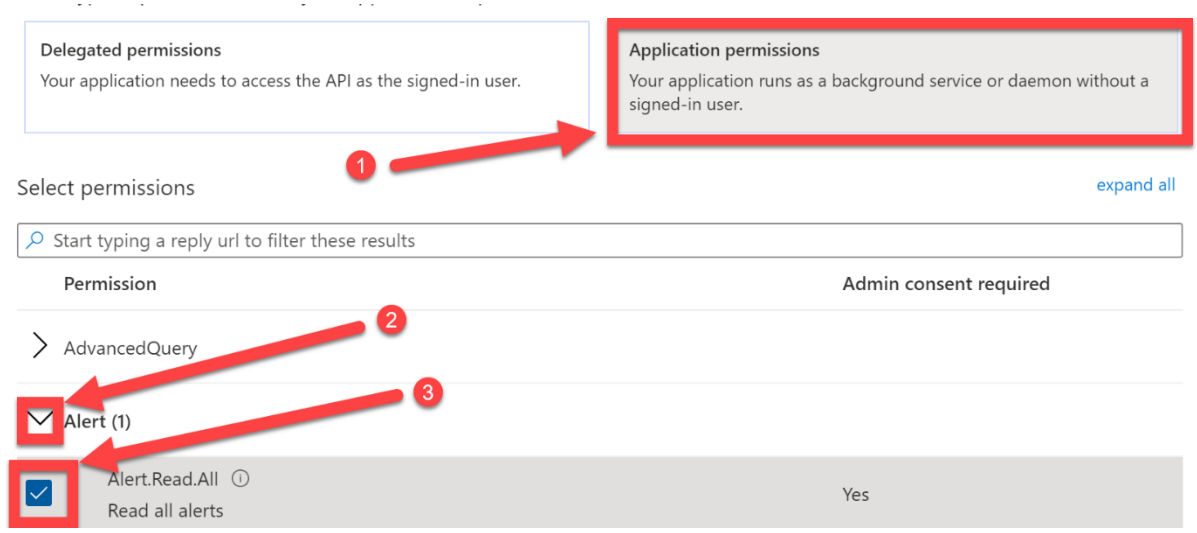


8. Select **APIs my organization uses** and in the search dialog box enter **WindowsDefenderATP**, select **WindowsDefenderATP**

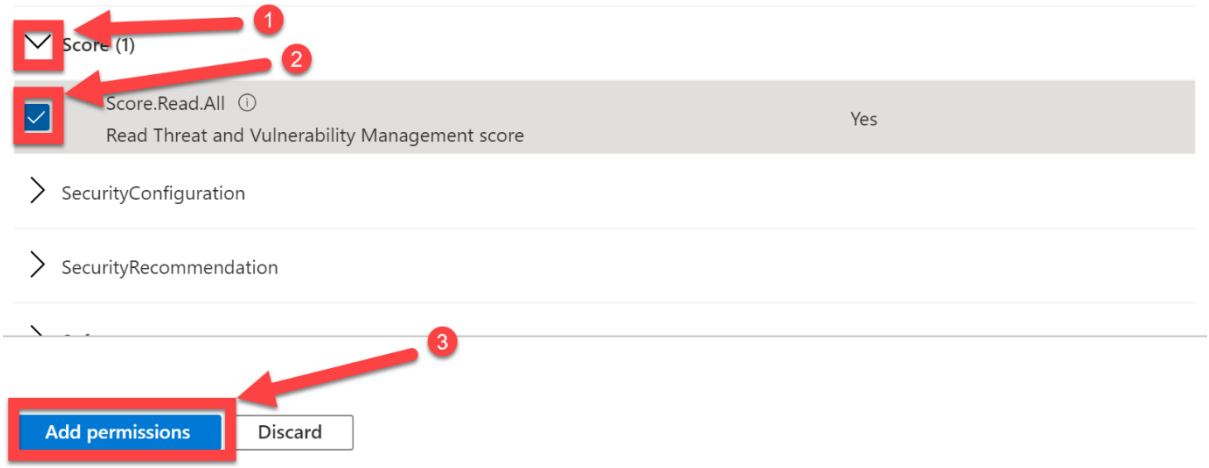
(note: if WindowsDefenderATP does not appear, then you are using a tenant that is not currently licensed for Defender for Endpoint. You will need to sign up for a Defender for Endpoint Trial, this can be achieved with an M365 E5 trial. Once you have activated a trial, go to securitycenter.microsoft.com to setup the tenant, and then return to this step once completed – if you followed the Labs Getting Started guide, you should already have Defender for Endpoint configured and available).



9. Select **Application permissions**, expand **Alert** and select **Alert.Read.All**



10. Scroll down, expand **Score**, select **Score.Read.All**, click **Add permissions**



11. You have now successfully configured your App Registration and it is now ready for you to ask customers to consent to the permissions you have just selected.

In the next section, we will walk through the customer consent experience, which will provide your app with the permissions required to interrogate the Defender for Endpoint APIs for Alert and Score status updates.

1) Provide Customer Consent for your Application to GET the required information from the Defender for Endpoint APIs

We will now role-play as a customer and provide Administrator Consent for the application we have just registered in Azure Active Directory. For this lab, we will need to perform the consent process for all 3 customer tenants.

Contoso – Tailspin Toys – Northwind Traders

1) Firstly, we need to generate an admin consent URL. To do this, we will need the Application (client) ID from section 1. We then need to replace ***CLIENTID*** in the following URL with your own client ID:

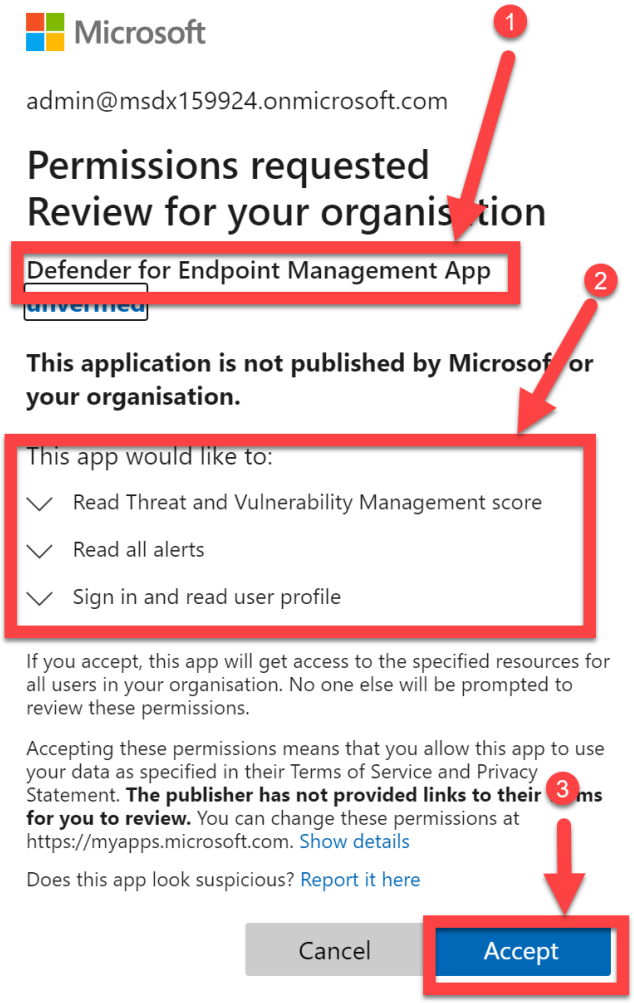
https://login.microsoftonline.com/common/adminconsent?client_id=CLIENTID&state=12345&redirect_uri=http://localhost/myapp/permissions

2) Now that we have the admin consent URL, we will need to visit this URL and authenticate as a Global Administrator using your **Customer** tenant credentials (check https://aka.ms/defendermasterclass-repo for these)

You will need to repeat this process for all 3 customer tenants.

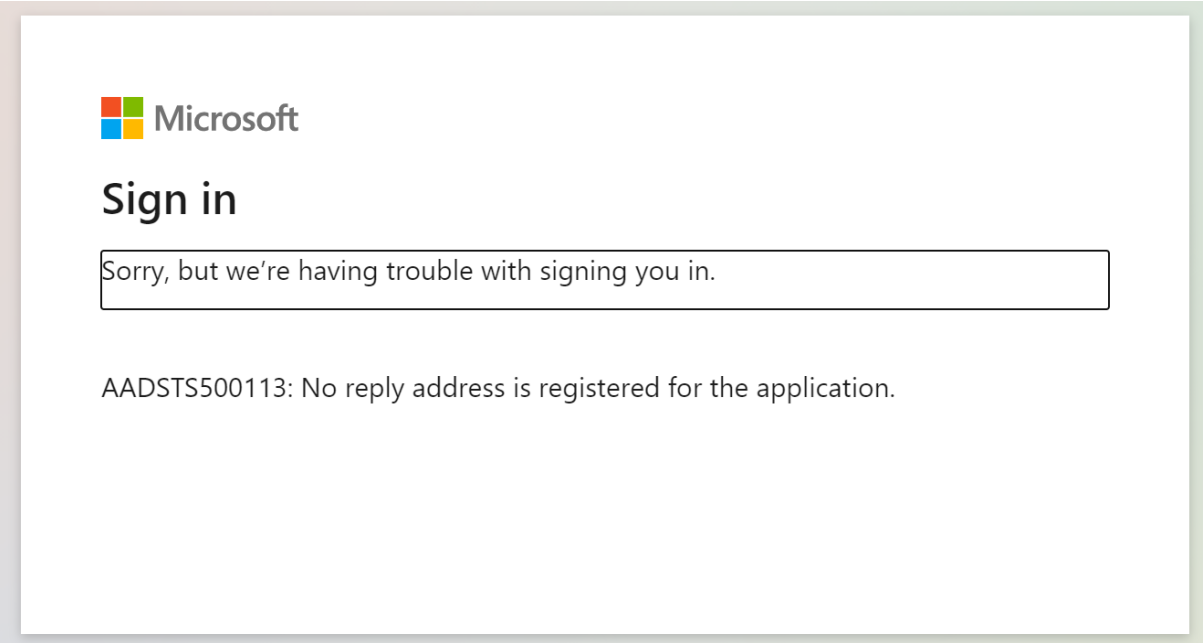
3) Once you have navigated to the admin consent URL, authenticate as the customer Global Administrator
4) Once you have authenticated as the customer, you can review the consent page. Firstly, make sure that the **App Name matches your App Registration**, secondly **review the permissions being asked for**, the permissions should match what is configured in your App Registration, finally click **Accept**

You will get an Error – this is expected!



5) Once you have accepted, you will be presented with the following error message. Do not worry, this is expected.
6) Repeat this process for the remaining customer tenants until complete, and then move on to the next section. – for this lab we have 3 x customer tenants.

(note: in a real world scenario, you would redirect the successful consent/authorisation request back to your application, and this error message would not be displayed, but for the purpose of this demo, it is fine for us to ignore this error message and move on)

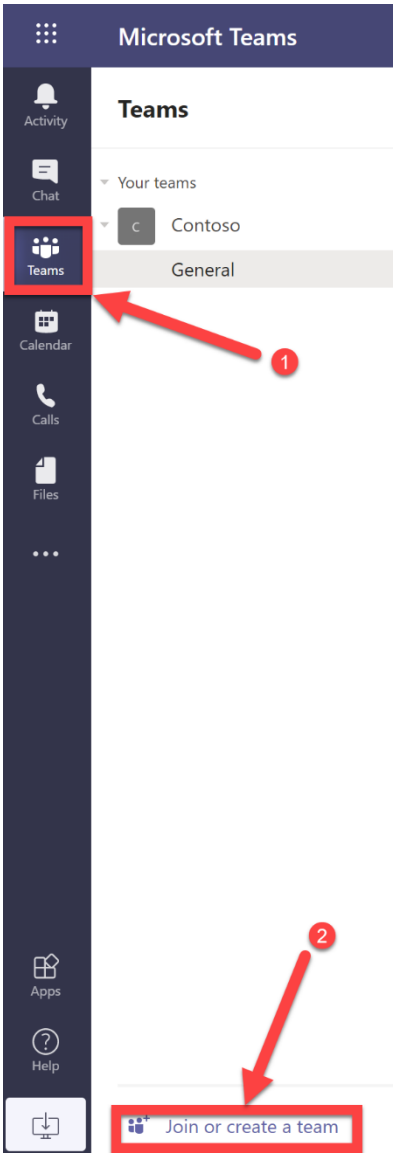


7) Your customers have now successfully granted your application the permissions required to perform HTTP GET requests, to the Defender for Endpoint API Alerts and Score resources. We can now create the Power Virtual Agents Bot.

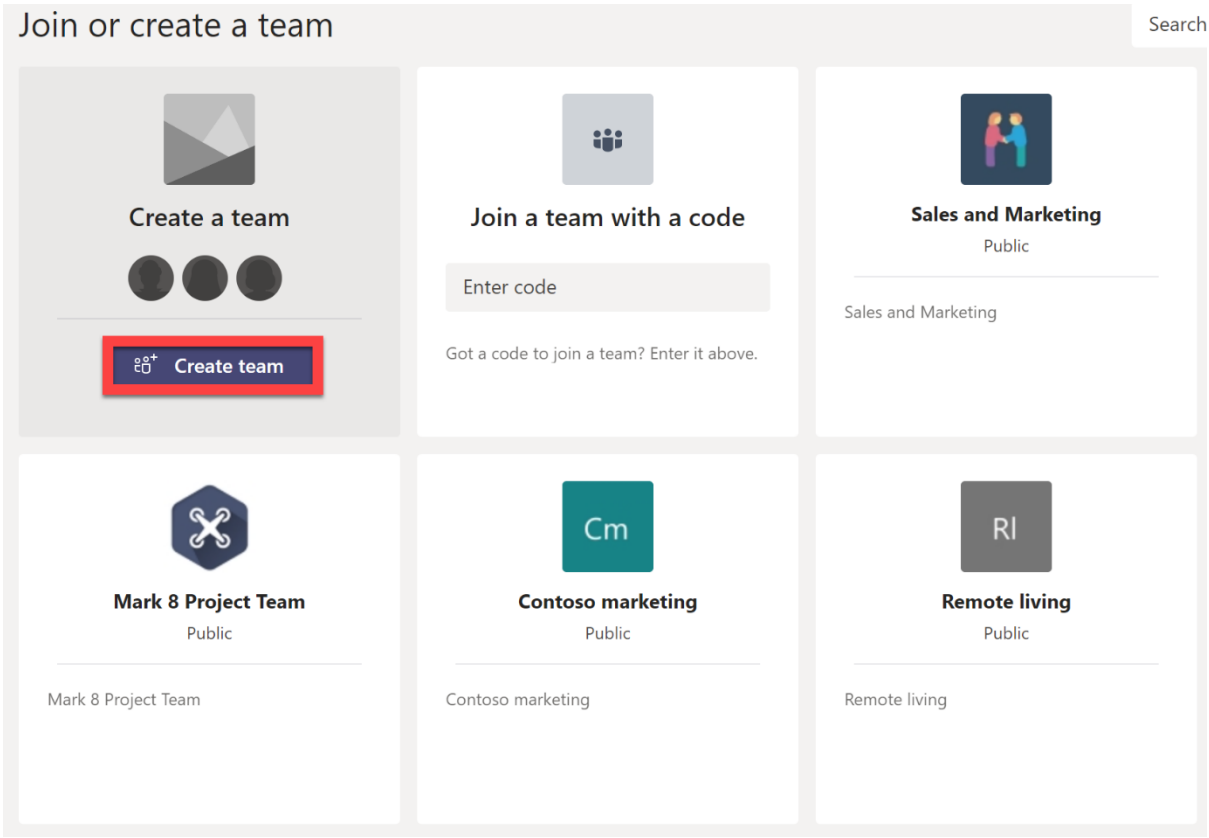
2) Create the Power Virtual Agents for Teams Bot

We will now create a Power Virtual Agents for Teams Bot and configure a topic to allow partners to query their customer's Microsoft Defender for Endpoint Alerts and Score status.

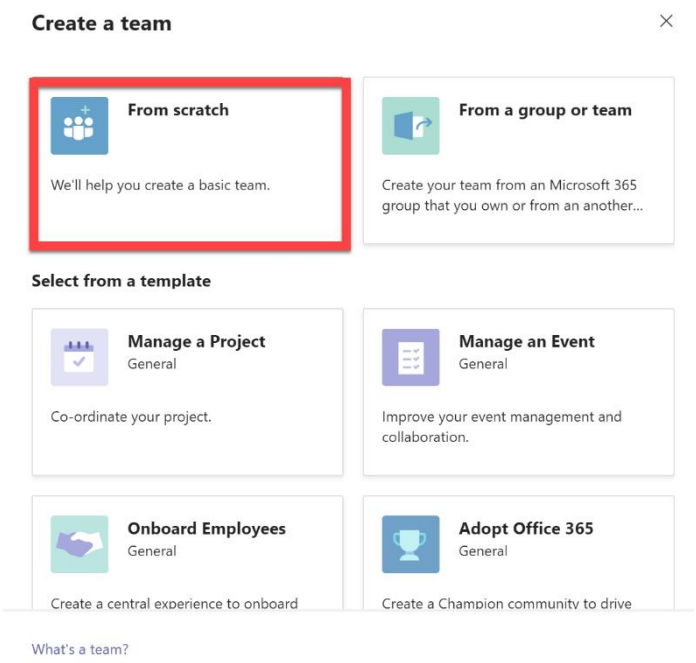
1. Navigate to https://teams.microsoft.com – login using your **Partner** tenant credentials
2. Before creating the Power Virtual Agents Bot, we need to create a Team for the bot to reside within. Select **Teams** from the Teams App Bar, click **Join or create a Team**



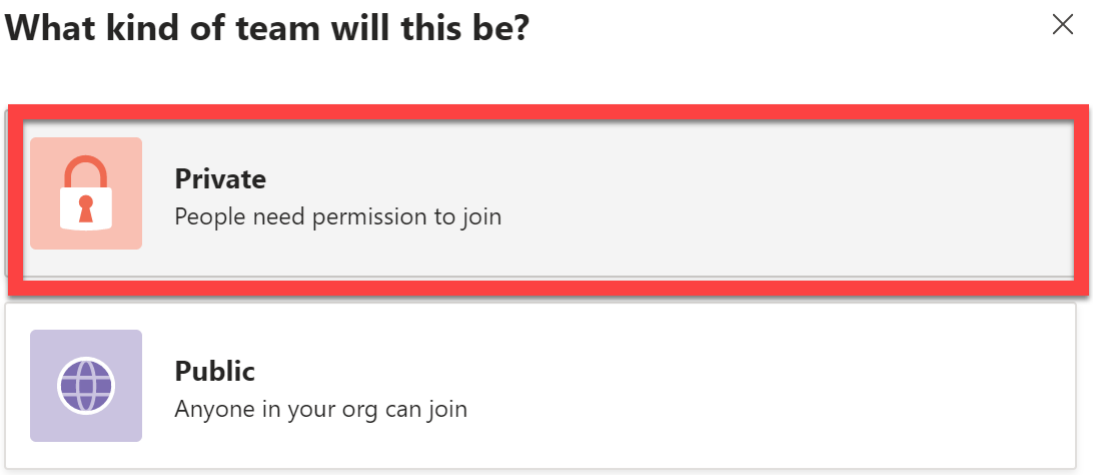
3. Click **Create Team**



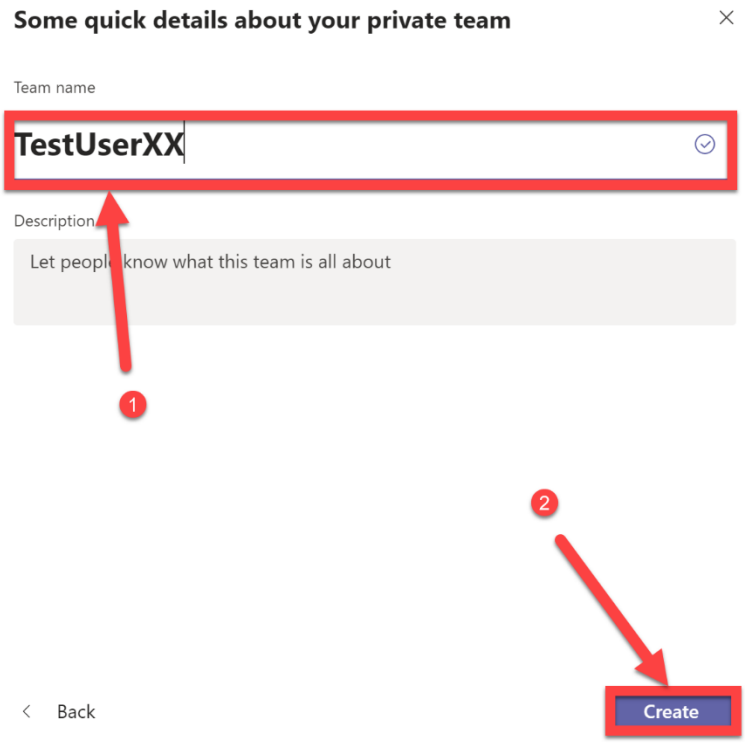
4. Select **From scratch**



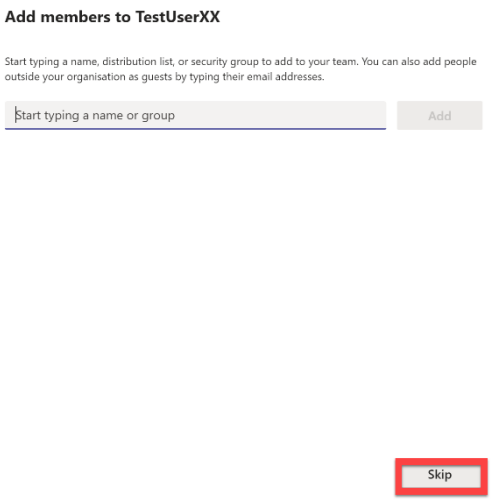
5. Select **Private**



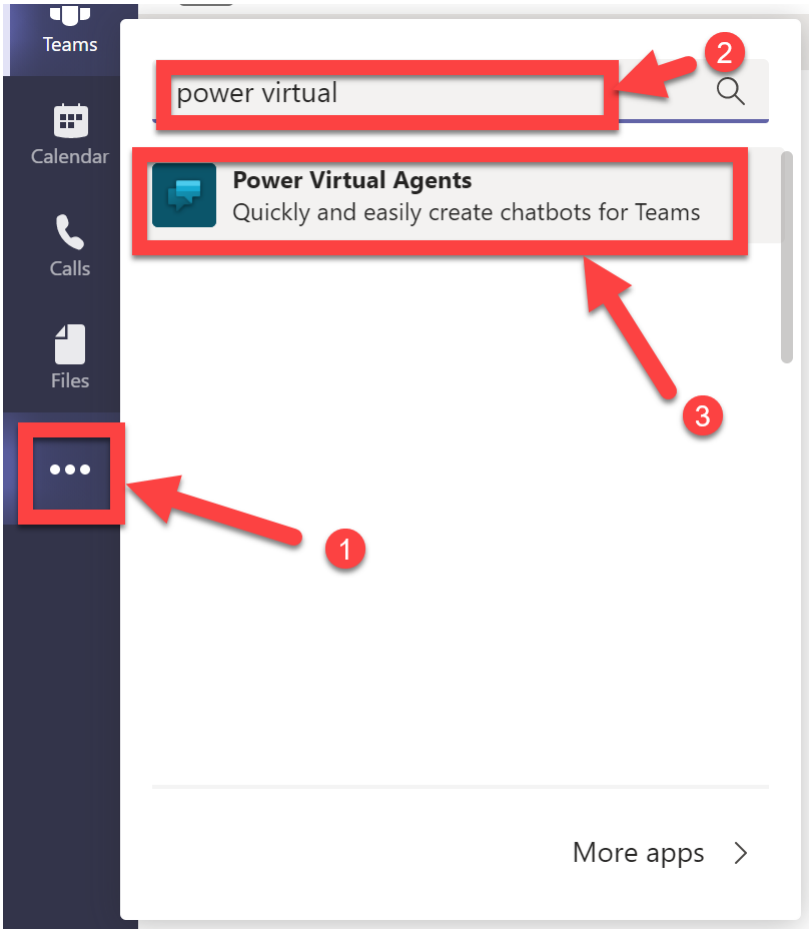
6. For the Team Name, call it after your username, for example, if you are TestUser12, call the Team **TestUser12**, click **Create**



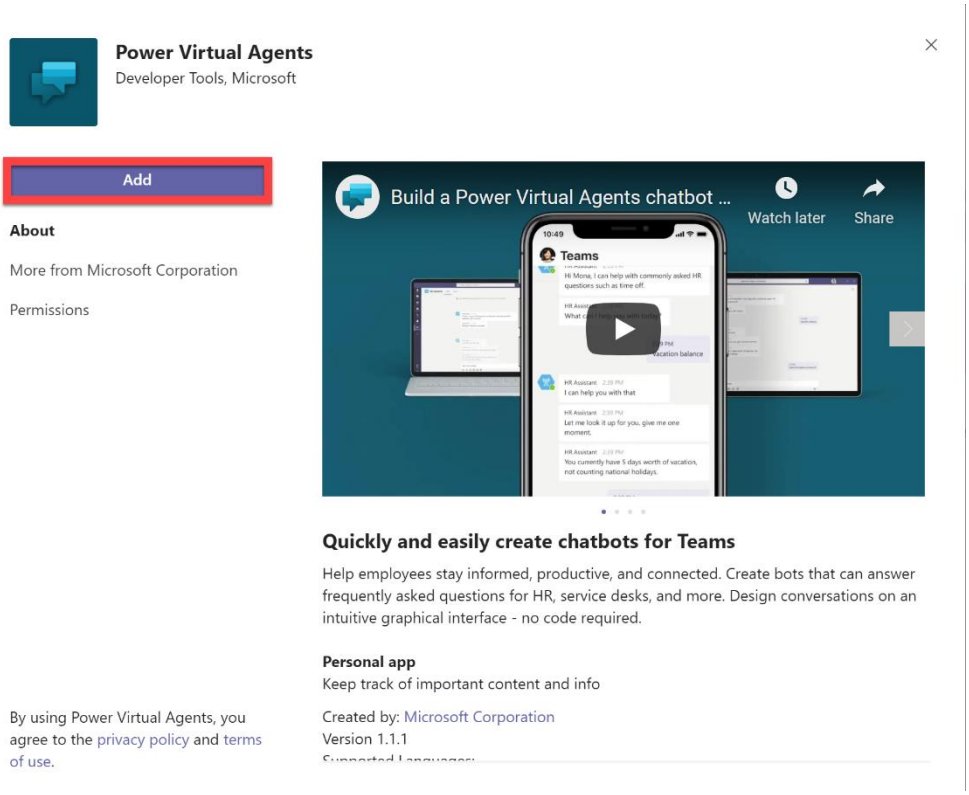
7. Click **Skip**



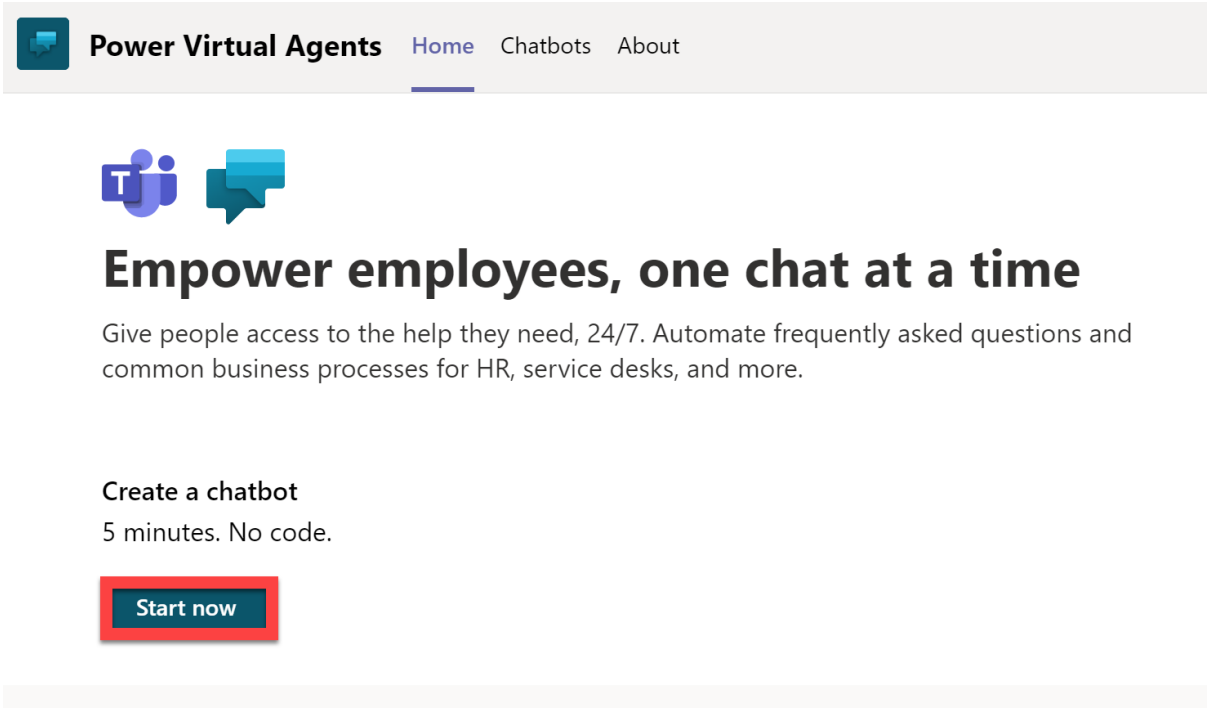
8. From the Teams App bar select the **…**, search for **Power Virtual Agents**, select the **Power Virtual Agents** app



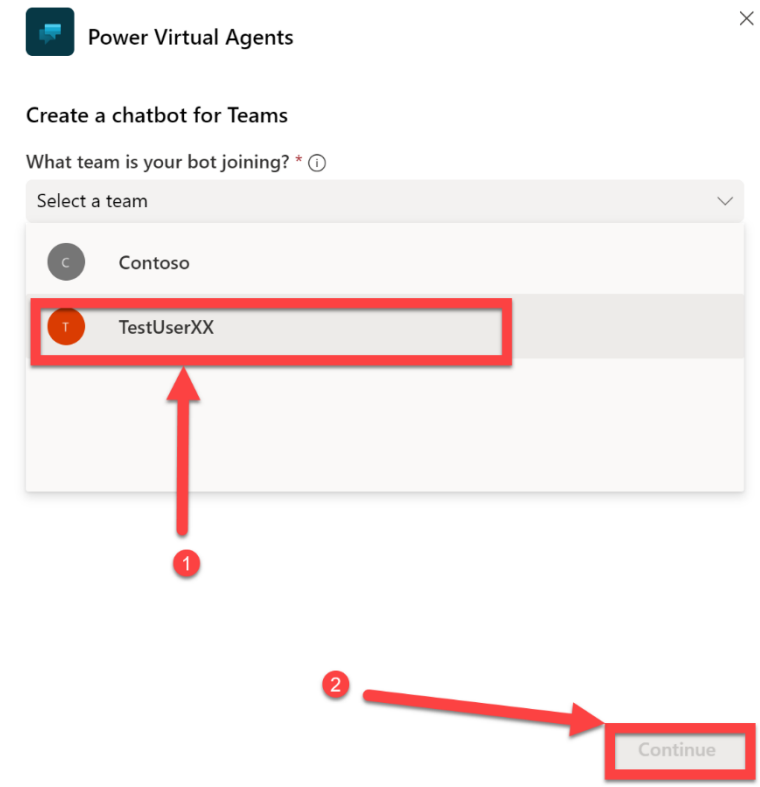
9. Click **Add**



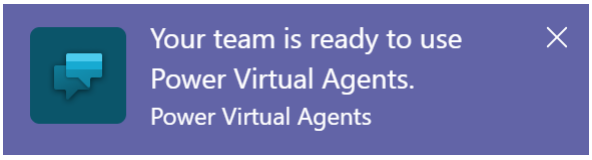
10. Click **Start now**



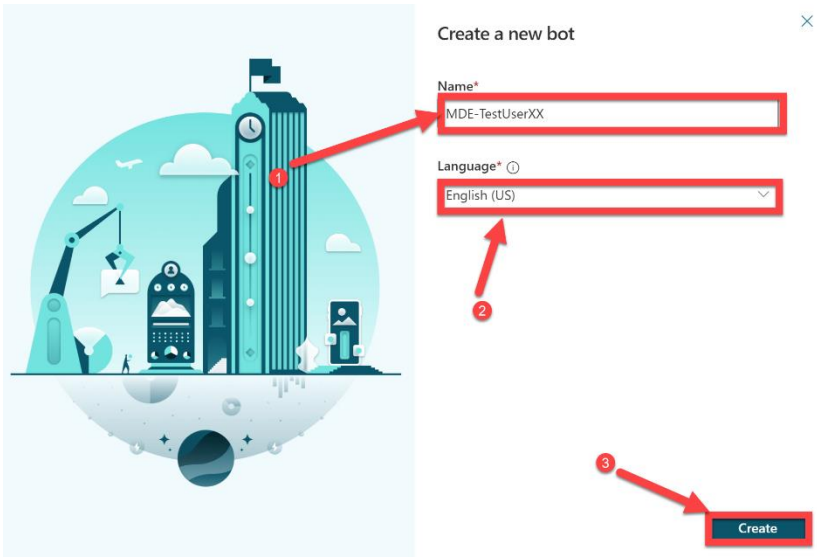
11. Select **the Team you just created**, and then click **Continue**



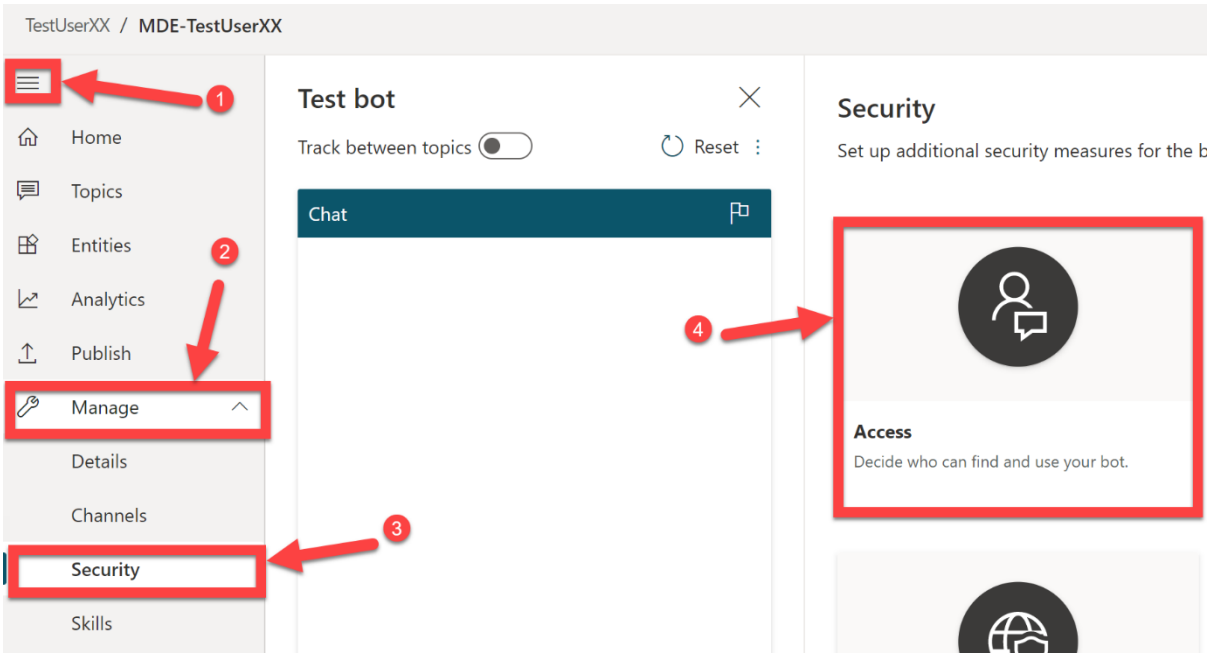
12. After a short period of time, you will receive the following notification. Click the notification



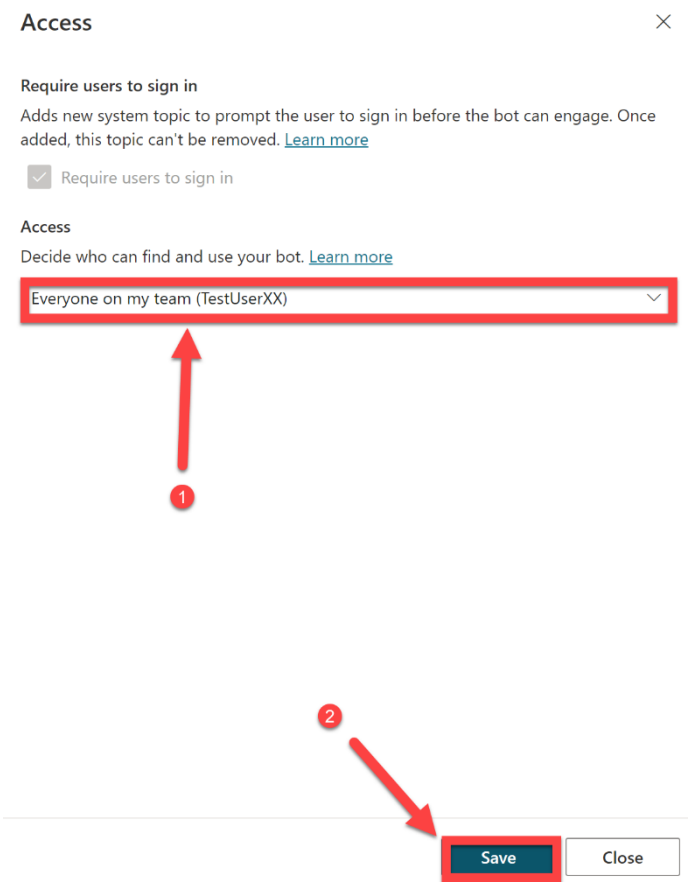
13. For the name, please enter **MDE-** and then the name of your user account. In this case I have used **MDE-TestUserXX**, select a **Language**, click **Create**



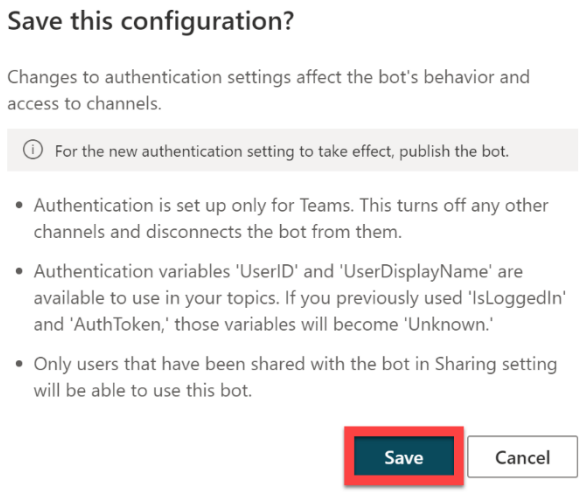
14. Select the **Hamburger menu**, to expand the navigation bar, click **Manage**, select **Security** and then click **Access**



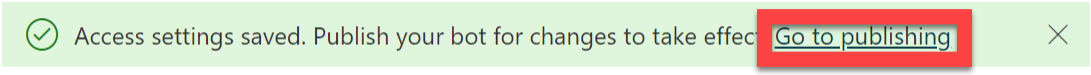
15. Using the dropdown menu, select **Everyone of my team**, click **Save**



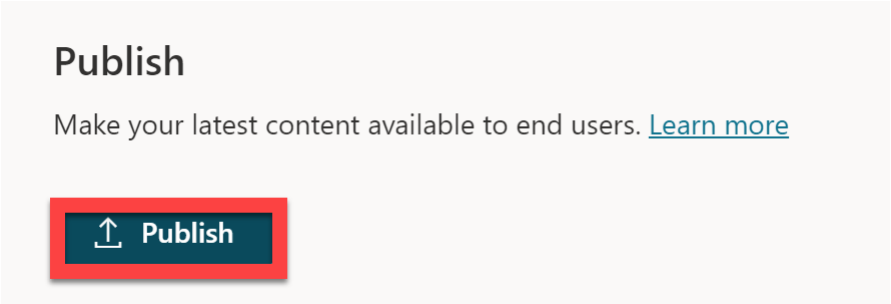
16. Click **Save**



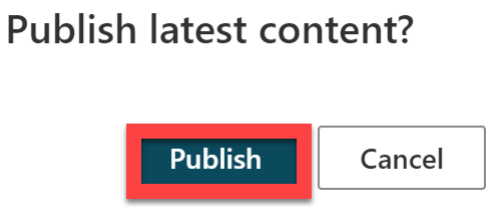
17. Click **Go to publishing**



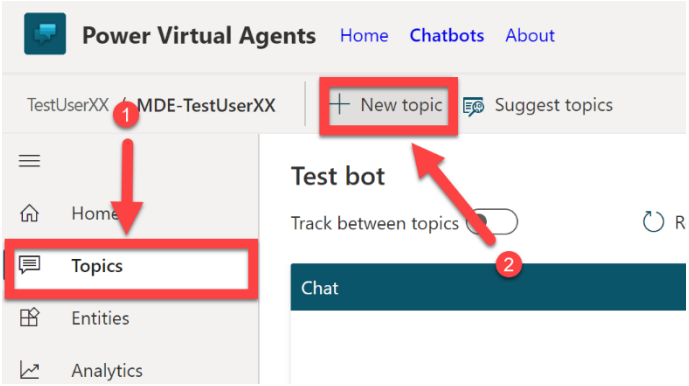
18. Click **Publish**



19. Click **Publish**



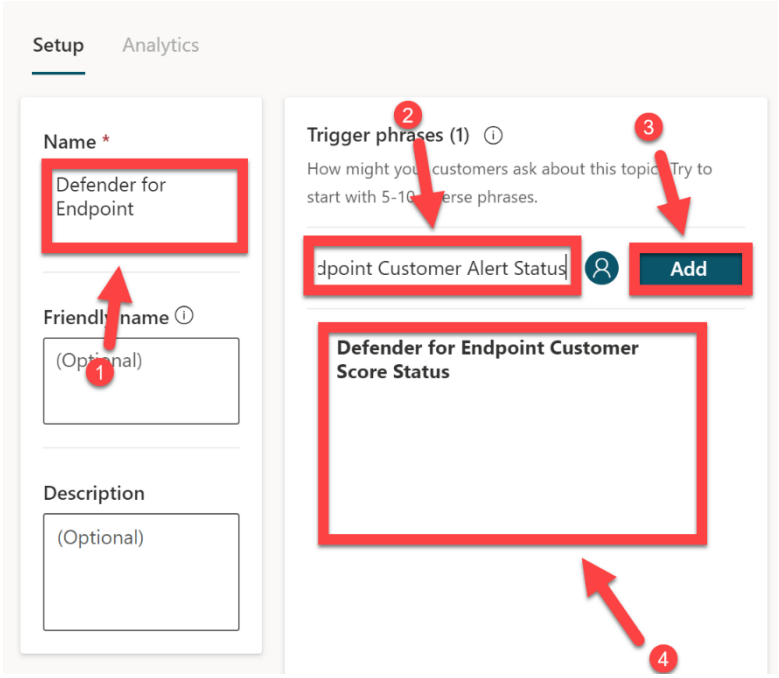
20. Select **Topics** from the navigational menu, click **New topic**



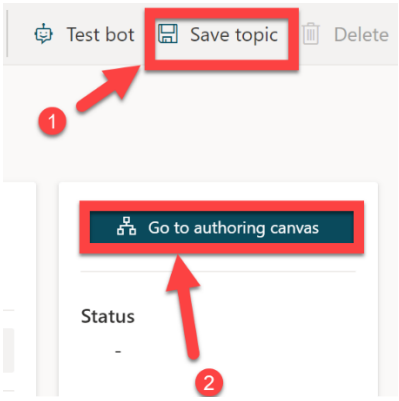
21. For the name of the topic, enter **Defender for Endpoint**, in the Trigger phrases enter two phrases:
    a. **Defender for Endpoint Customer Score Status**, click **Add**.

b. Add the following addition trigger phrase, **Defender for Endpoint Alert Status**, click **Add**.

22. You have now added the phrases that will trigger this topic, you can add additional trigger phrases if required

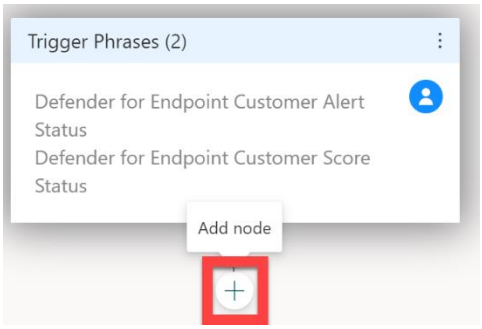


23. Select **Save topic**, click **Go to authoring canvas**

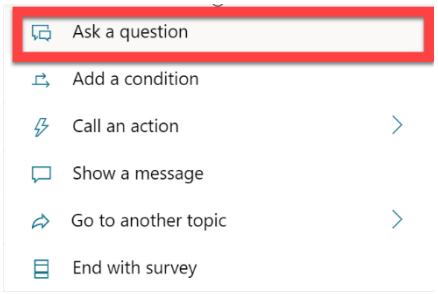


24. On the Message, select the **…** (more options), and select **Delete**

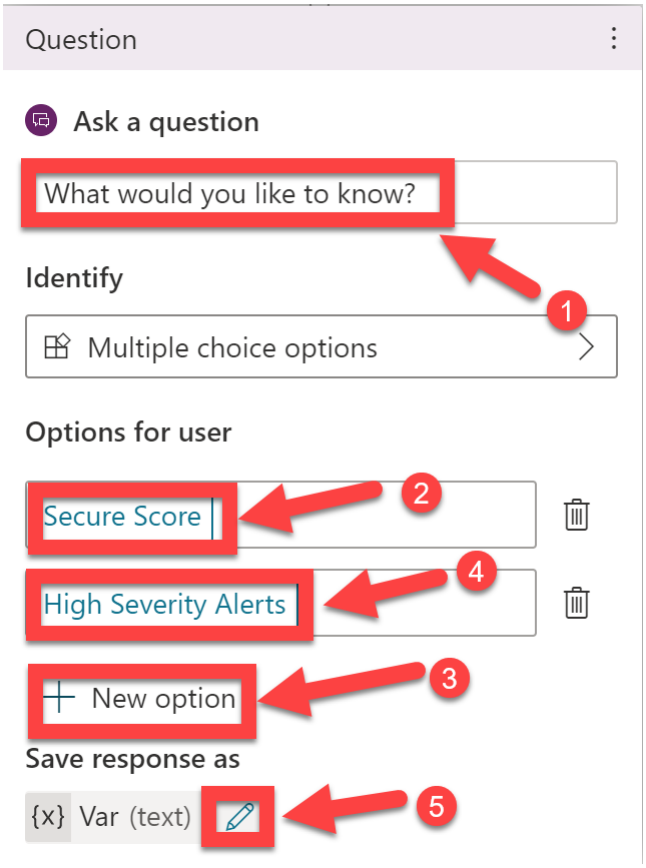
25. Select the **+** button



26. Select **Ask a question**

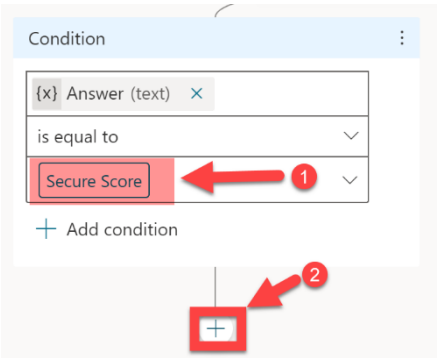


27. Enter **What would you like to know?** In the text dialog box, enter **Secure Score** in the Options for user dialog box, select **New option**, enter **High Severity Alerts** in the additional Options for users dialog box, click on the **pencil icon** (edit)

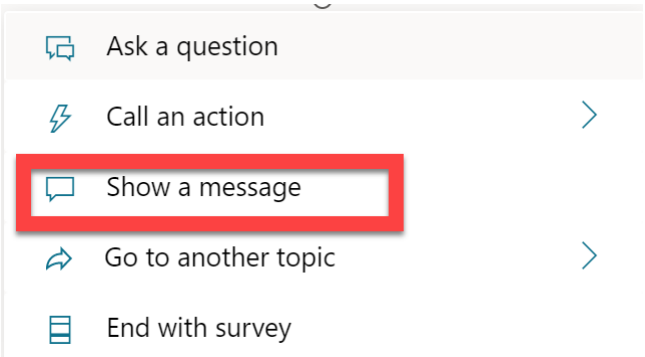


28. Change the variable name to **Answer**

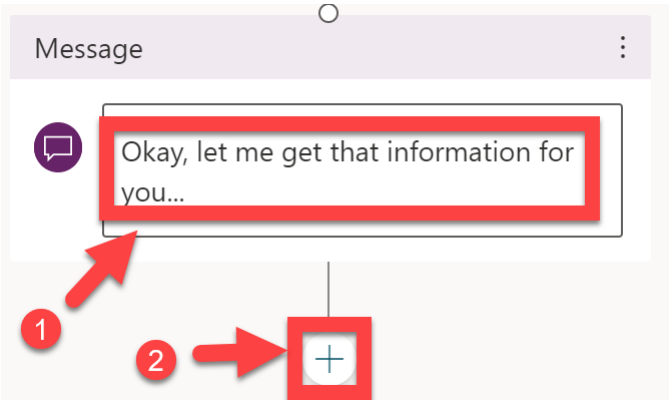
29. There will now be multiple branches that have started to establish below the question you just created, and depending on how the questions are answered, it will take users down a defined path. We will use the answer variable, you just created, later in our flow to decide which Defender for Endpoint APIs to retrieve information from. Under the **Secure Score** branch, select the **+** button



30. Select **Show a message**

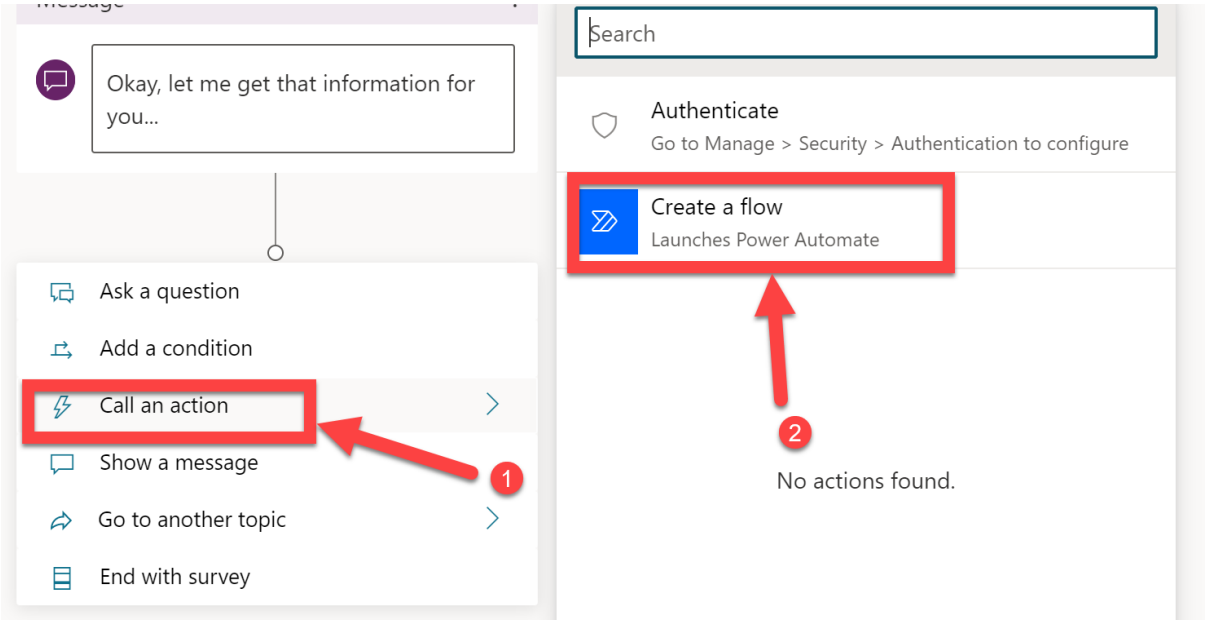


31. In the message dialog box enter, **Okay, let me get that information for you…**, click the **+** button

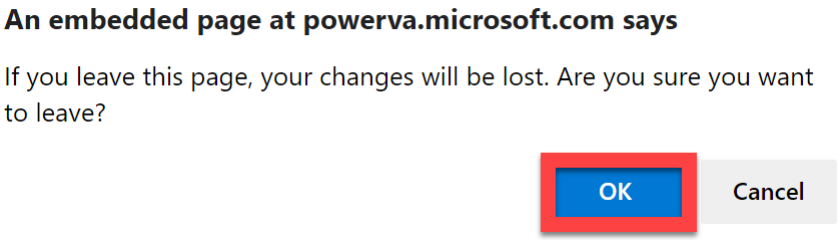


32. Click on **Save – this is important!**

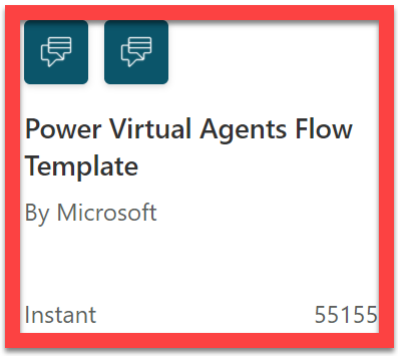
33. Select **Call an action**, click **Create a flow**



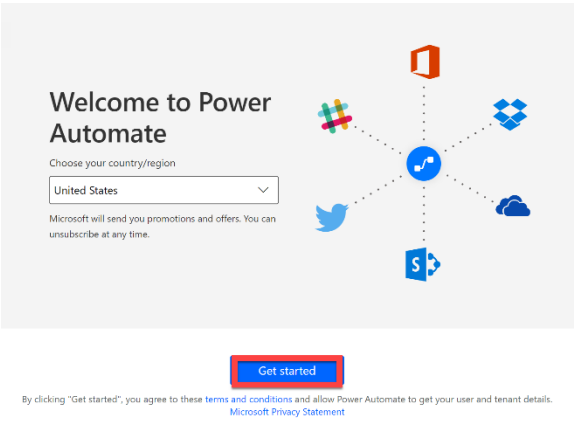
34. Click **OK – make sure you have saved your Bot progress before leaving!**



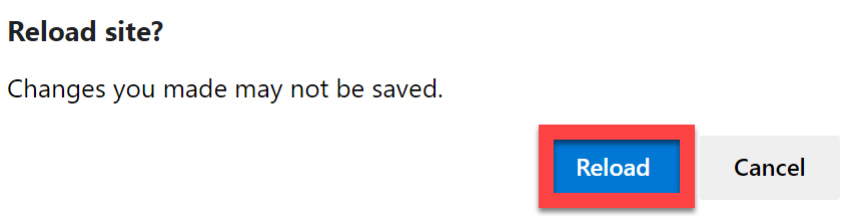
35. Select the **Power Virtual Agents Flow Template**



36. Click **Get Started**



37. Select **Reload**

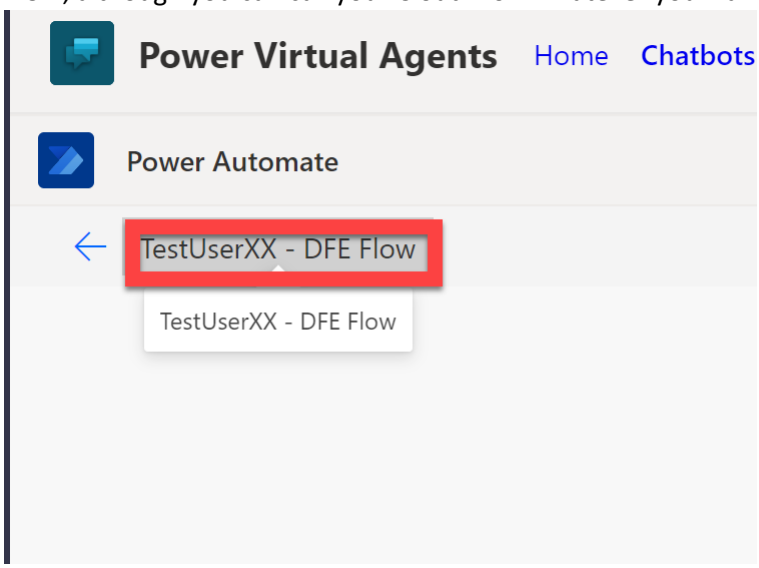


You have now configured the initial topics and initialised the answer variable that we will use in the upcoming flow to determine which API Resources we should gather information from. **We will now move on to creating the Flow, which is the most complex and in-depth of the tasks completed so far.**

3) Create the Power Automate Cloud Flow to call the Defender API Endpoints

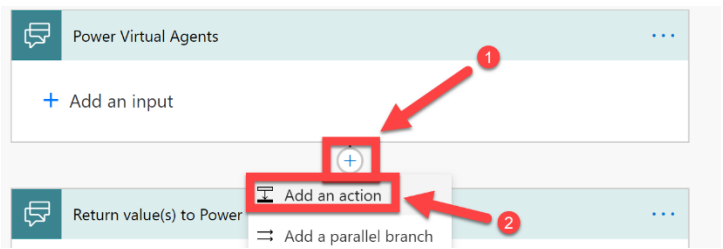
In this step we will define the actions in the Power Automate Cloud Flow that will be used to gather information from the Defender for Endpoint API Resources. This information will then be passed back to the Virtual Agent Bot, so that it can provide the information to the end user.

1) Firstly, let's give the Cloud Flow a useful name, I have used the name **TestUserXX – DFE Flow**, although you can call your Cloud Flow whatever you want

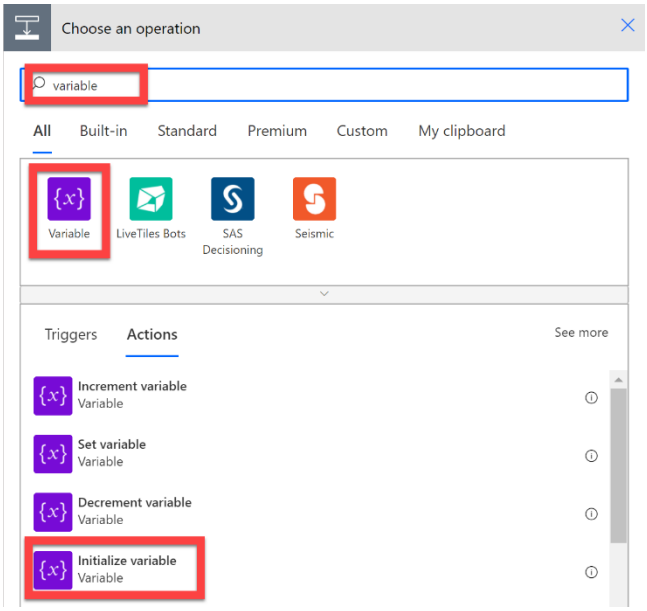


We need to start by initialising a bunch of variables that we will use within the Cloud Flow to make decisions and store data within.

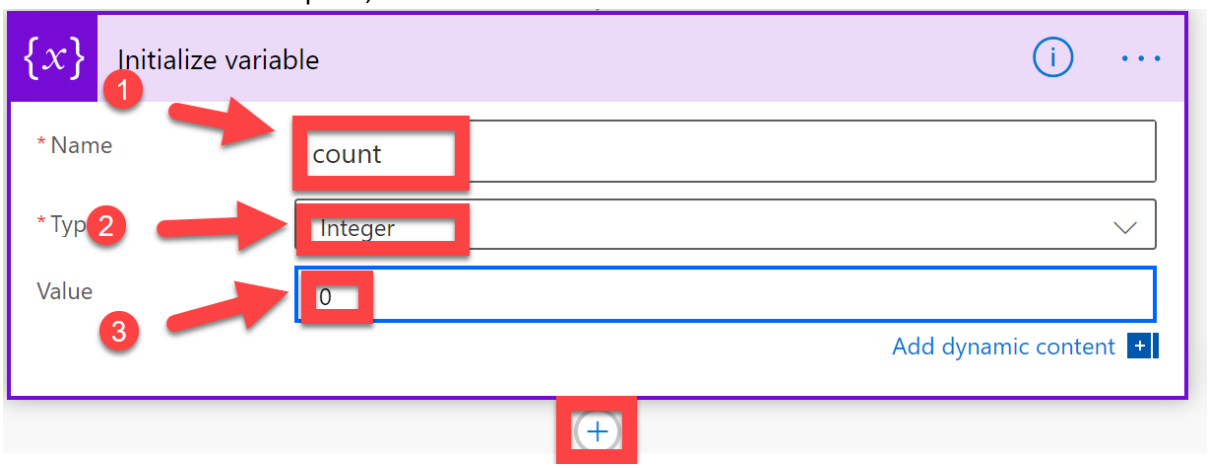
2) Select the **+** button, click **Add an action**



3) In the search dialog box enter **variable**, select **Variable** and then click **Initialize variable**



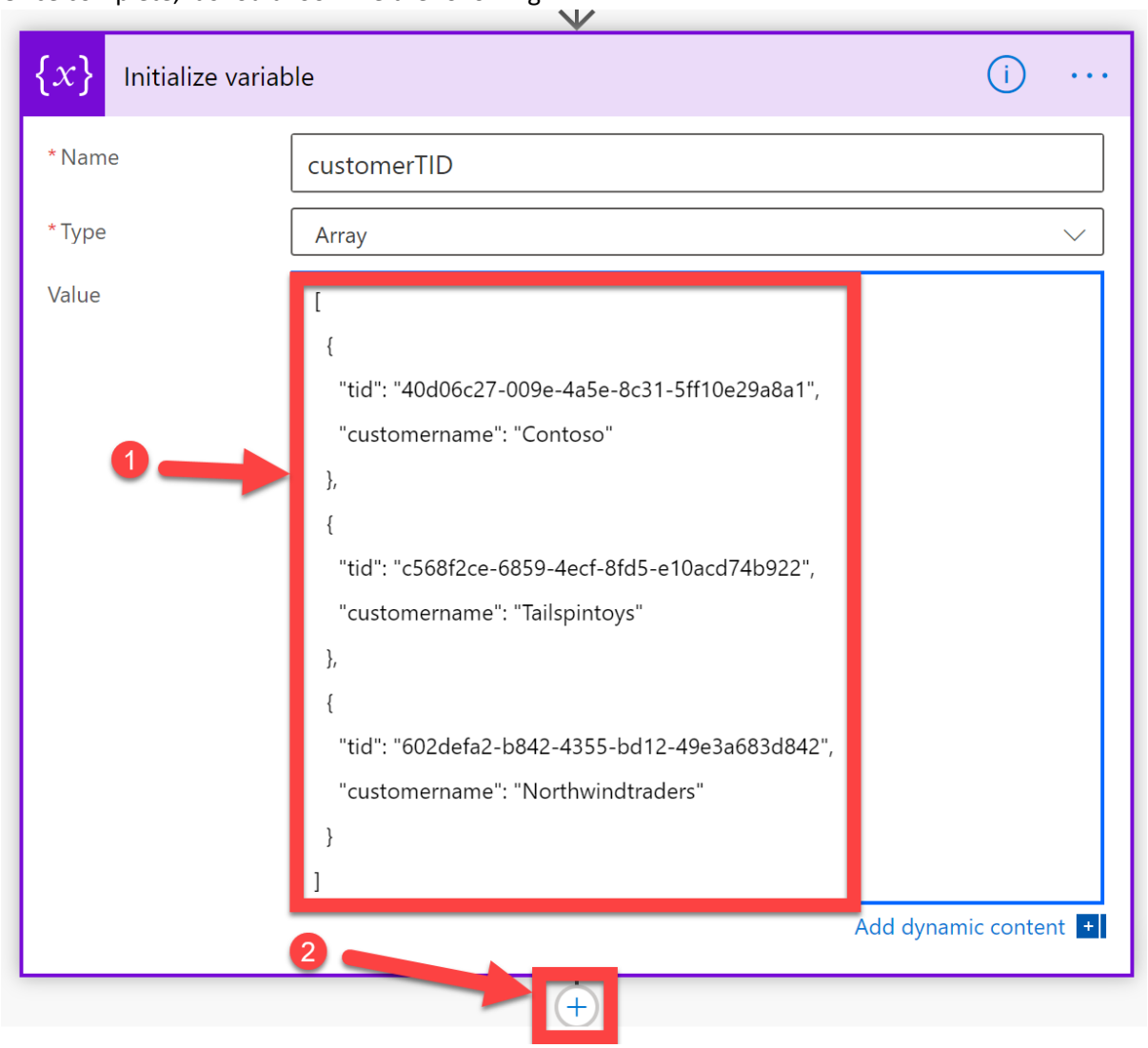
4) The first variable we need to initialise will be used to count and store the number of high severity alerts for each customer. For this we will need an Integer (whole number) variable, that has a zero value to begin with. Enter the Name **count**, select **Integer** as the Type, enter **0** as the Value. Once complete, click the **+** button



5) We now need an additional **2** variables.
   a. For this, please initialise a variable of type, **String**, with the name **Response**, the initial value can be left empty for the time-being. We will use this variable to pass the results of the API calls back to the virtual agent.
   b. The next variable we need is of type **Array**, with the name **customerTID**. We will use this to store the customer names and Azure AD Tenant ID of the customer, in JSON format. The values of an array using JSON can look a little complicated as first, but don't worry, you are simply working with items (or objects if you prefer). And within those items are keys and values.
      i. For the array, you will need to enter the following Value:

```
[
  {
    "tid": "40d06c27-009e-4a5e-8c31-5ff10e29a8a1",
    "customername": "Contoso"
  },
  {
    "tid": "c568f2ce-6859-4ecf-8fd5-e10acd74b922",
    "customername": "Tailspintoys"
  },
  {
    "tid": "602defa2-b842-4355-bd12-49e3a683d842",
    "customername": "Northwindtraders"
  }
]
```

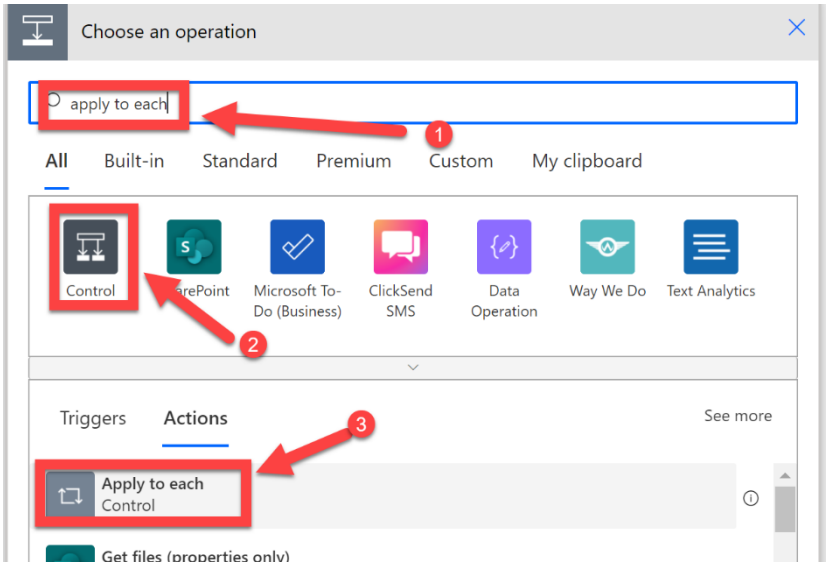
Once complete, it should look like the following:



(note: in a real-world scenario, you would likely store customer information, such as their name and AAD Tenant ID, in a database (probably Cosmos DB or Azure SQL), and then

retrieve those values from the database during your Cloud Flow. But for this scenario, we can skip that step and store this information directly inside an array. Should you want to include additional customers in this scenario, you will need to extend the array with additional objects, that contain the customer name and TID.)

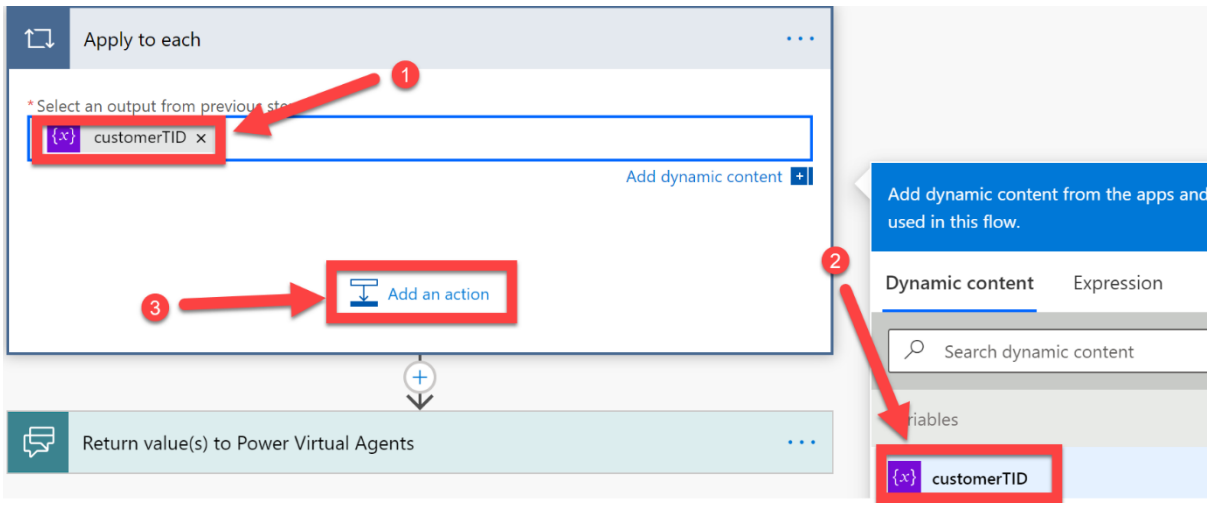
6) Once complete, click the **+** button
7) We now need to process each item within your array and run steps in sequence for each customer. Search for **Apply to each**, select **Control** and then select **Apply to each**



8) Click the **Select an output from the previous step form** and select the **customerTID** variable from the dynamic content selector.

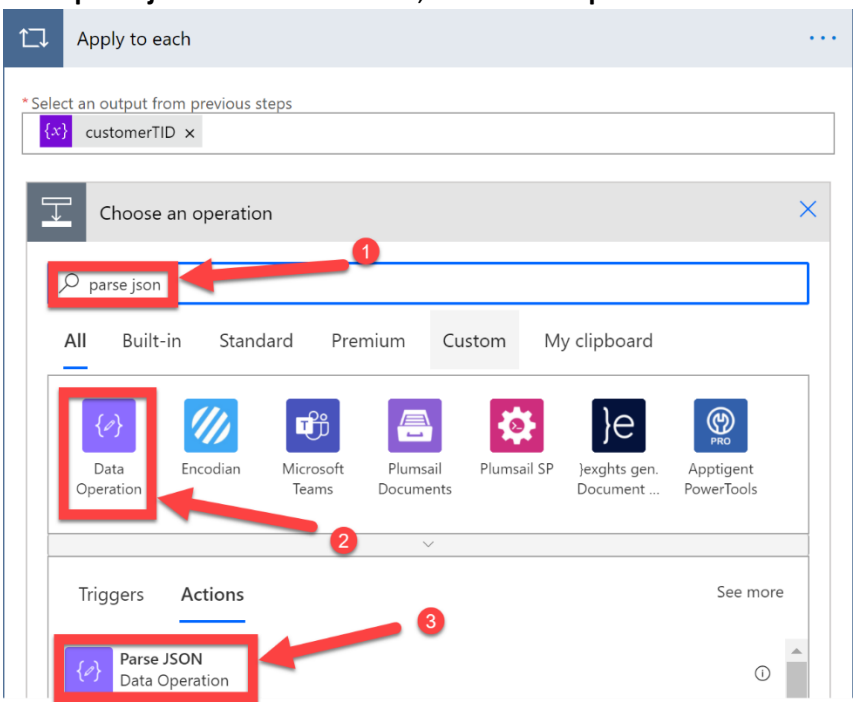
This will then instruct your Cloud Flow to repeat any actions, that are contained within this apply to each action, for every item (in sequence) inside of the customerTID array

9) Select **Add an action**



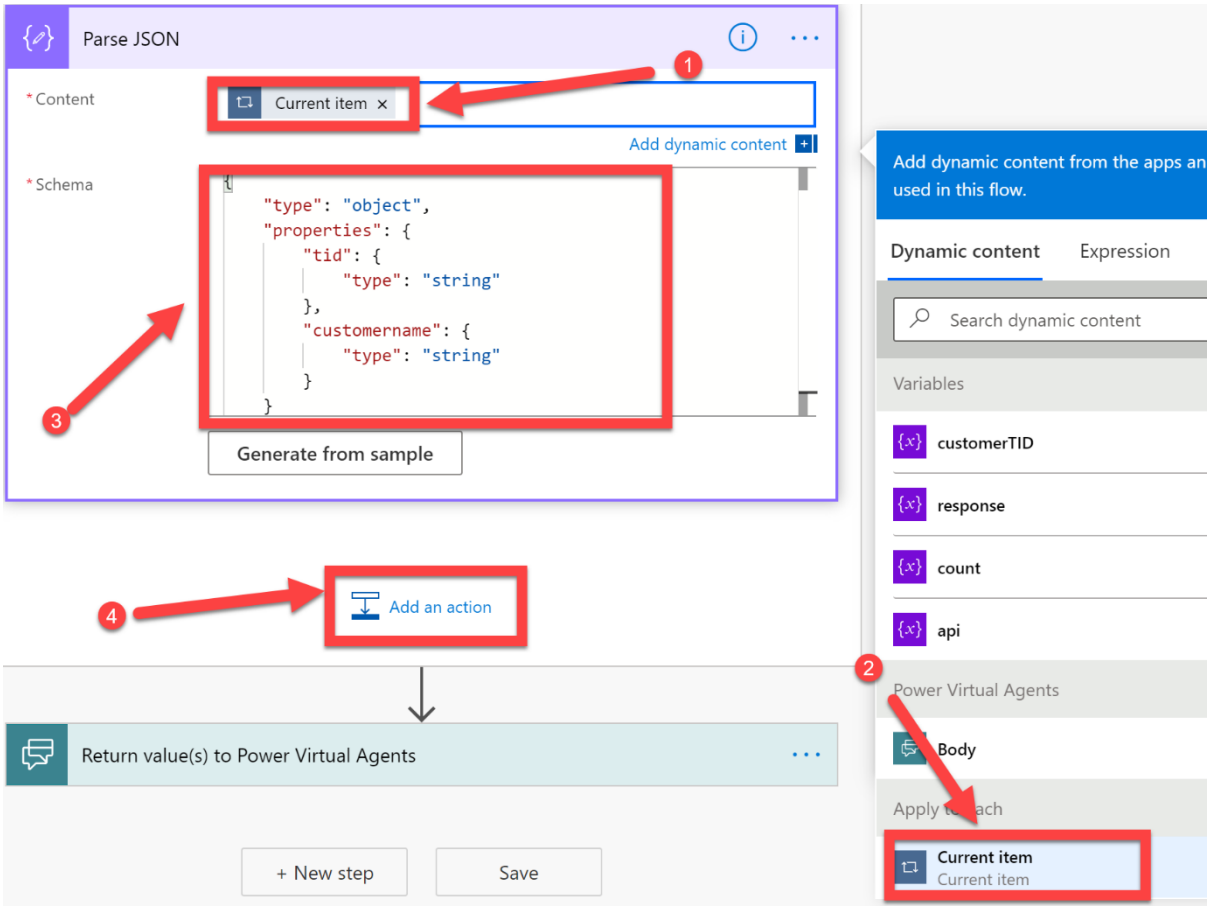
Now we need to parse each item in the array, to allow us to use the CustomerName and TID in future steps. As this action is performed within the Apply to Each action, we can use the Item() function to achieve this in the next few steps.

10) Enter **parse json** in the search field, select **Data operation** and then select **Parse JSON**



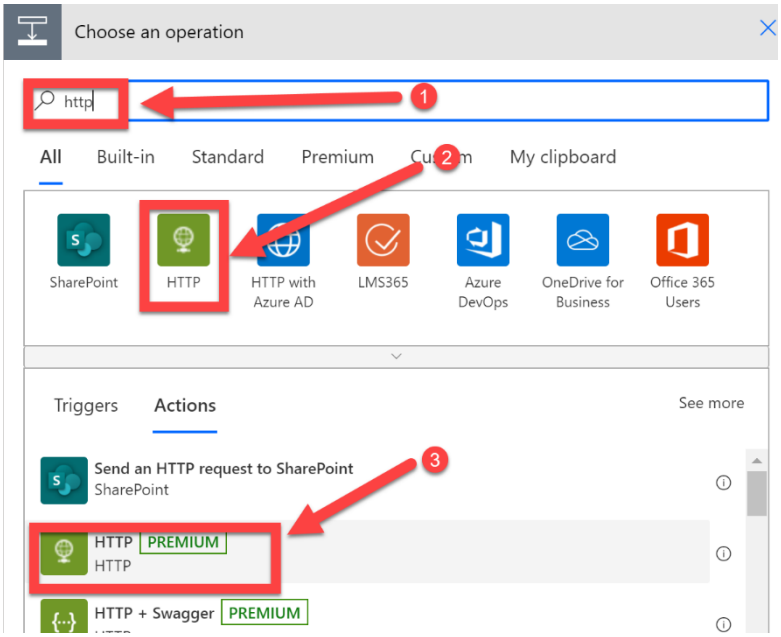
11) For Content select **Current Item** from the Dynamic Content selector. You will then need to enter the following into the schema, so that it is able to parse the information and allow us to use it later in this Cloud Flow. Enter the following:

```
{
  "type": "object",
  "properties": {
    "tid": {
      "type": "string"
    },
    "customername": {
      "type": "string"
    }
  }
}
```

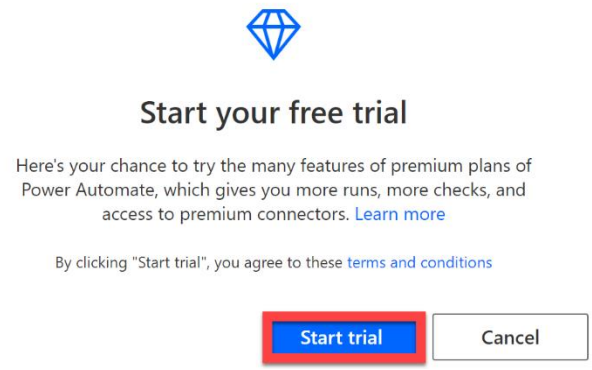


We now need to get an access token from Azure Active Directory, under the context of the app you registered, and your customers consented to, in steps 1 and 2. To do this, we will use the HTTP action to authenticate against Azure AD.

12) Click **Add an action (<u>still inside the "Apply to each" box</u>)**
13) Search for **HTTP**, select **HTTP**, click **HTTP**



You **may** be asked at this point to start a trial, as the HTTP action requires a premium license, click **Start trial**

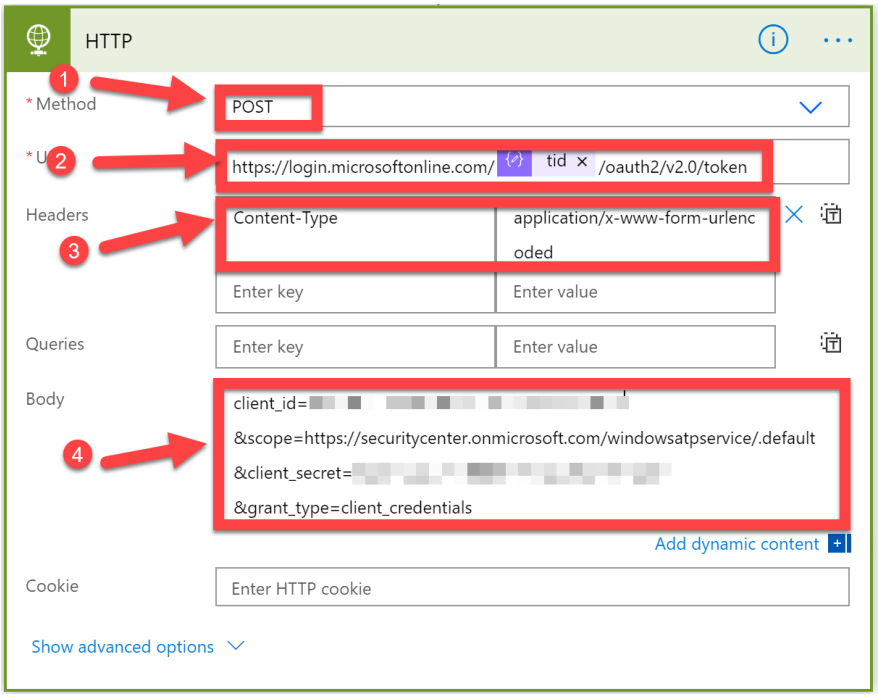


14) Select **Post** as the Method type,
15) In the URI enter **https://login.microsoftonline.com/** then select the **tid** from the Dynamic Content selector, under the Parse JSON step, then after tid type **/oauth2/v2.0/token**

16) In the headers section, enter a key type of **Content-Type** and a value for that key of **application/x-www-form-urlencoded**

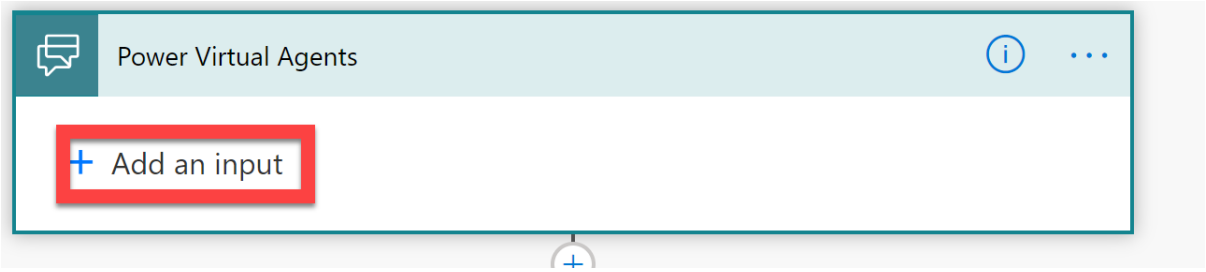
17) In the body, enter the following (replacing the ***CLIENTID*** *and* ***APPSECRET*** values with your own app registrations information that you capture in step 1 of this lab):

client_id=CLIENTID
&scope=https://securitycenter.onmicrosoft.com/windowsatpservice/.default
&client_secret=APPSECRET
&grant_type=client_credentials

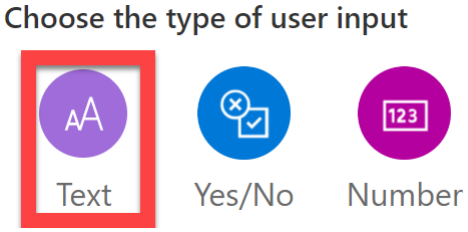


This HTTP action will authenticate against Azure AD using your apps ClientID and secret, and will get an Access Token, for the Defender for Endpoint service. You can then use this when you make calls to the Defender for Endpoint APIs to authenticate and be granted access to Alert and Score information

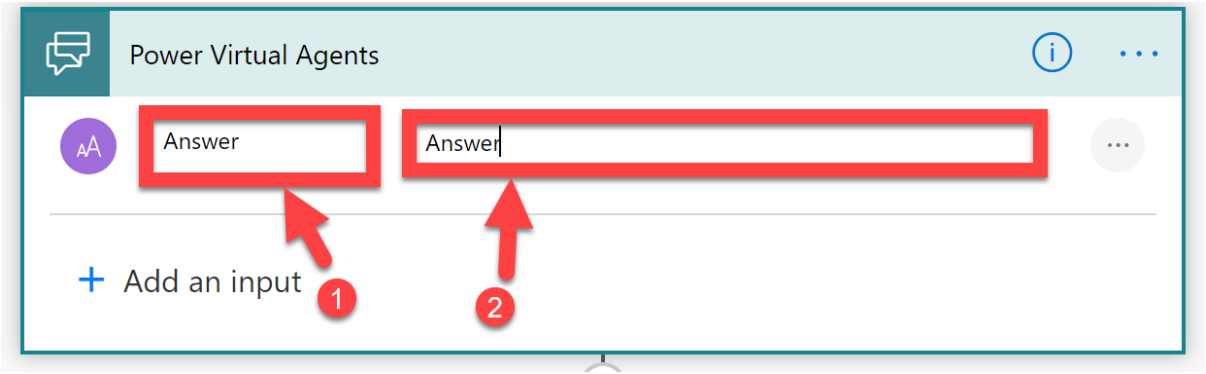
18) It's now time for us to perform an initial test of the steps you have completed so far. But before we do that, we need to make sure that the bot can pass a variable to the Cloud Flow. At the beginning of your Cloud Flow, expand the Power Virtual Agents action and select **+ Add an input**



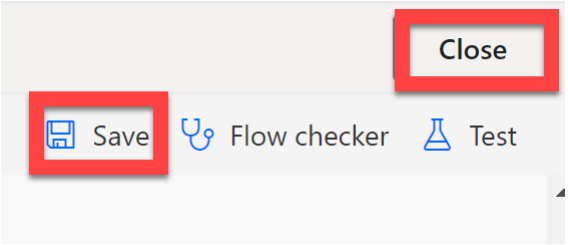
19) Select **Text**



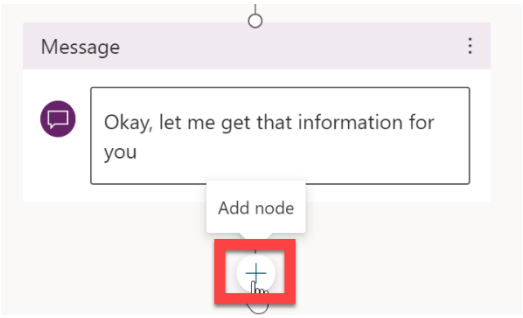
20) Enter **Answer** into both dialog boxes



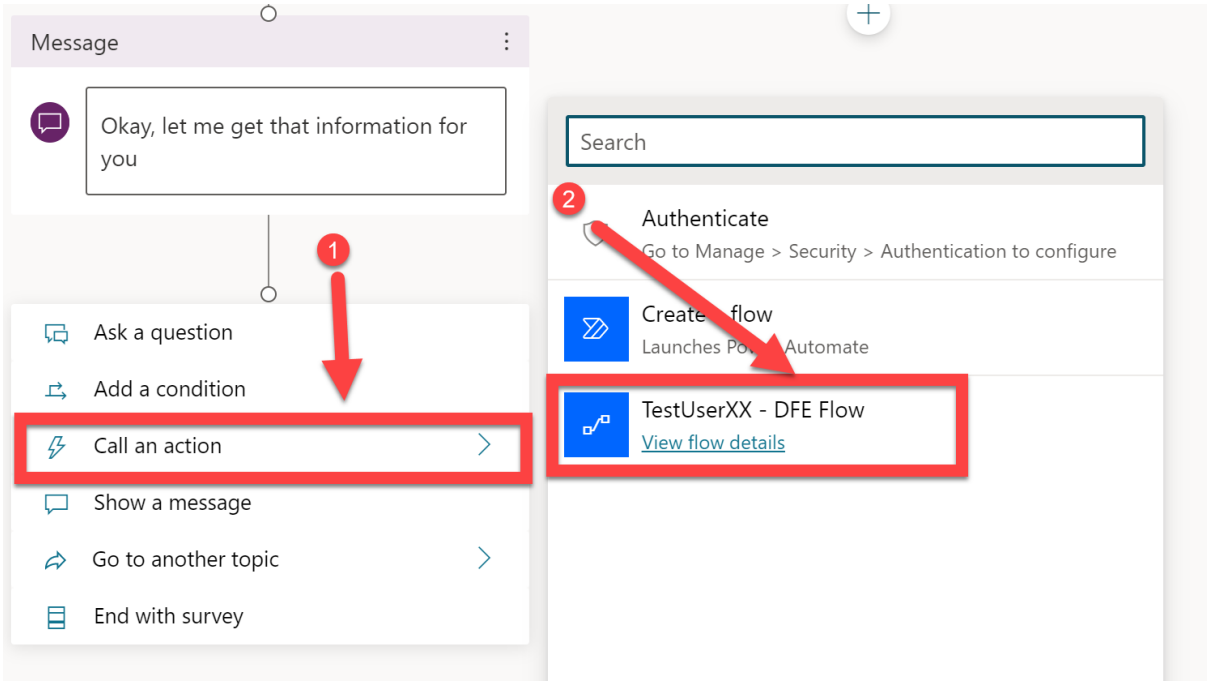
21) At the top right of the screen, select **Save**
22) Click **Close**



23) You should now have been returned to the Topics configuration editor. If not, navigate back to the Defender for Endpoint Topic. Under the Message you previously created, click the **+** button

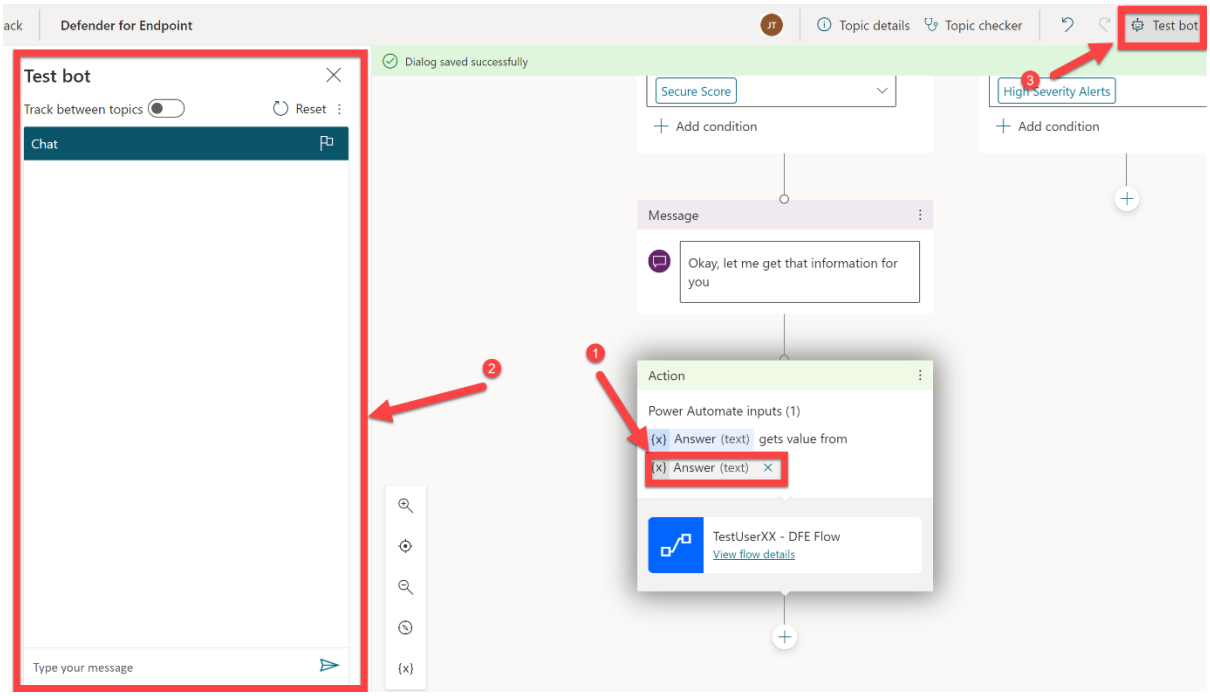


24) Select **Call and action**, click on the **Cloud Flow that your just created** in this example, TestUserXX – DFE Flow

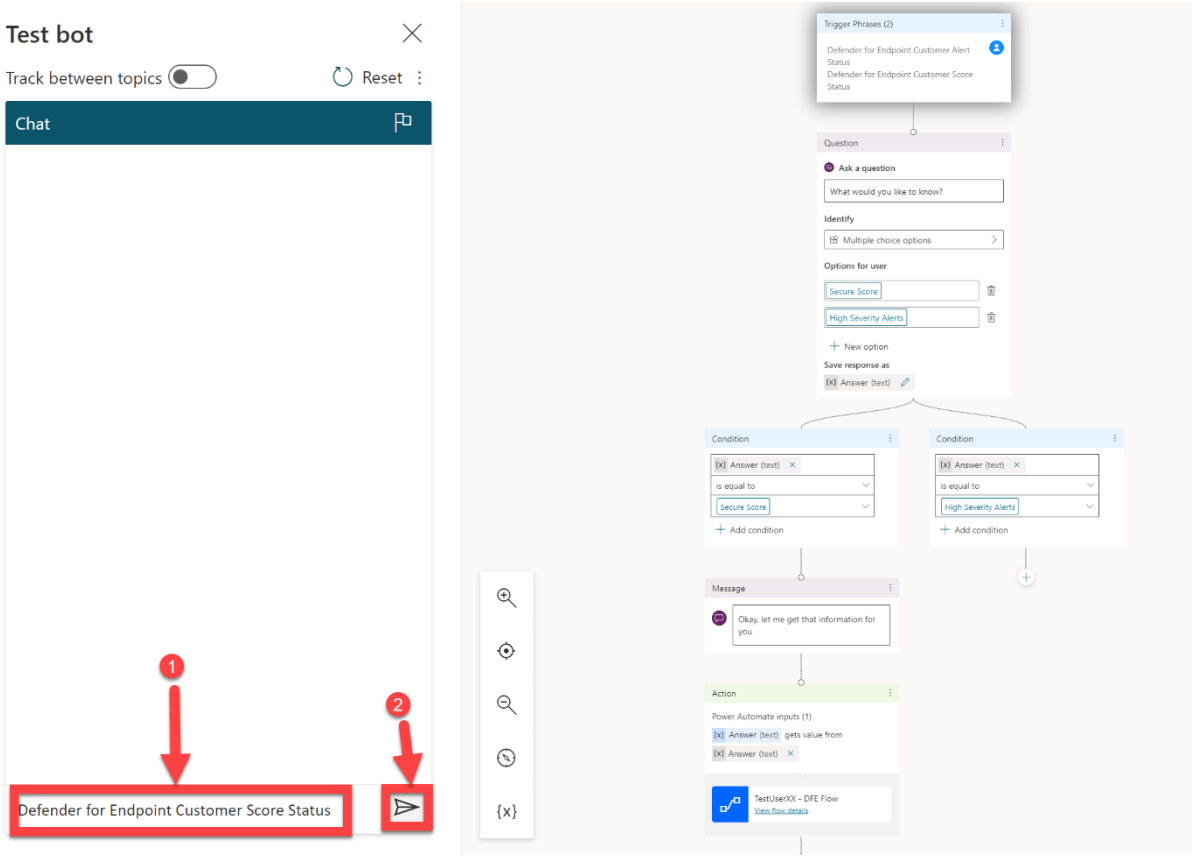


25) Select **Answer** as the value.

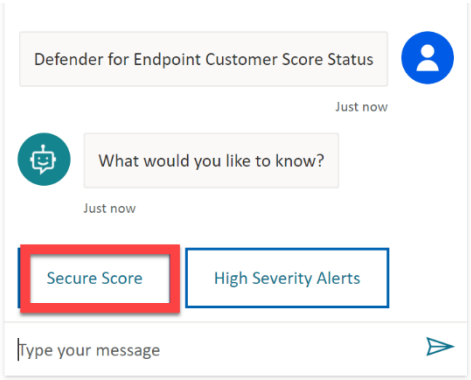
26) We now need to test the Cloud Flow you have created. To do this, we need to trigger the flow by having a conversation with the bot. If you can see the Test Bot component, move onto the next step, if not, click **Test Bot** and it should appear



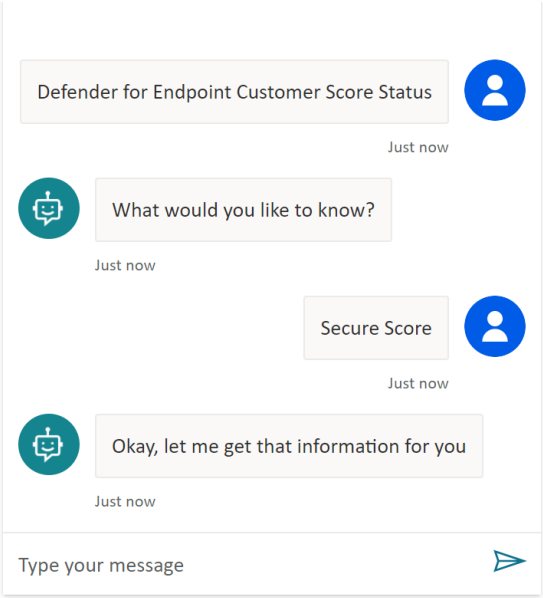
27) In the Type your message dialog box, enter **Defender for Endpoint Customer Score Status**, and click the **Submit** (paper-plane) button

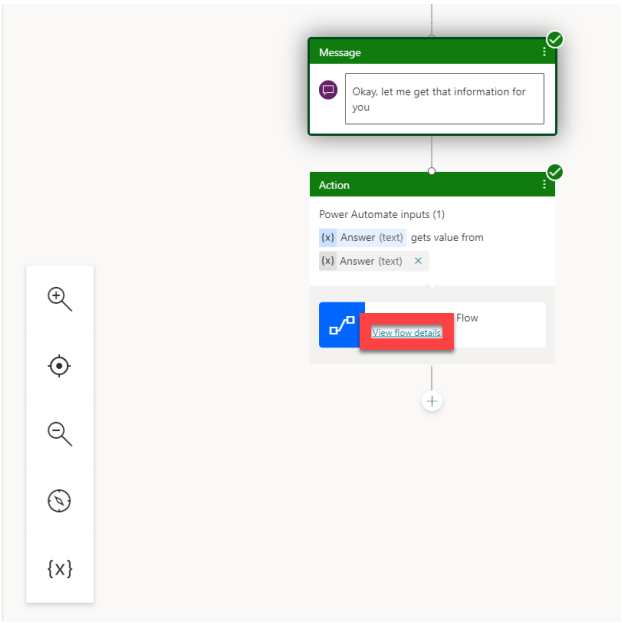


28) Select **Secure Score**

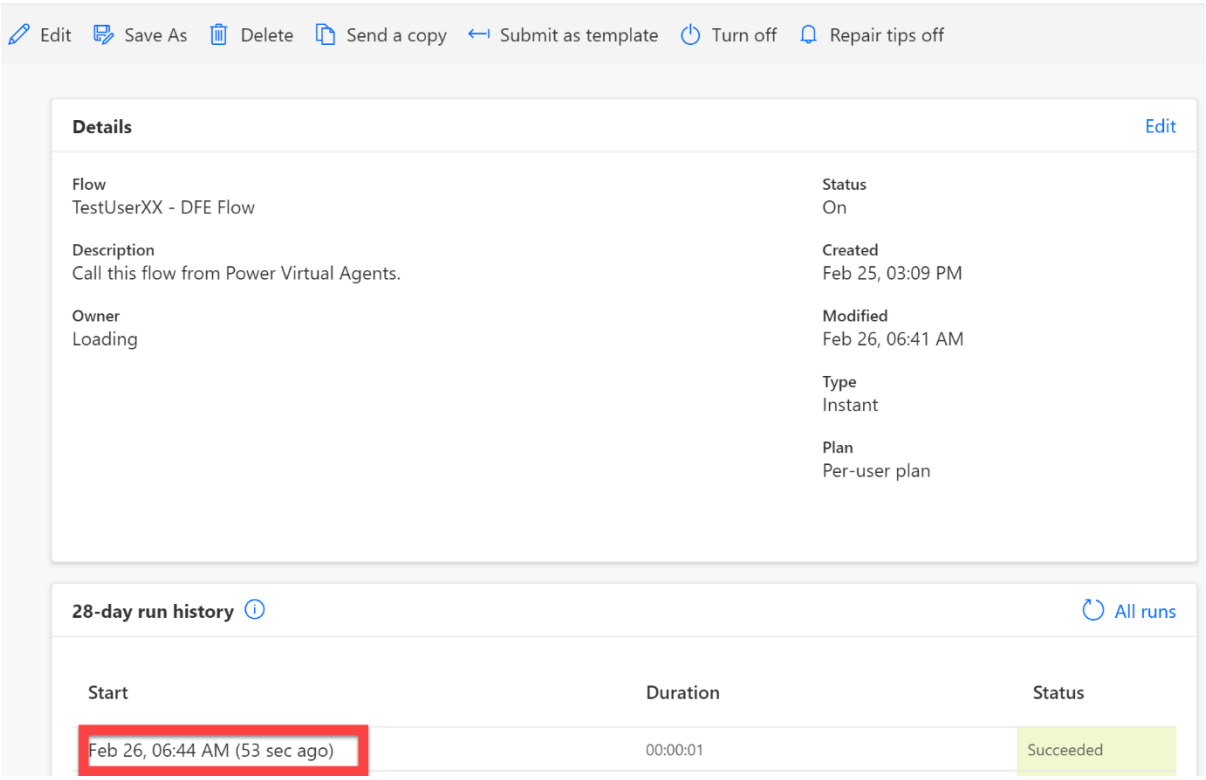


29) This will now trigger your Cloud Flow to run.
30) We now need to take a look at what is happening within that Cloud Flow. Select **View flow details**

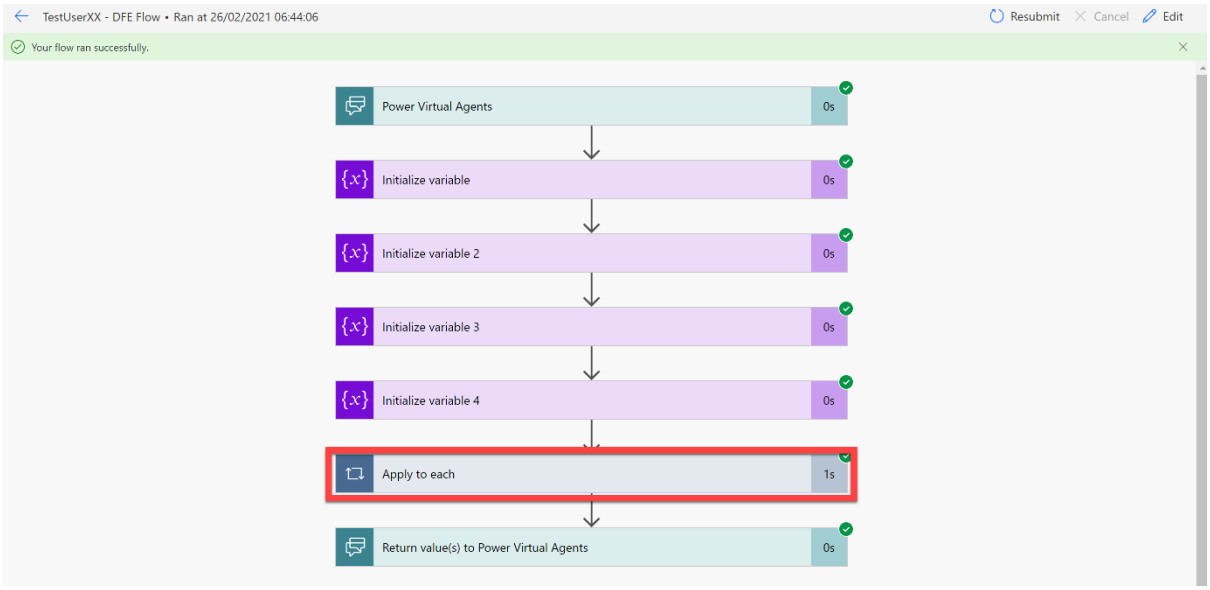




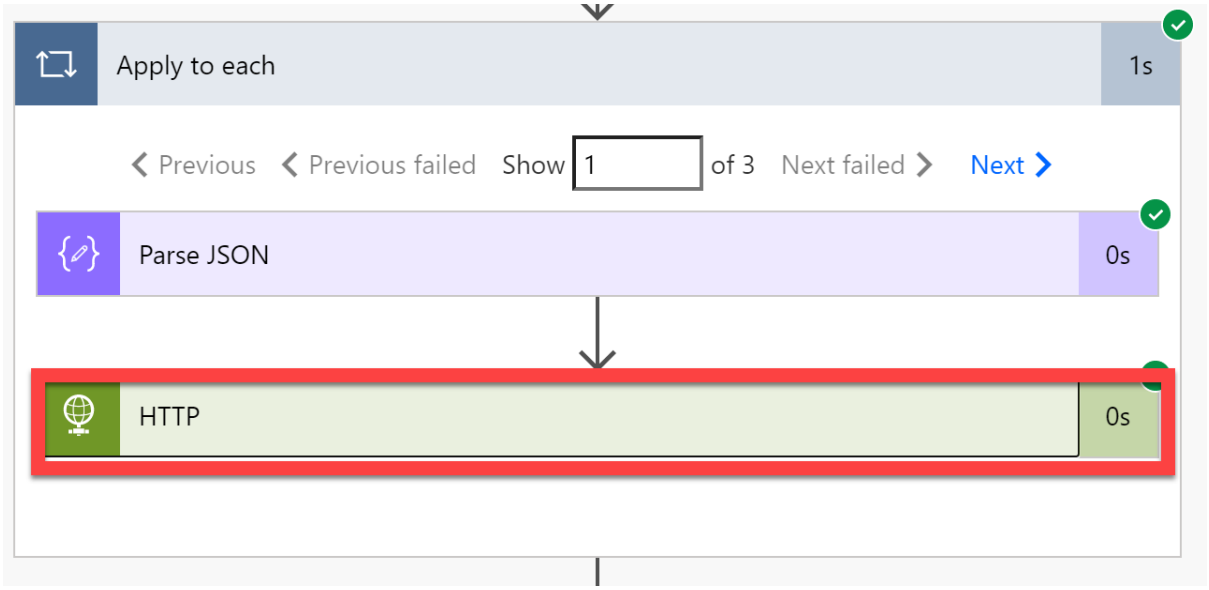
31) In the 28-day run history section, select **your most recent Cloud Flow run**



32) In this example, we can see that the Cloud Flow ran successfully (as expected!). **Click on the Date**
33) We need to know more information about the Apply to Each action, and the sub-actions contained within it.
34) Select **Apply to each**

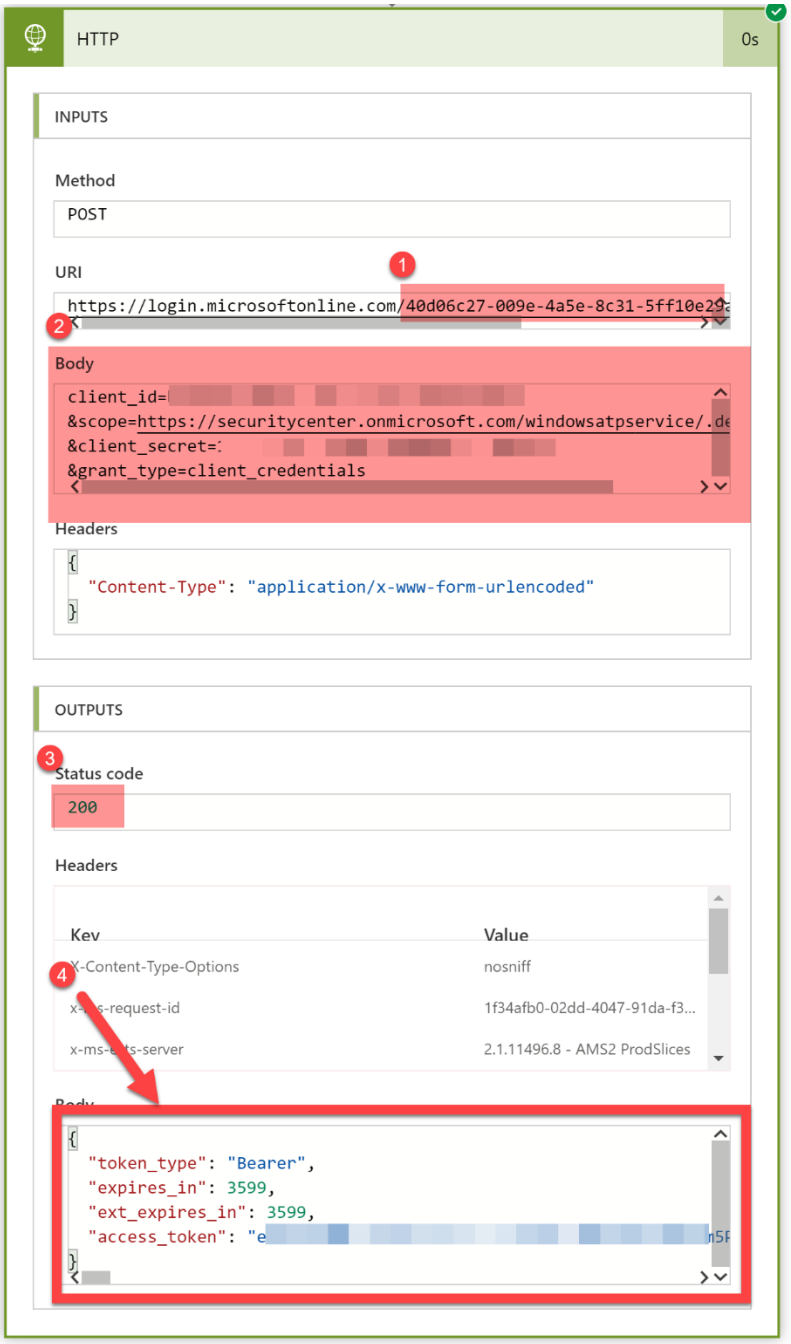


35) After the Apply to each action has been expanded, we can see the actions that are contained within it. Select **HTTP** to view more details *(note: from within an Apply to each step, you can review every time the action ran, and review the results from the individual items. Just select the **Next >** button to view more information about a different item)*

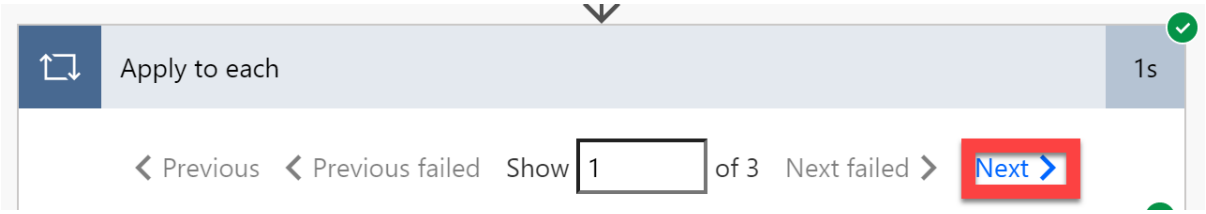


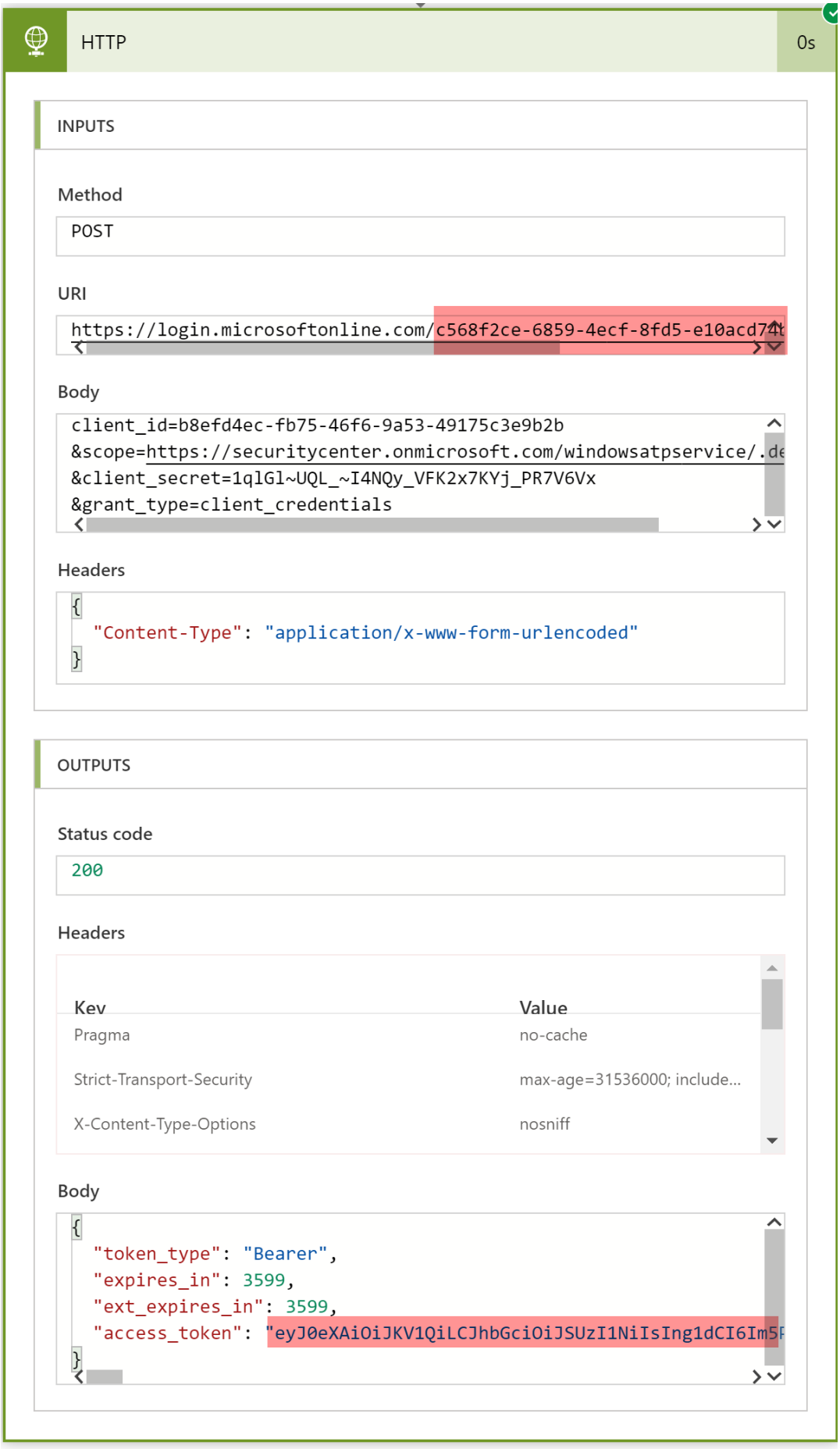
There is a lot of useful information inside of this action. Firstly, if you look at the input section, under URI you will see that the request has been made to login.microsoftonline.com, but then your customer's Tenant ID has been used to customise the request URI. In the body section you can see the request body you built inside of your Cloud Flow. Looking into the Output section (this is what you received from Azure AD as a result of this request), you can see the HTTP status code is 200 (which is expected) and that the Body of the Output response contains an access token

36) **Copy everything in the "Outputs -> Body" (4) to a notepad as we will need this shortly.**

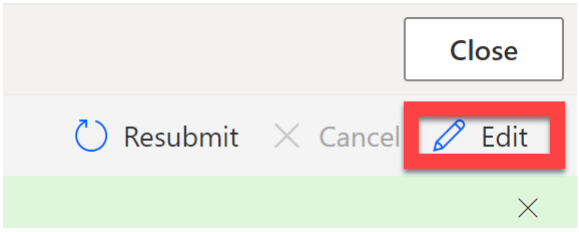


37) If you then click the **Next >** button, and review the **HTTP** action, you will see that, in comparison to the preview step, a different URI has been used, and a different Access Token has been provided by Azure AD in the Output response

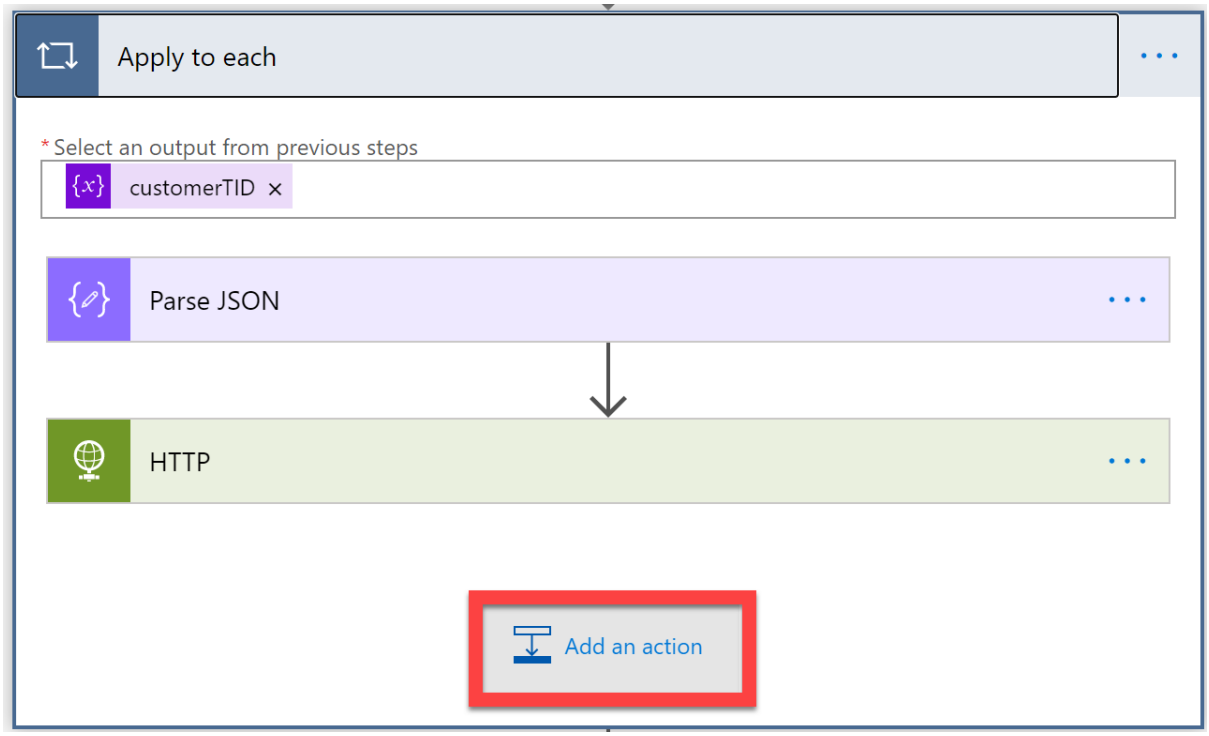



HTTP
0s
INPUTS
Method
POST
URI
https://login.microsoftonline.com/c568f2ce-6859-4ecf-8fd5-e10acd74
Body
client_id=b8efd4ec-fb75-46f6-9a53-49175c3e9b2b
&scope=https://securitycenter.onmicrosoft.com/windowsatpservice/.de
&client_secret=1qlGl~UQL_~I4NQy_VFK2x7KYj_PR7V6Vx
&grant_type=client_credentials
Headers
{
"Content-Type": "application/x-www-form-urlencoded"
}
OUTPUTS
Status code
200
Headers
Key
Value
Pragma
no-cache
Strict-Transport-Security
max-age=31536000; include...
X-Content-Type-Options
nosniff
Body
{
"token_type": "Bearer",
"expires_in": 3599,
"ext_expires_in": 3599,
"access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Im5
}

38) Now we need to complete the rest of the flow, so that it uses the access token to request information from the Defender for Endpoint API Endpoints.
39) Click **Edit**

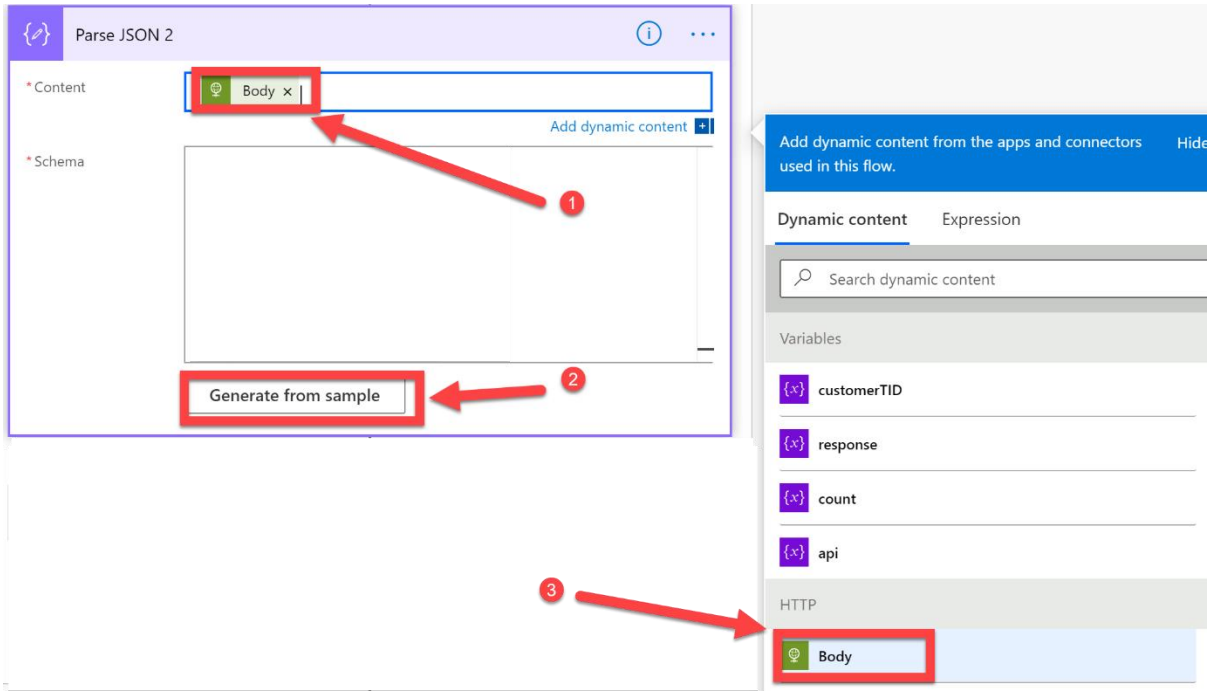


40) Under the Apply to each action, select **Add an action**



We need to get the access token from the previous HTTP call, to use in our subsequent HTTP actions when we request information from the Defender for Endpoint API Endpoints. To do this, we need to use the Parse JSON step to process the response Body from the HTTP action, so that the Access Token is available as **dynamic content**

41) Select **Parse JSON**
42) In the Content section, select the **Body** of the HTTP action, then click **Generate from sample**

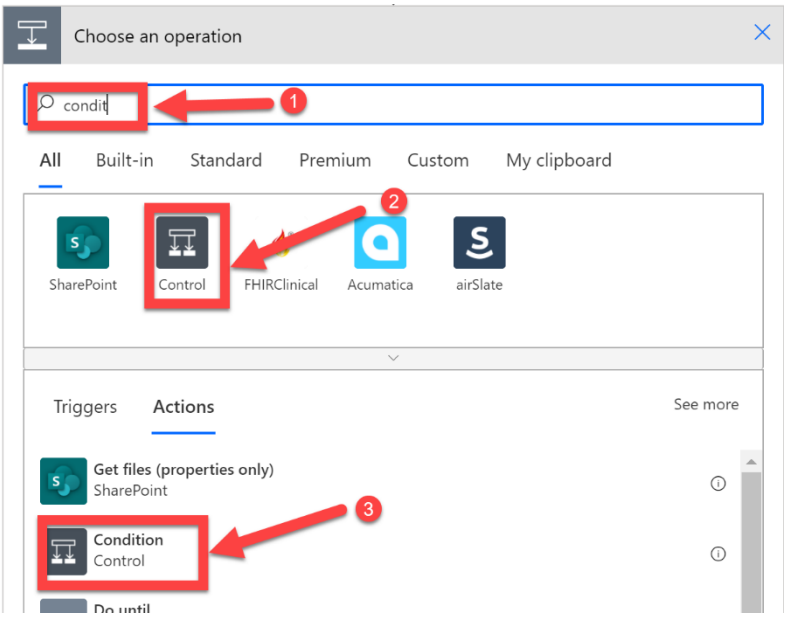


43) **Paste in the output that you captured in section 4**. Click **Done**

44) Once completed, the Parse JSON step should look like this

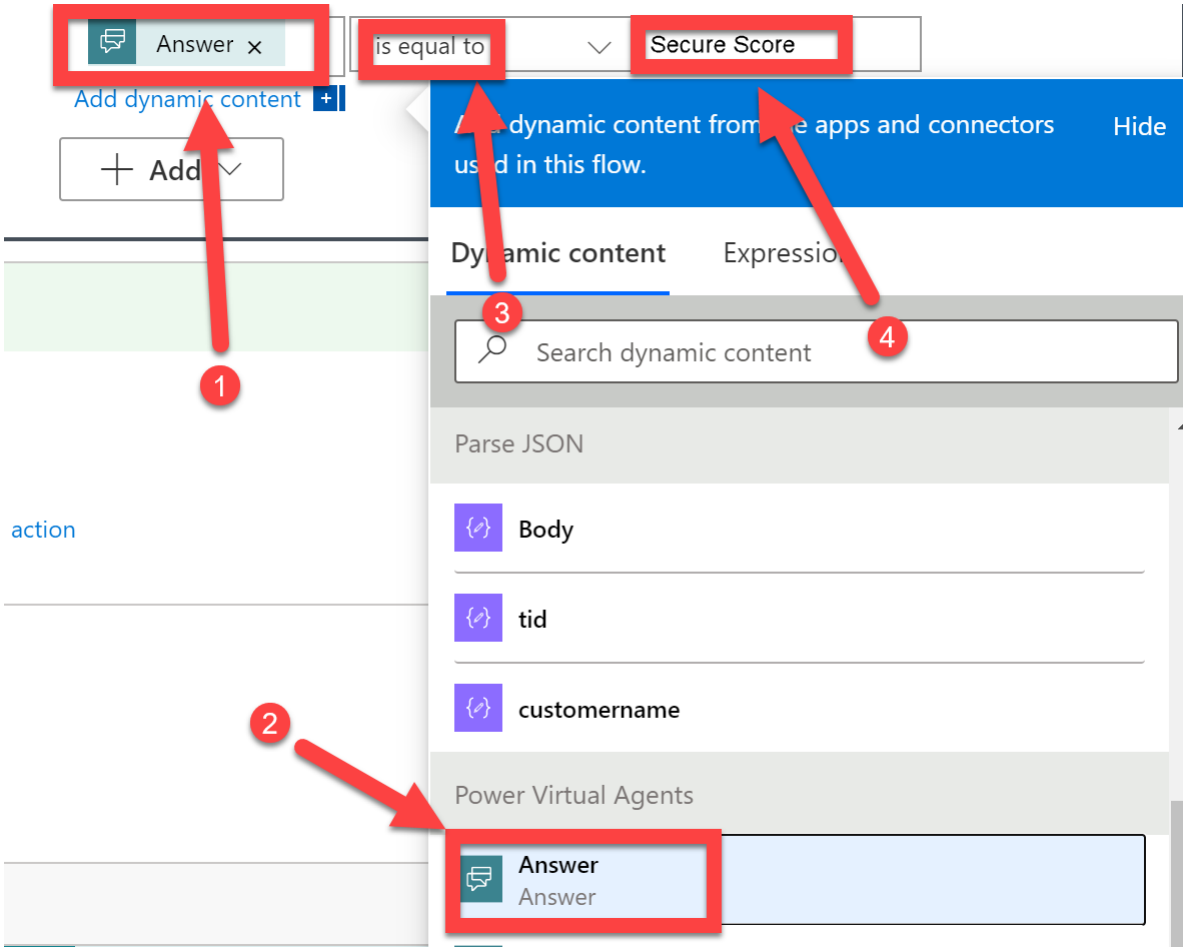
45) Directly underneath select **Add an action**

46) Add a **Condition**

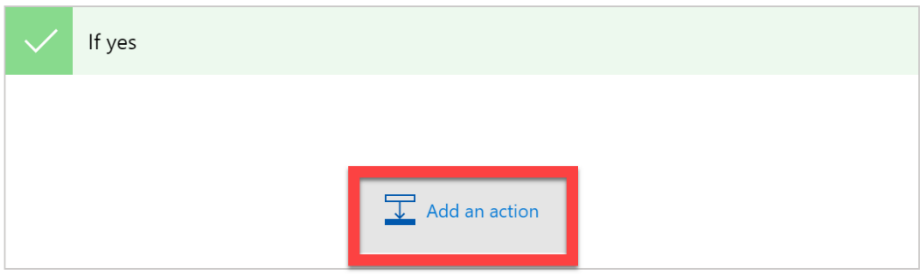


This condition will be used to control what actions/steps take place, depending on if the end-user want to know information about Secure Score or High Severity Alerts. For this, we will use the answer variable that is passed to us, from Power Virtual Agents

47) Select **Answer** from the dynamic content list as the item to be evaluated. Set the operator to **is equal to**, set the value to **Secure Score (corresponding to the value you set in your Bot choice box)**



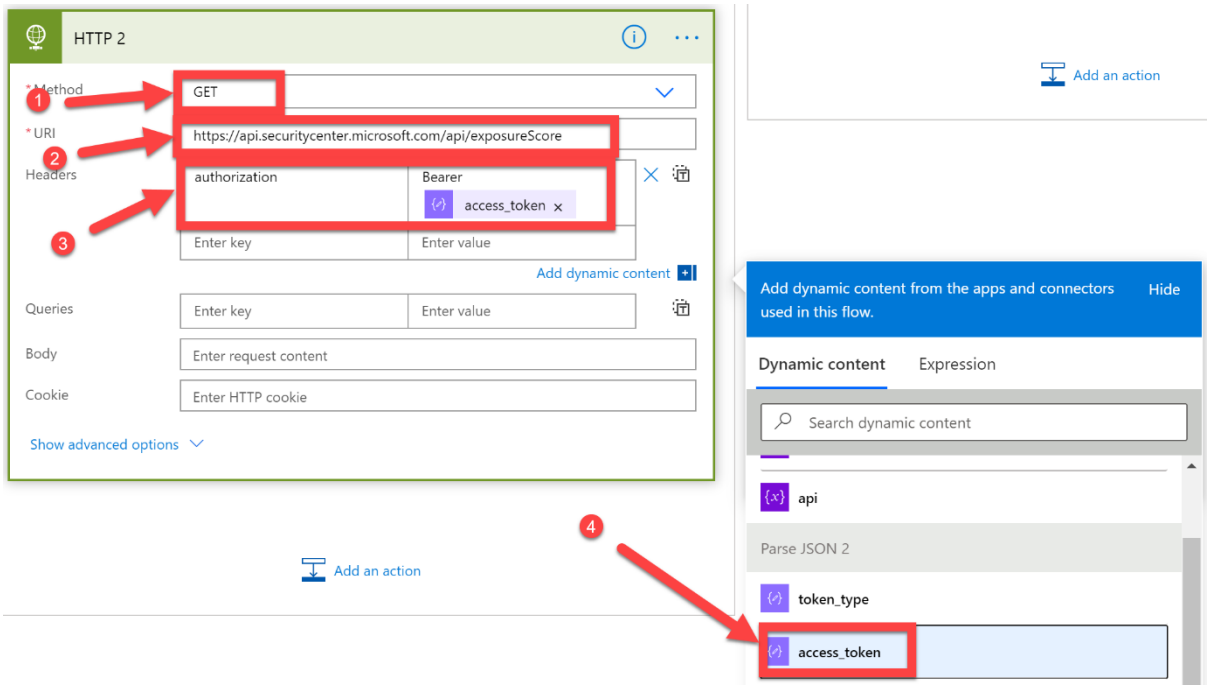
48) Below the If Yes condition, select **Add an action**



We have the access token available in the Dynamic Content selector, we will now need to create a HTTP action that references the access token for authentication, when requesting information about a customer Secure Score from the Defender for Endpoint API Endpoints.

- **Add a HTTP action**, select the **GET** method
- In the URI enter **https://api.securitycenter.microsoft.com/api/exposureScore** ,
- In the Headers, enter **Authorization** as the key, and then for the value enter **Bearer** ***ACCESSTOKEN*** , replacing ***ACCESSTOKEN*** with the **access_token** dynamic content

from your previous Parse JSON step – make sure there is a space between Bearer and Access_Token!



This action will get the Defender for Endpoint exposure score for your customers, and respond back to the Cloud Flow with the information, in the same way the previous HTTP action was able to retrieve an access token from Azure AD.

Now that we have this information, **we need to use another Parse JSON action**, so that the result of the API call can be referenced in the dynamic content section

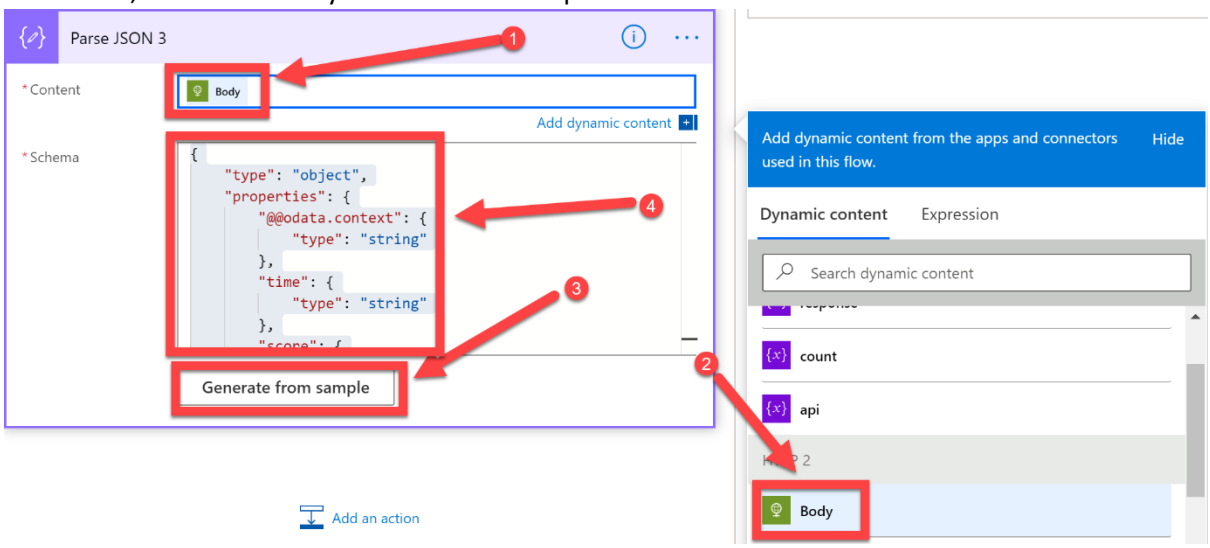
49) Select **Add an Action** and **Parse JSON**.
50) In **Content box** select **Body** from the last HTTP, i.e. HTTP 2

Make sure that you reference the correct **Body** in the Content section (it will more than likely be from **HTTP 2** if you have been following this guide)

51) Select **Generate from sample**, and enter the following:

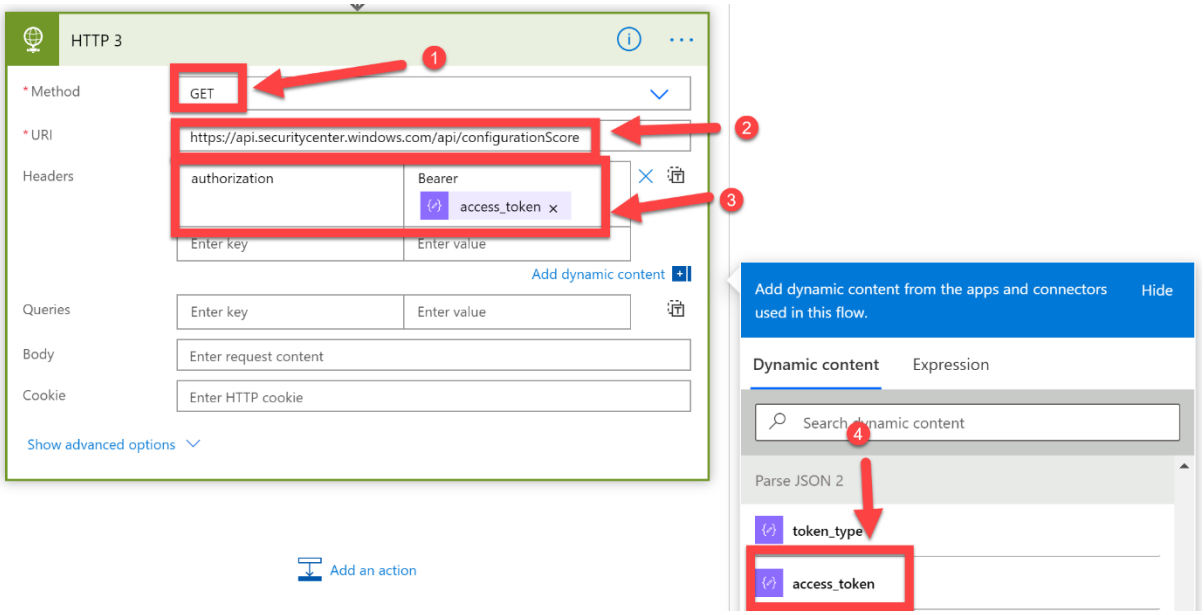
```
{
  "@odata.context":
"https://api.securitycenter.microsoft.com/api/$metadata#ExposureScore/$entity",
  "time": "2021-02-26T07:37:40.8520763Z",
  "score": 32.9,
  "rbacGroupName": null,
  "rbacGroupId": null
}
```

Click **Done**, and check that your Parse JSON step looks like the below:



We now have all the information we need regarding the Exposure Score, but in Defender for Endpoint there is an additional Score, the Configuration Score. In this lab, we want to report both scores back to the end-user via the Power Virtual Agents Bot

52) Add a **HTTP** action after the previous Parse JSON action.
53) For Method select **GET**,
54) In the URI enter **https://api.securitycenter.windows.com/api/configurationScore** ,
55) In the Headers, enter **Authorization** as the key, and then for the value enter **Bearer *ACCESSTOKEN*** , replacing ***ACCESSTOKEN*** with the **access_token** dynamic content from you're the relevant Parse JSON step
56) It should look like this once completed:

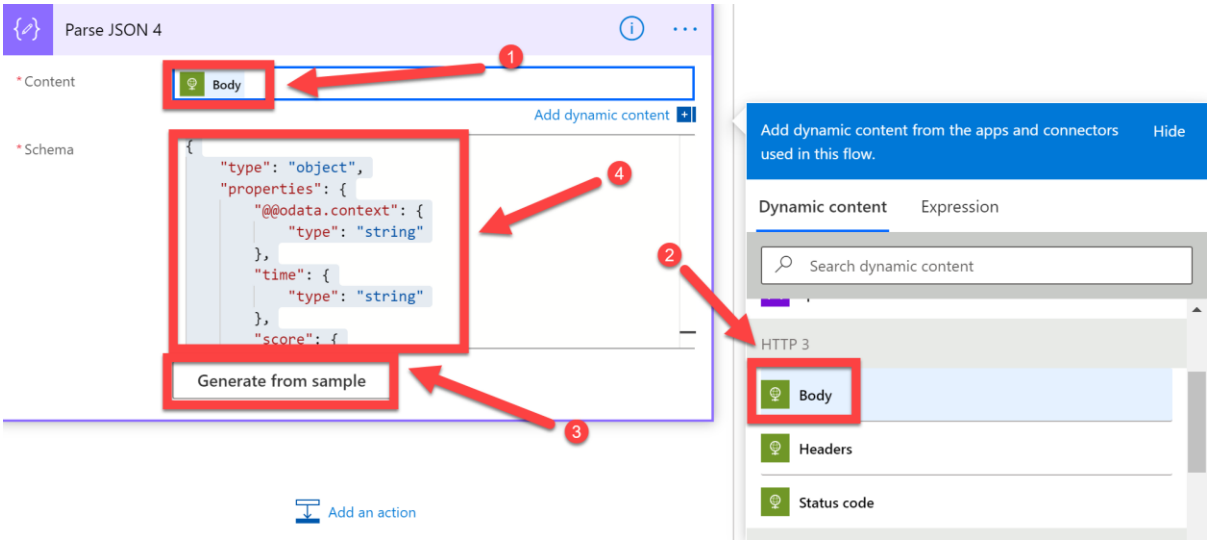


57) We then need to follow this HTTP action with another (!!!) Parse JSON action, to allow us to use the Configure Score in the Dynamic Content section.

58) You will need to reference the **Body** of the previous HTTP action, from the Dynamic Content selector – in our lab, this is from **HTTP 3**, make sure your selection matches what is relevant to your Cloud Flow. Select **Generate from sample** and enter the following:

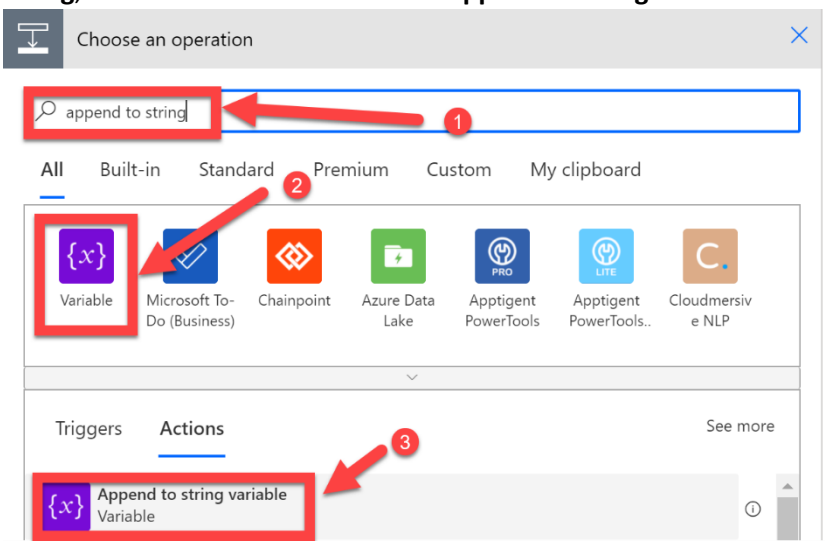
```
{
  "@odata.context":
"https://api.securitycenter.windows.com/api/$metadata#ConfigurationScore/$e
ntity",
  "time": "2021-02-26T07:37:42.0774883Z",
  "score": 349,
  "rbacGroupName": null,
  "rbacGroupId": null
}
```

Click **Done**, once complete it should look like this:

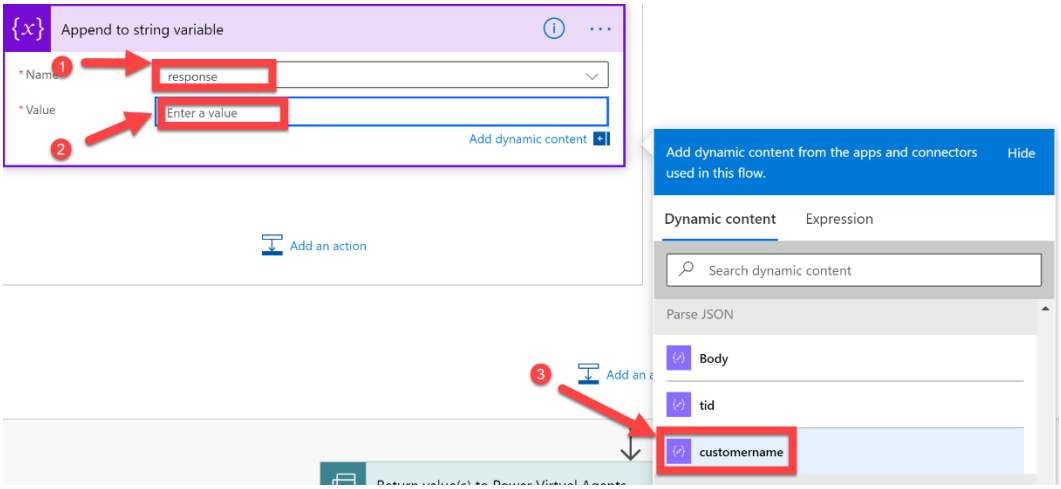


Now that we have all the information required about the customer's Defender for Endpoint Secure and Configuration Scores, we now need to append that to the response string variable that you initialised at the beginning of this section of the lab.

59) To do this, select **Add an action** after the previous Parse JSON action, search for **Append to String**, select **Variable** and then click **Append to string variable**



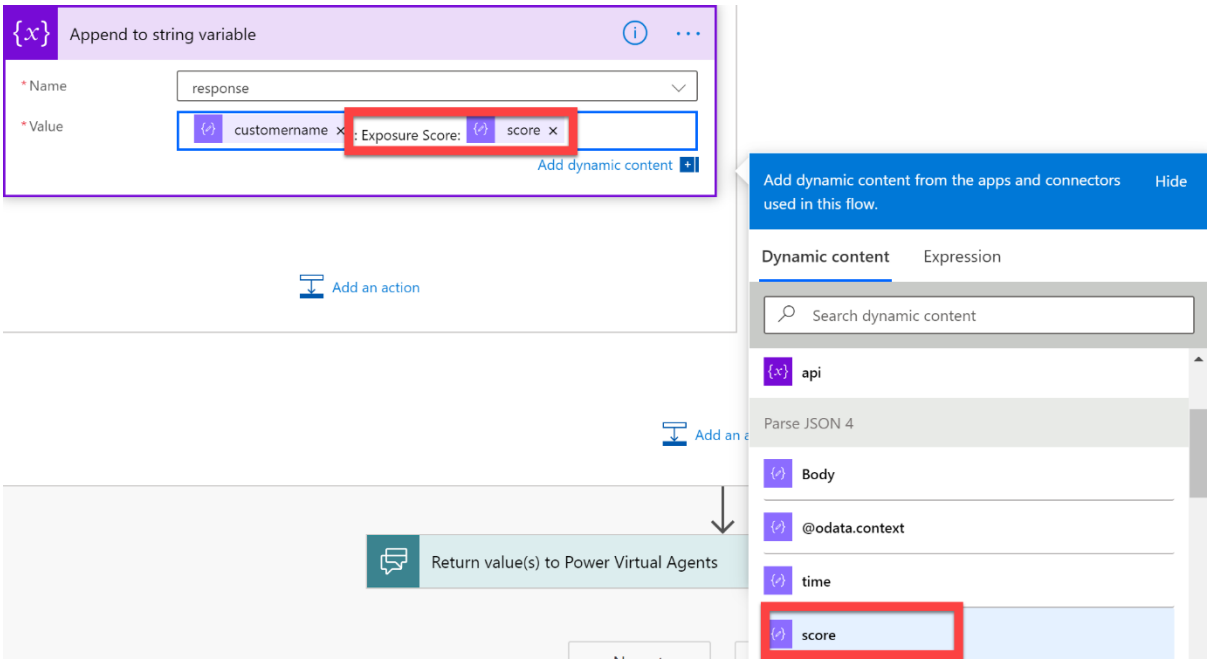
60) Select the **response** string variable, select the Value dialog box and select **customername** from the first Parse JSON action



61) After the customername, enter:

: Exposure Score:

select the **Score** from the most recent Parse JSON step



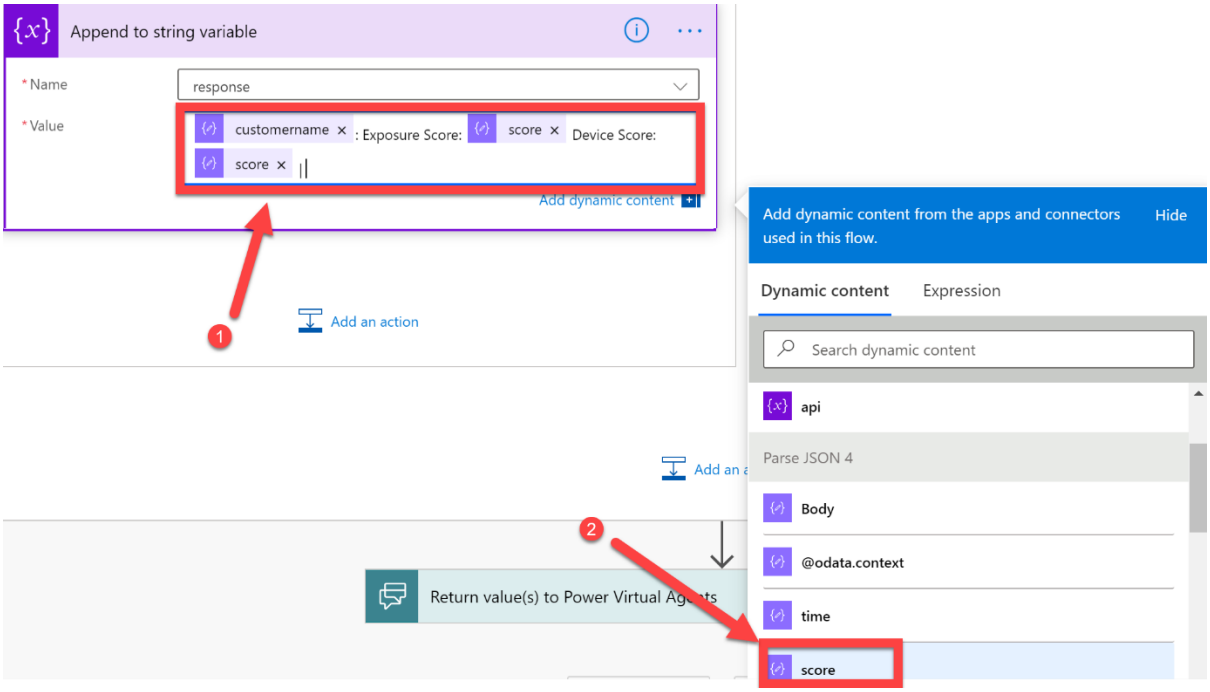
62) After the score, enter:

Device Score:

Select the **Score** from the Parse JSON step that is associated with the Device (configuration) Score (in our lab Parse JSON 3)

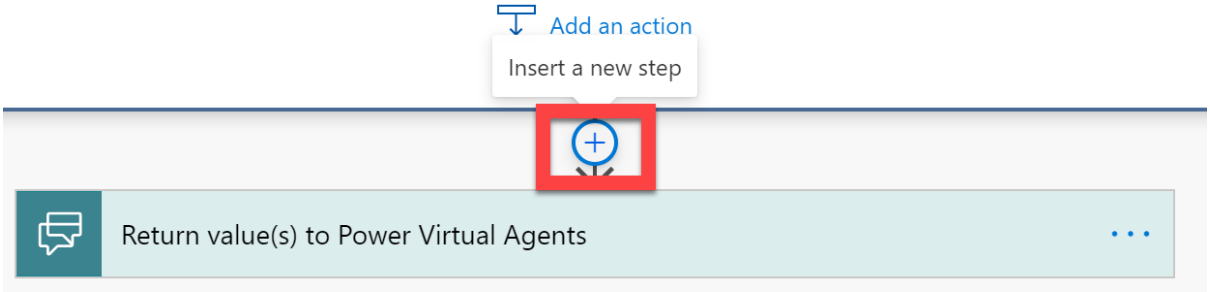
Add a delimiter, in my example I am using:

|

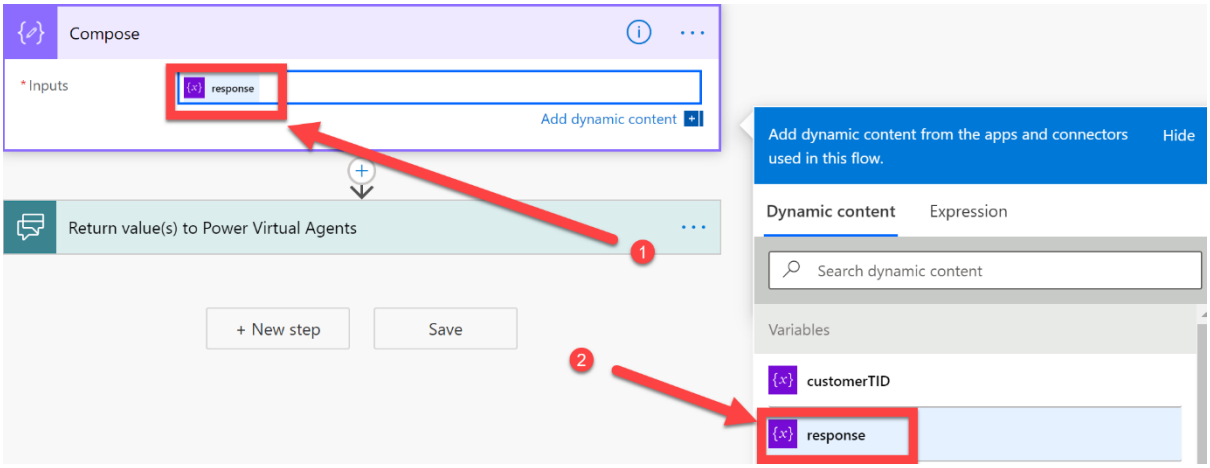


The steps in this section of the lab will repeat in sequence for each customer you have defined in the customerTID array. This will result in the response string variable being populated with the Device and Exposure scores for all customers, in a text string format. We can now send this back to the Power Virtual Agents Bot to allow it to send it onto the end-user. Before doing that though, it is probably wise to test the Cloud Flow now that we have made some significant changes to it.

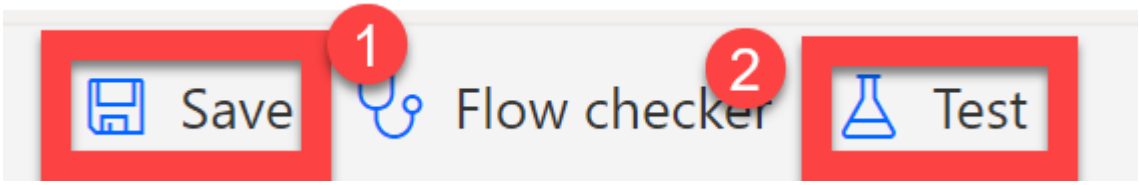
63) With that in mind, let's add a **compose** step, above the final step, as it will allow us to review what is stored within the Response string. At the end of your cloud flow, select the **+** button (above the return value(s) to Power Virtual Agents action)



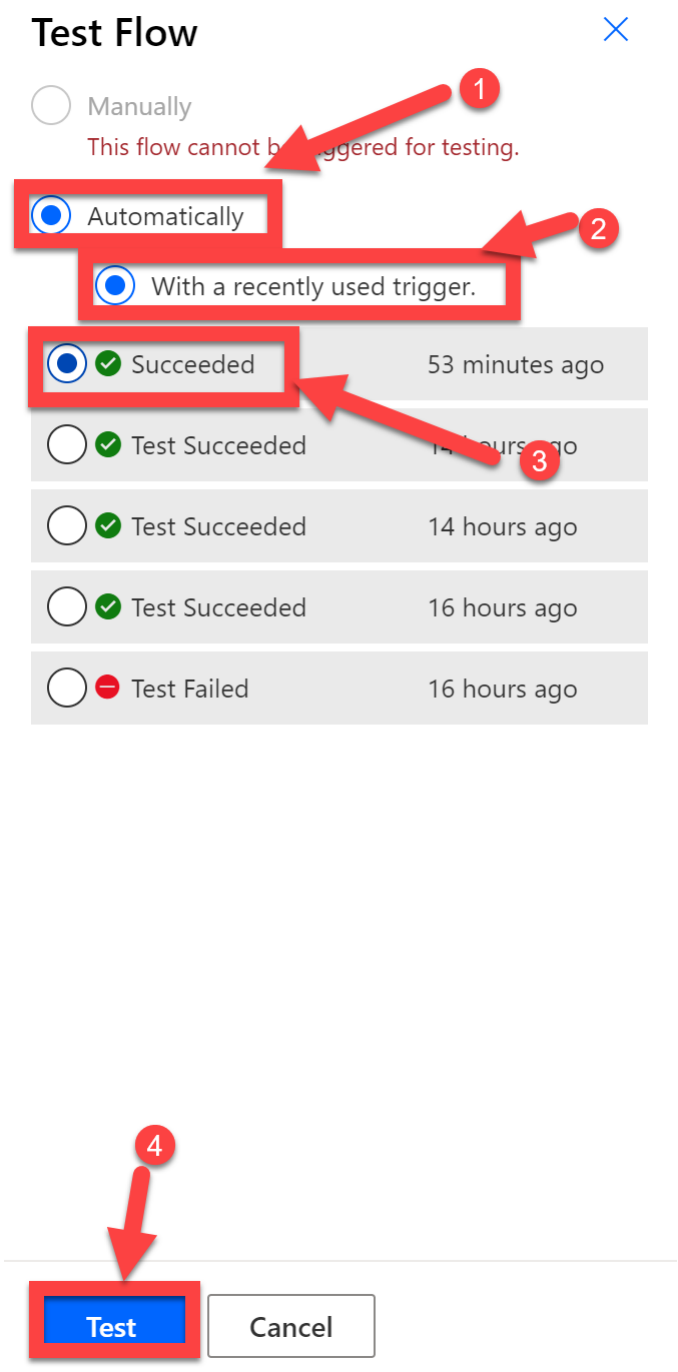
64) Add the **response** variable, from the Dynamic Content selector, into the Input dialog box



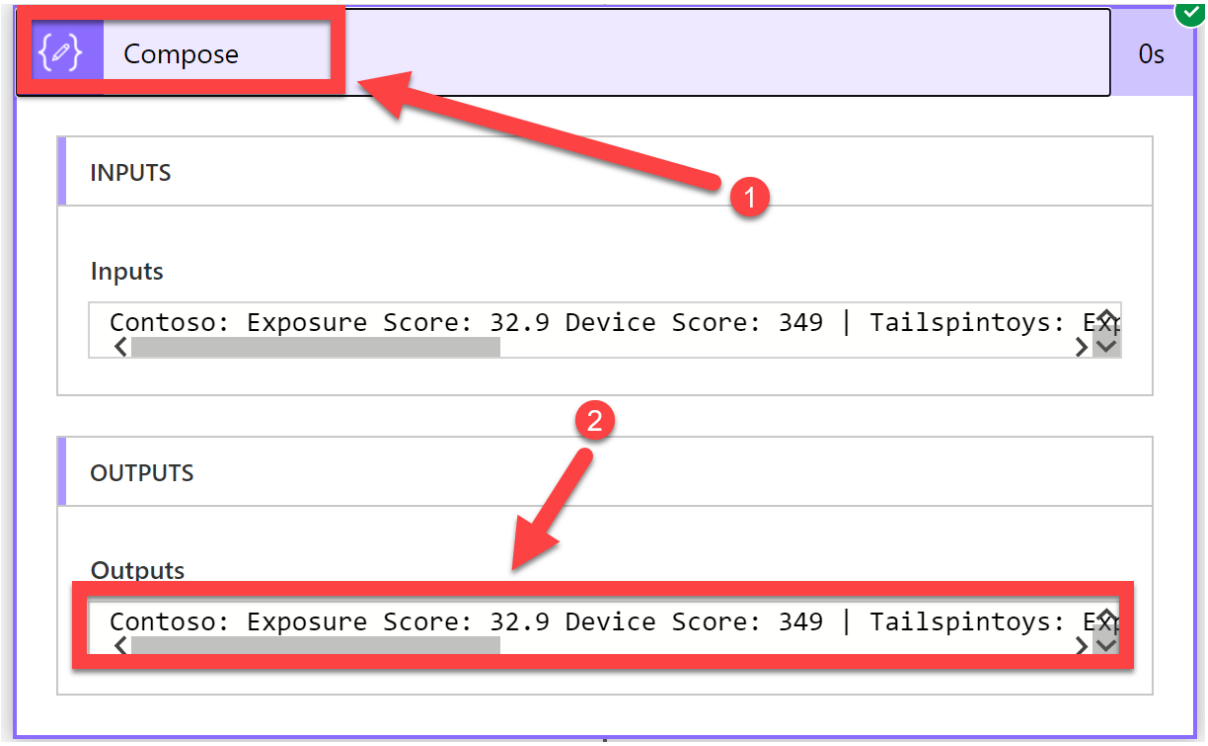
65) Select **Save** and then select **Test**



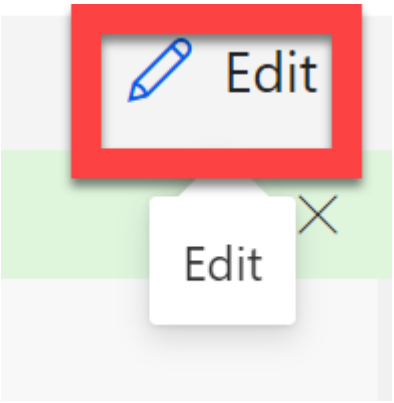
66) Select **Automatically**, select **With a recently used trigger**, select a **trigger** *(note: do not worry if the test failed, it should still initiate the Cloud Flow)*, click **Test**



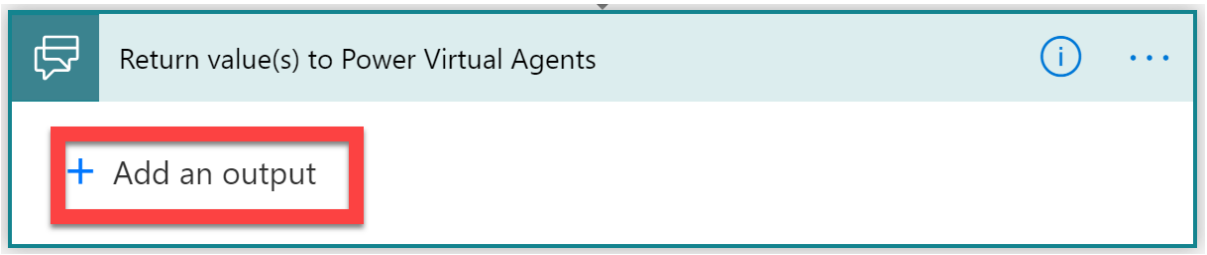
67) Once the Cloud Flow has successfully completed, scroll down to the end, select **Compose** and then review the **Outputs**. You should see that your Outputs list the Customer and then their associated Exposure and Device Scores.



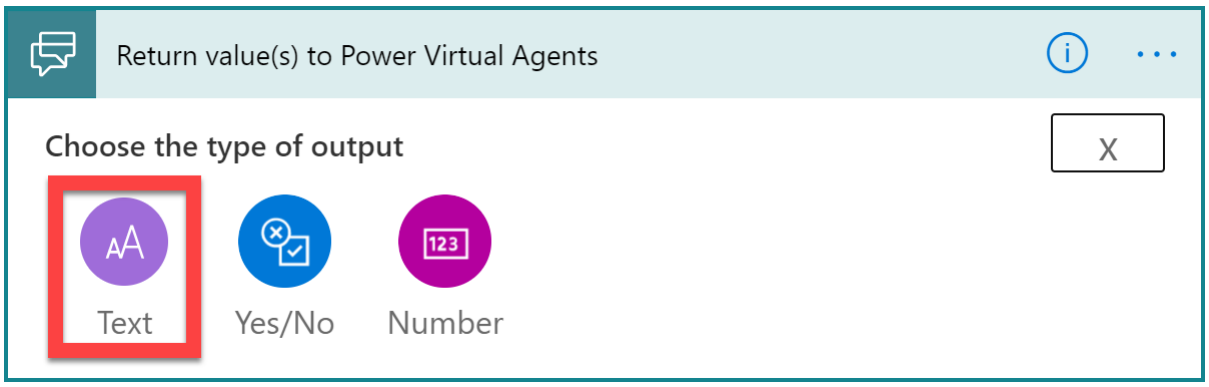
68) We now need to go back into the Cloud Flow editor to pass this information back to the Power Virtual Agents Bot. Click **Edit**



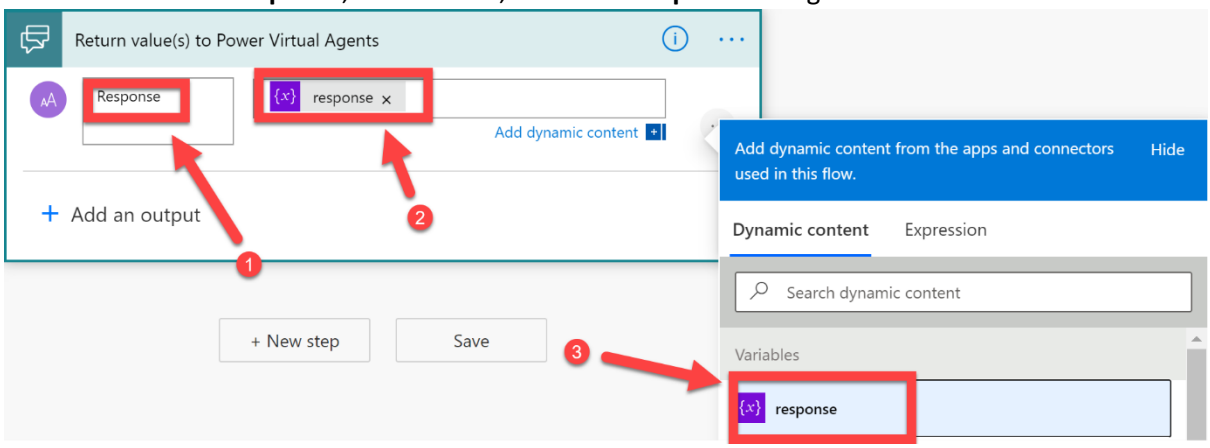
69) Scroll down to the end of your Cloud Flow and expand the **Return value(s) to Power Virtual Agents** action, select **+ Add an output**



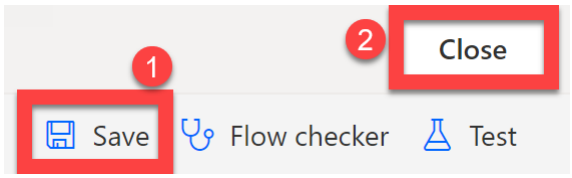
70) Select **Text**



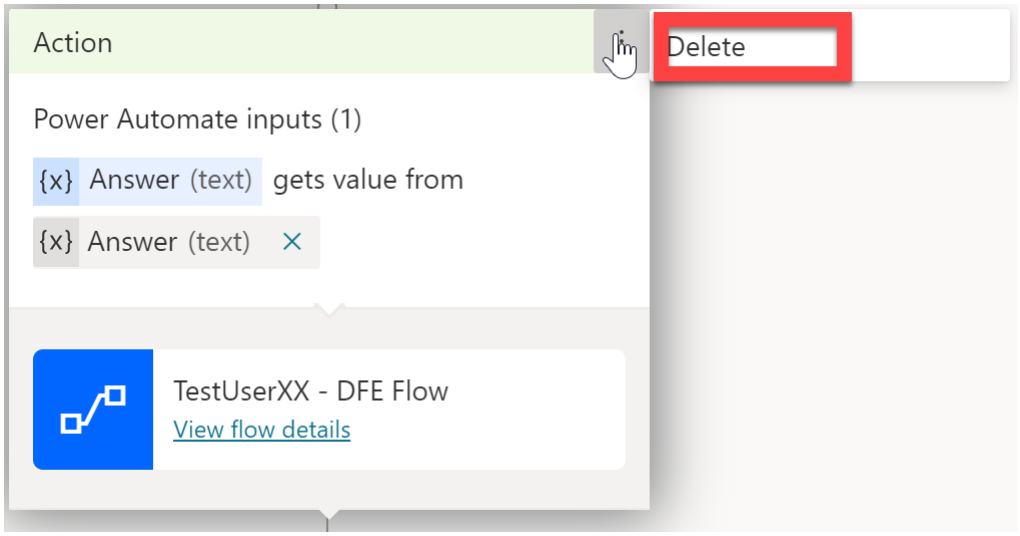
71) In the name enter **Response**, in the Value, select the **response** string variable



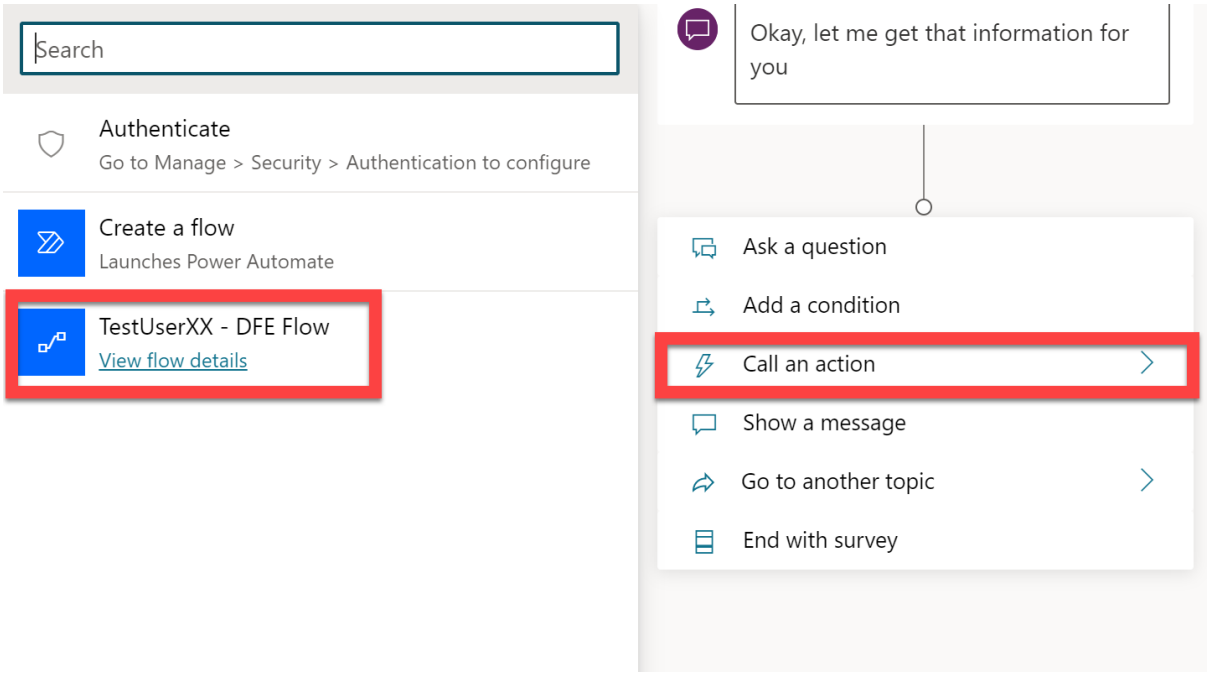
72) Select **Save**, and then select **Close**



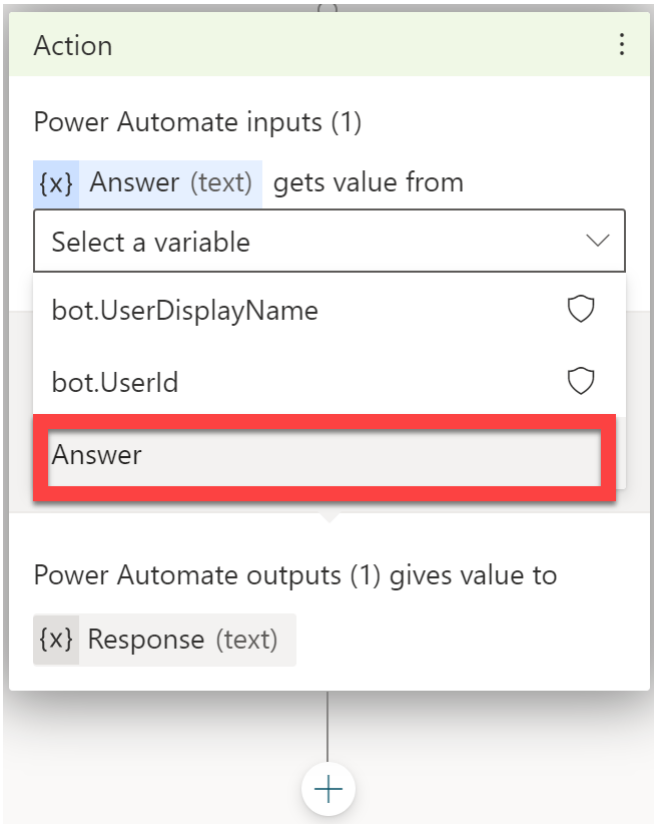
73) You should now be returned to the Power Virtual Agents Topic Authoring app. As we have made changes to the Output of the Cloud Flow, we need to remove the Action that was previously triggering the Cloud Flow, select the **…** and click **Delete**



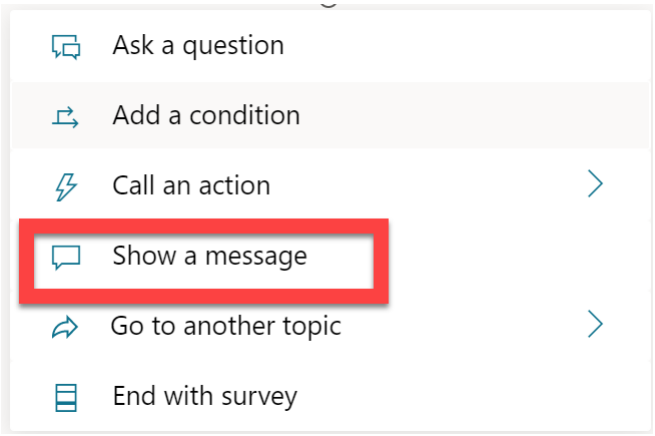
74) Under the message, add the Cloud Flow back into the Topic Flow



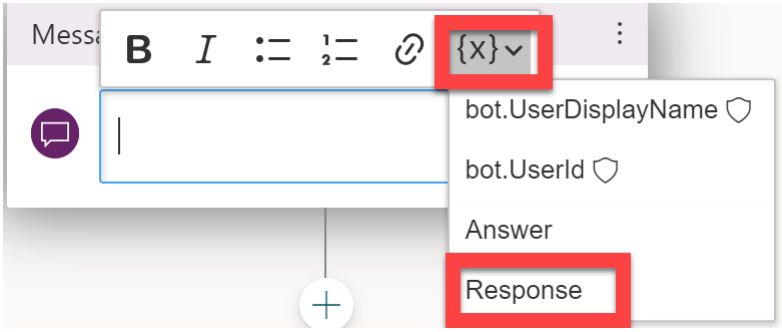
75) Similar to before, select **Answer** as the value for the Answer variable



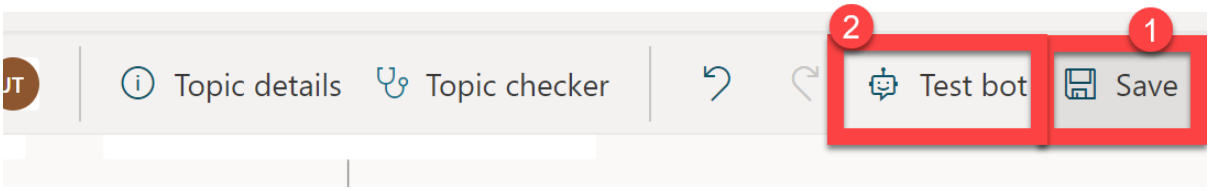
76) Click the **+** button, underneath the Cloud Flow action, and select **Show a message**



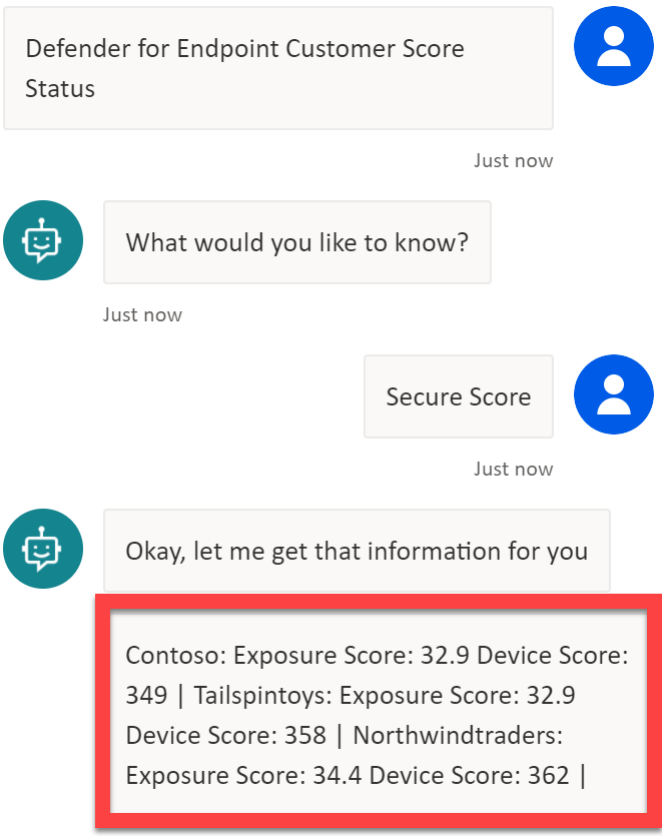
77) Select the **{x}** button and select **Response**



78) We now need to test the Power Virtual Agents bot, to ensure it is able to receive the information from the Cloud Flow, and then send that onto the end-user. Click **Save**, and then **Test bot**

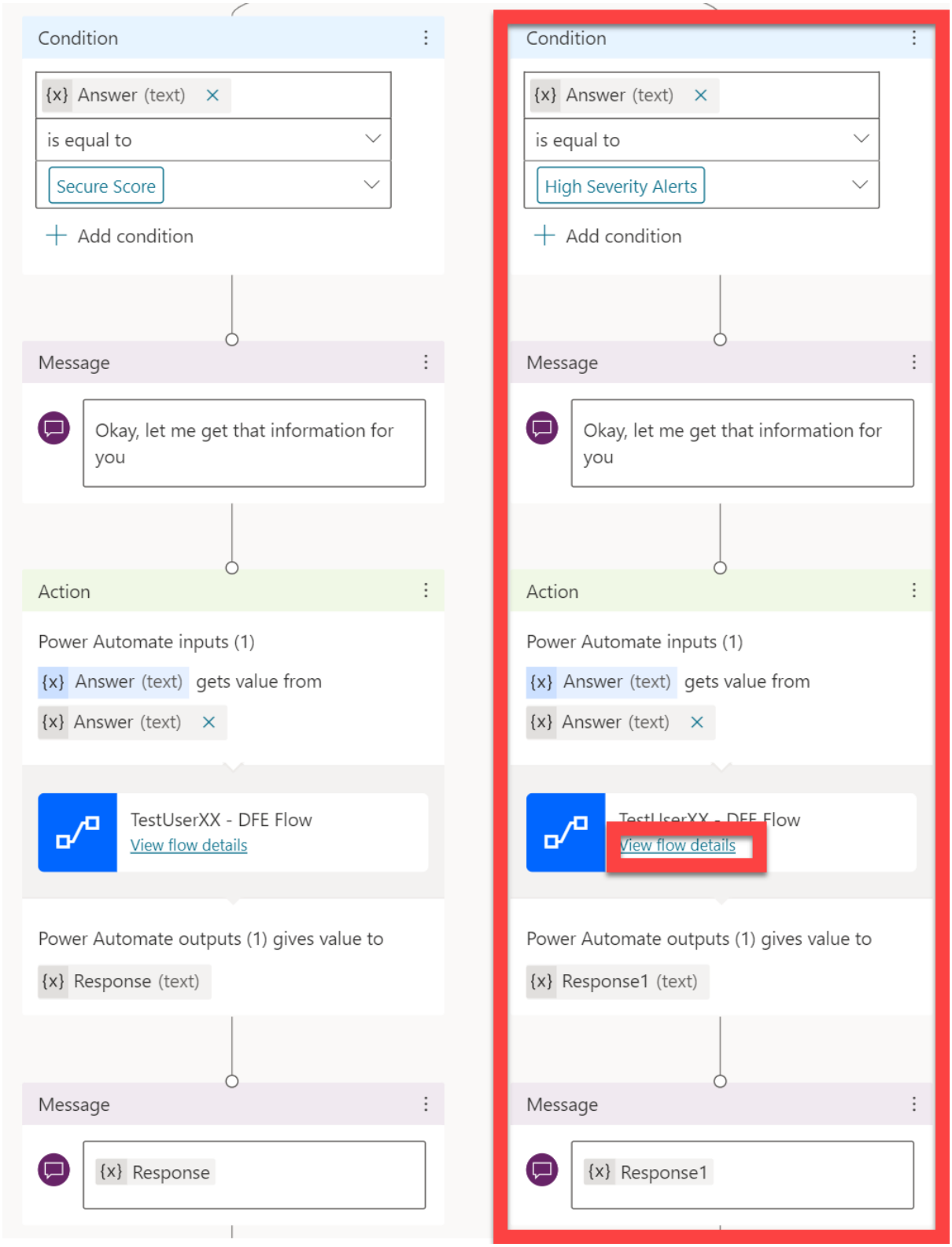


79) Ask the bot to send you information about your customer's Defender for Endpoint Score status. You should see the following:



Congratulations! You have successfully completed the first of the API flows. We now need to go back into the Cloud Flow editor, to add Actions and Logic so that it can gather a report on the customer's Defender for Endpoint High Severity Alerts

80) Create the Topic flow, you have done this previously, so for brevity, here is a screenshot of what your Topic flow should look like once completed:

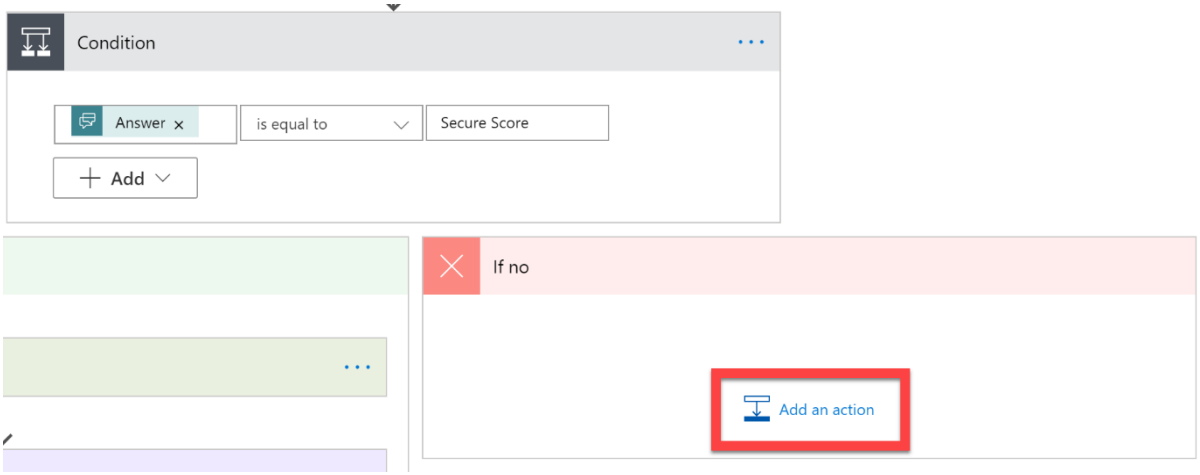


Note that we are using the same Cloud Flow for both Secure Score and Alert Topic Flows – this is a conscious choice that was made, to demonstrate that you can add logic into these types of flows, that mean you can optimise and reduce complexity by planning ahead, and passing variables through from Power Virtual Agents to Power Automate Cloud Flows.

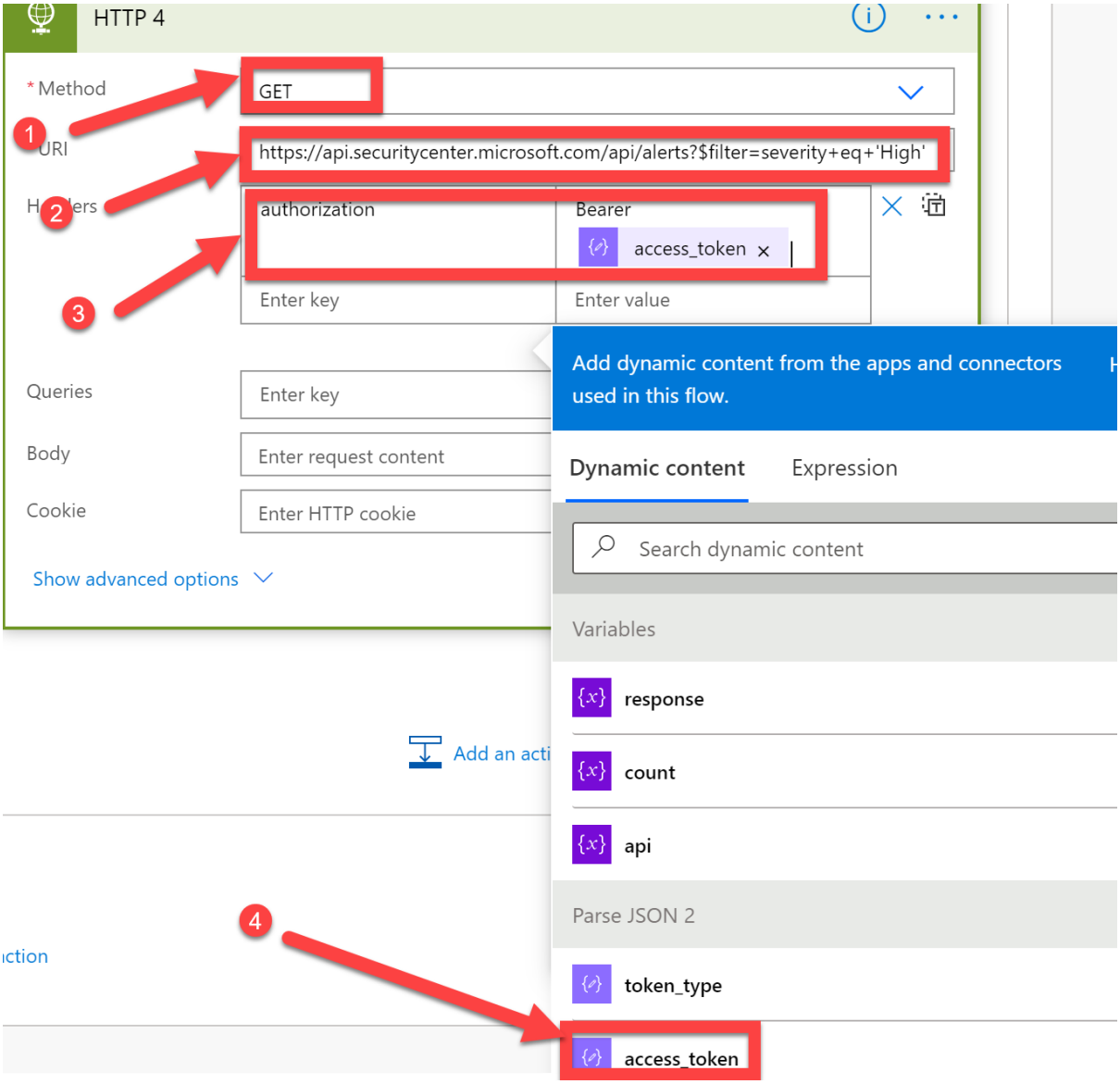
Once your Topic view looks like the above, **save** it and then select **View flow details.**

We now need to expand out the actions when the Answer variable does not equal Secure Score (i.e. they are asking about Alerts!).

81) Expand out your Condition, and then select **Add an action** under the **If no** section



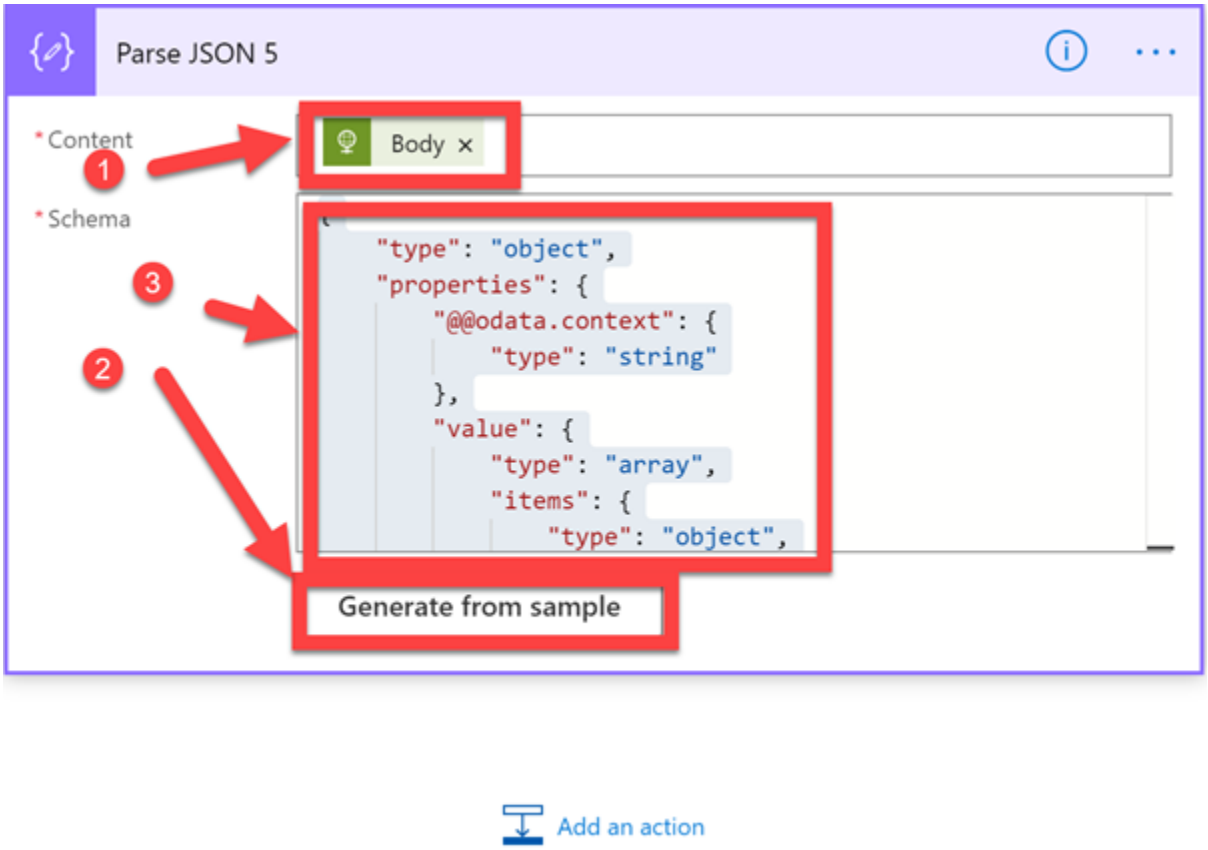
82) Add a **HTTP** action
    a. The Method of **GET**
    b. URI of **https://api.securitycenter.microsoft.com/api/alerts?$filter=severity+eq+'High'** *(note: the OData filter we are applying will only bring back alerts that have a severity of High)* .
    c. Add the **authorization** Header, referencing the **Access Token** we received earlier in the Flow.



83) Next, add a **Parse JSON** action to process the response from the HTTP request. In the Content dialog box enter the **Body** from the previous HTTP request. Select Generate from Sample and enter the following:

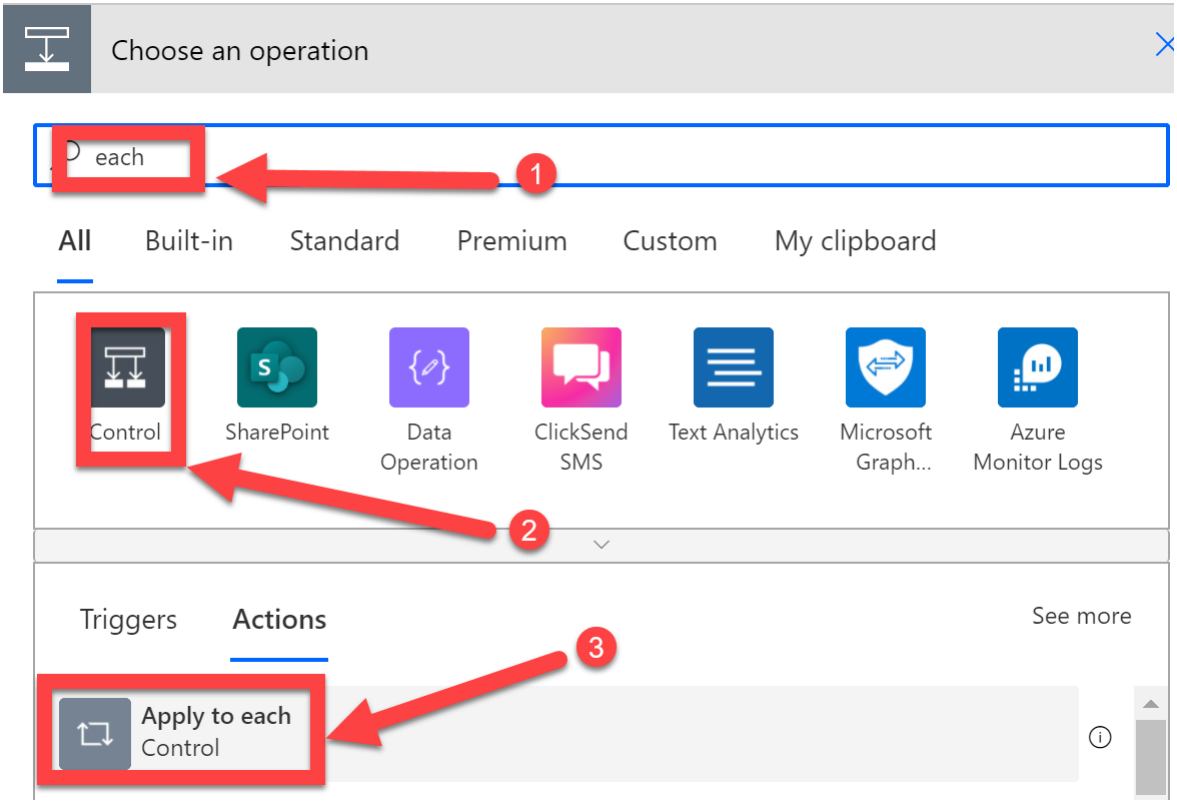
```
{
  "@odata.context":
"https://api.securitycenter.microsoft.com/api/$metadata#Alerts",
  "value": [
    {
      "id": "ed637496037226491247_1670104689",
      "incidentId": 9,
      "investigationId": null,
      "assignedTo": null,
      "severity": "High",
      "status": "New",
      "classification": null,
      "determination": null,
      "investigationState": "UnsupportedAlertType",
      "detectionSource": "CustomDetection",
      "detectorId": "72021870-e602-4766-a817-1e9153b72191",
      "category": "Execution",
      "threatFamilyName": null,
      "title": "CustomAlert",
      "description": "custom",
      "alertCreationTime": "2021-02-22T15:15:22.6241288Z",
      "firstEventTime": "2021-02-20T13:08:28.6566837Z",
      "lastEventTime": "2021-02-20T19:19:33.7203334Z",
      "lastUpdateTime": "2021-02-22T15:15:32.13Z",
      "resolvedTime": null,
      "machineId": "ef26c838f026b773fb1fb3dae1c518aa32923cda",
      "computerDnsName": "arifb-win10edun-1909",
      "rbacGroupName": null,
      "aadTenantId": "602defa2-b842-4355-bd12-49e3a683d842",
      "threatName": null,
      "mitreTechniques": [],
      "relatedUser": null,
      "comments": [],
      "evidence": []
    }
  ]
}
```


Click **Done**, once complete it should look like the following:

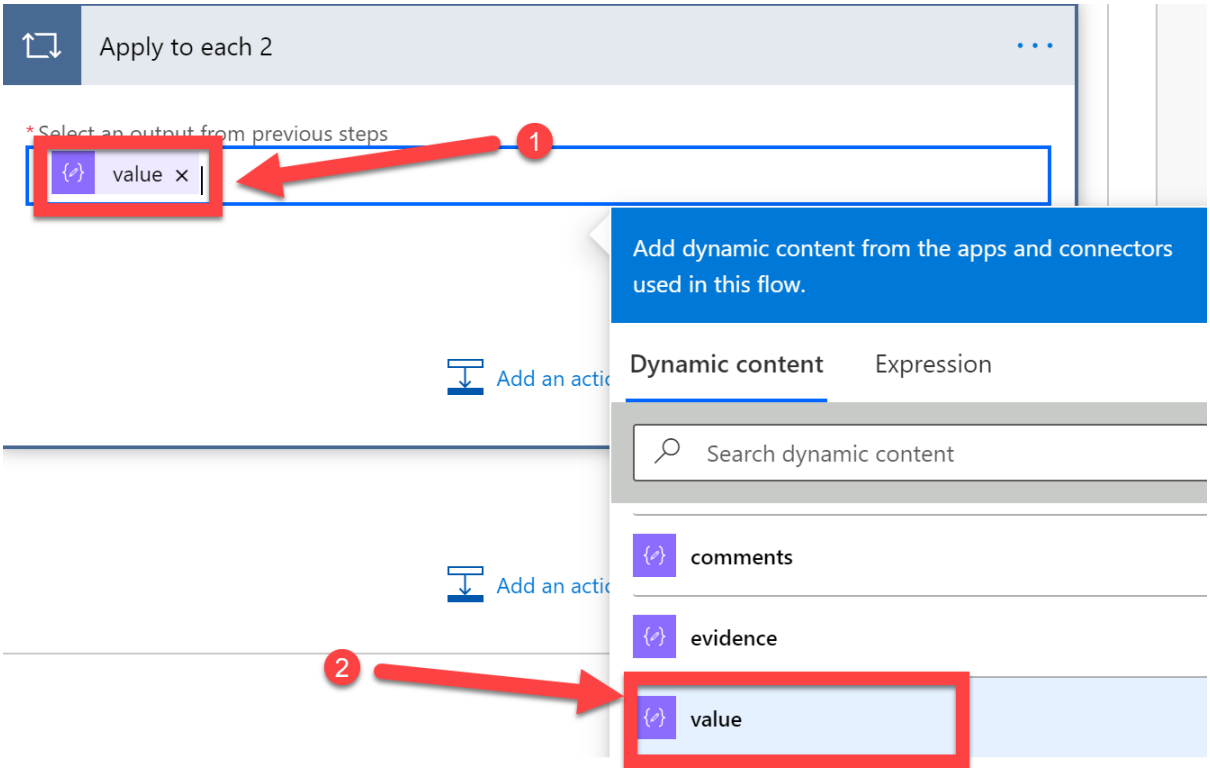


Add an action

84) Next, add an **Apply to each** Action, as we will need to use this to process multiple High Severity Alerts (assuming your customers have more than 1!)

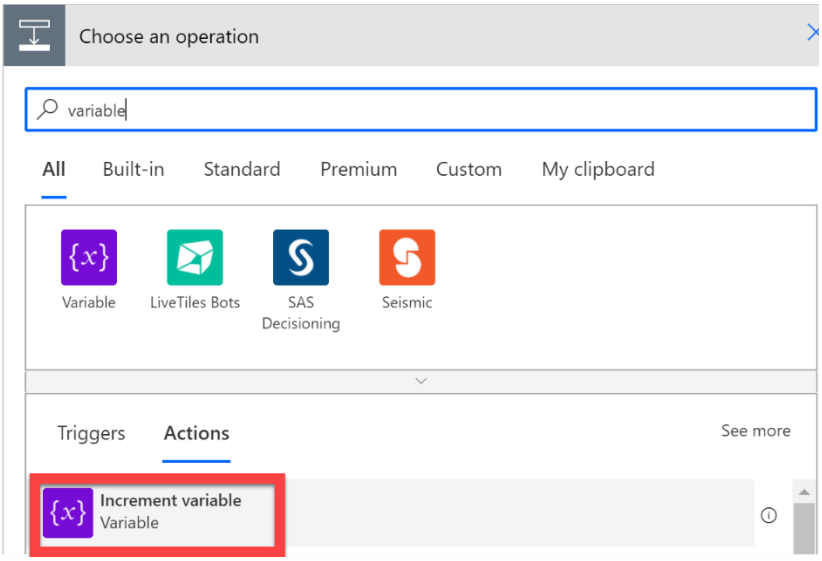


85) In the Select an output from previous steps dialog box, input the **value** from the previous Parse JSON step (in our lab, Parse JSON 5).



This will then instruct the Cloud Flow to repeat any steps contained withing this Apply to Each action, for each item/object returned from the HTTP request. For example, if a customer has 3 High Severity alerts returned from the API, it will repeat these steps, in sequence, for each alert.

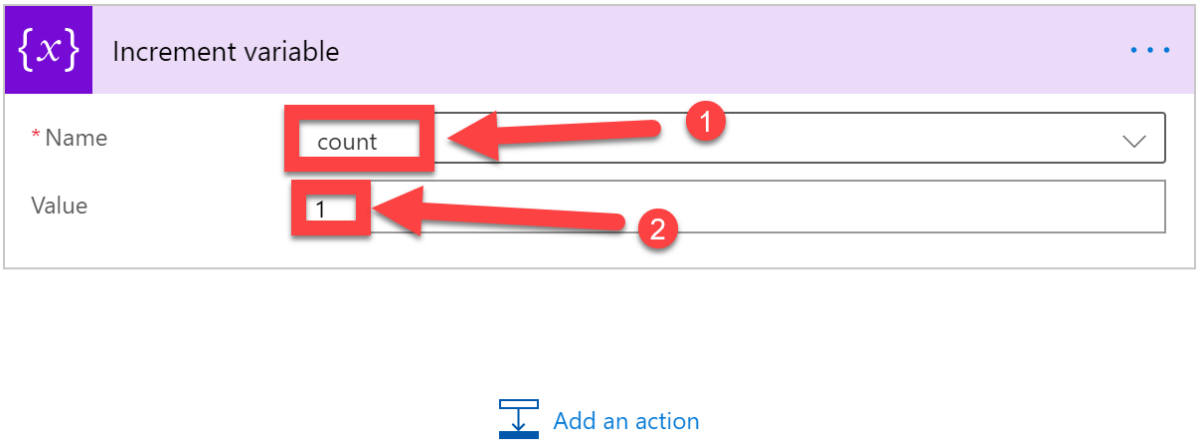
86) Next, we need to add the **Increment Variable** action inside the Apply to Each action:



87) Select the **count** variable, set the Value to **1**.

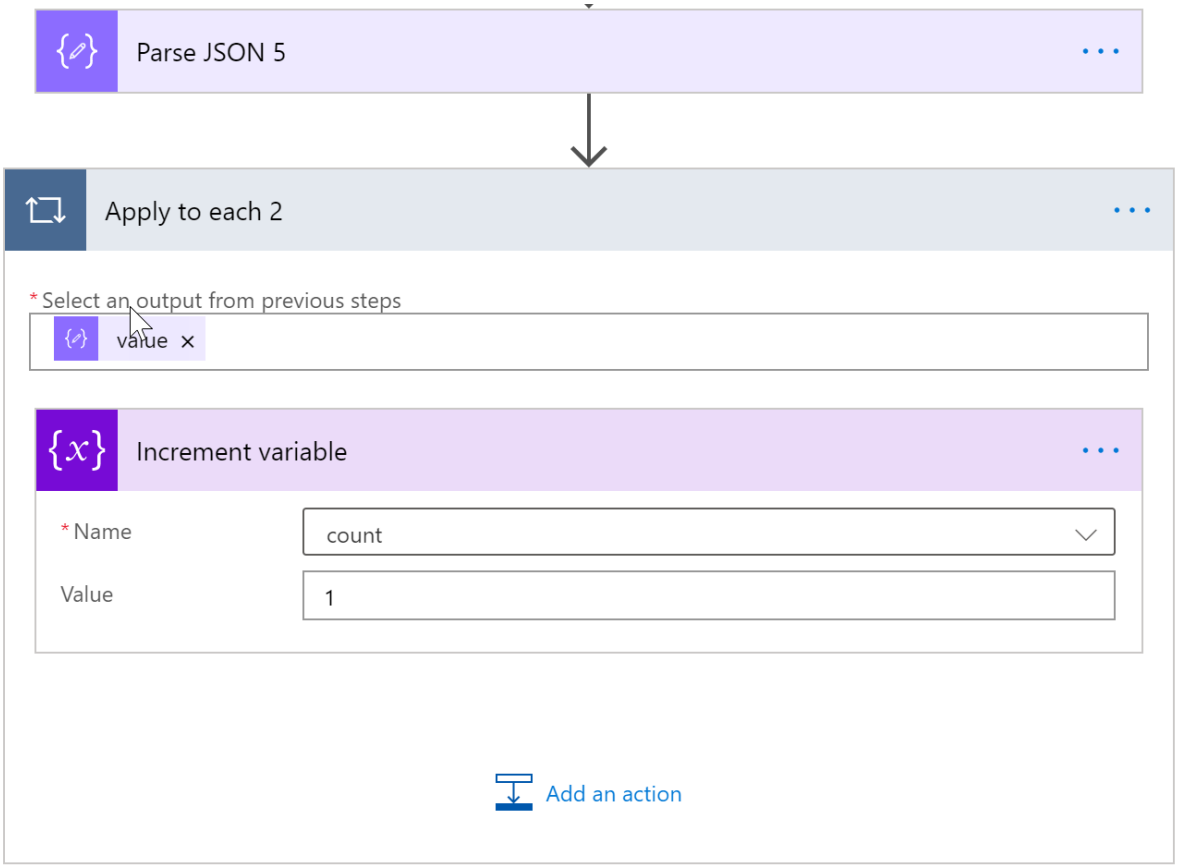
This will instruct the variable to increase by 1, every time this Apply to each action is executed. We will use this to count the number of High Severity alerts for each customer

88) Click **Add an action**

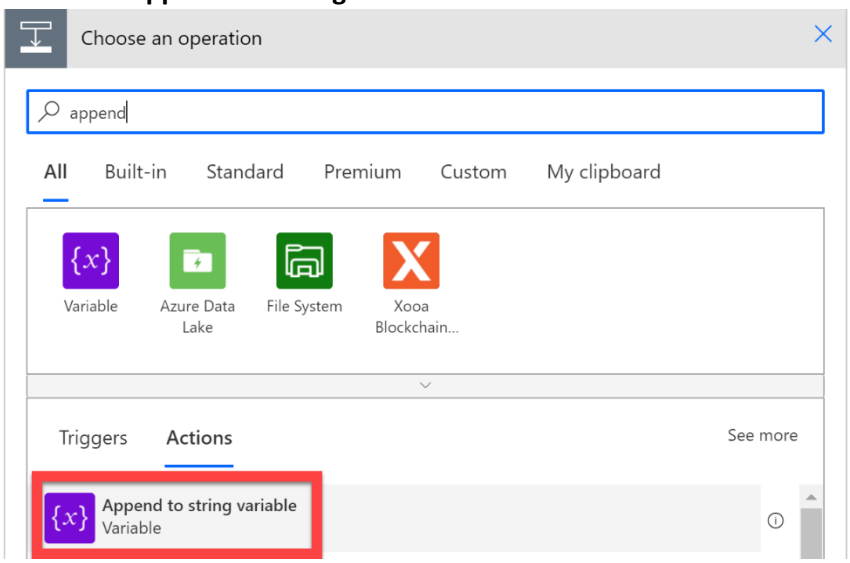


Now we need to add some actions outside of the Apply to Each action. This will allow us to report back the number of High Severity Alerts, and also reset the count variable back to 0 for the next customer.

89) Click the **Add an action** button detailed below (outside of the Apply to Each).

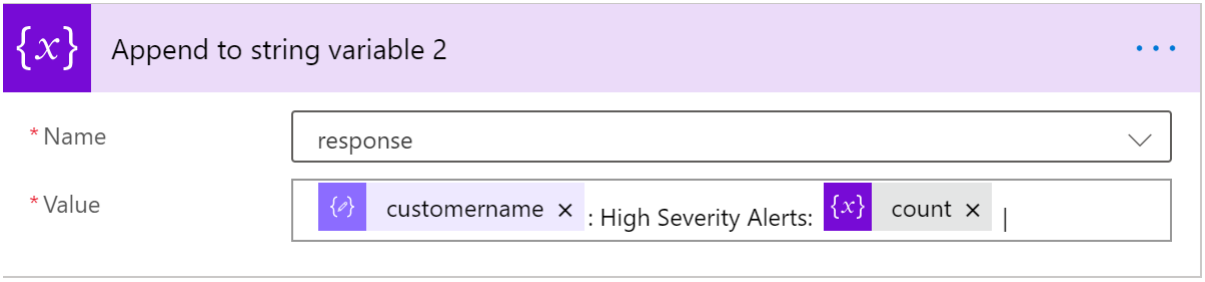


90) Select the **Append the string variable** action



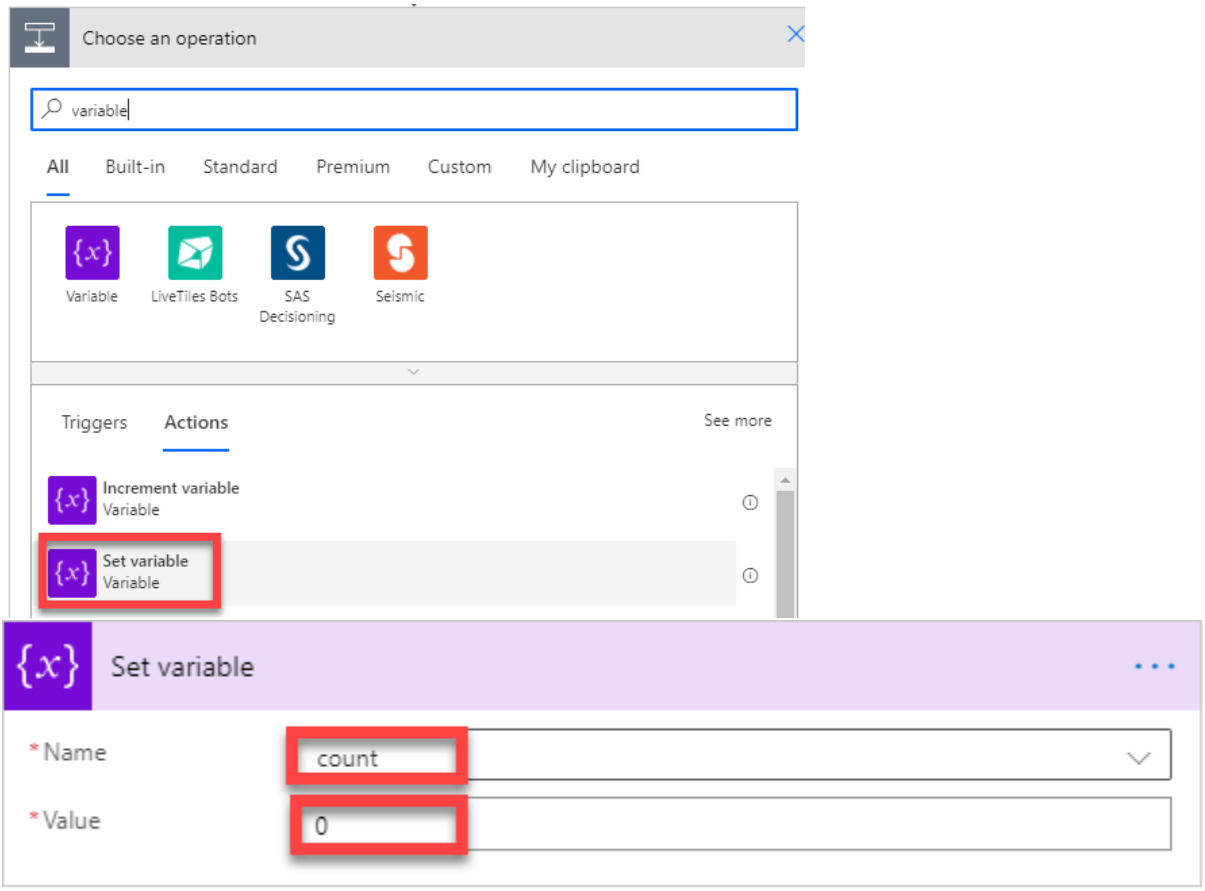
Similar to earlier in the lab, we need to append the customer name, and the number of High Severity alerts to the response string variable, using the dynamic content selector.

91) . Complete as follows:

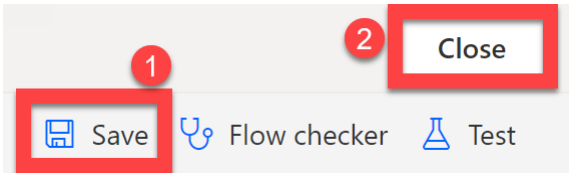


92) Finally, we need to make sure that we reset the count variable back to 0 – to achieve this,

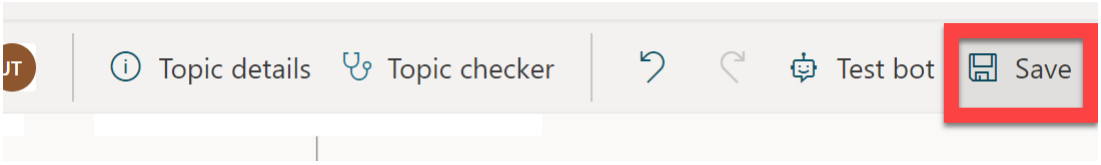
93) Add the **Set variable** action next, and set the **count** variable back to **0**



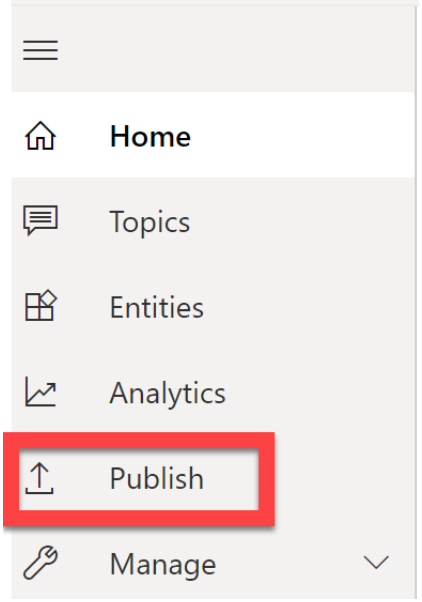
94) We now need to get the Cloud Flow saved, the Bot published and deployed into Teams. Click **Save** and **Close**



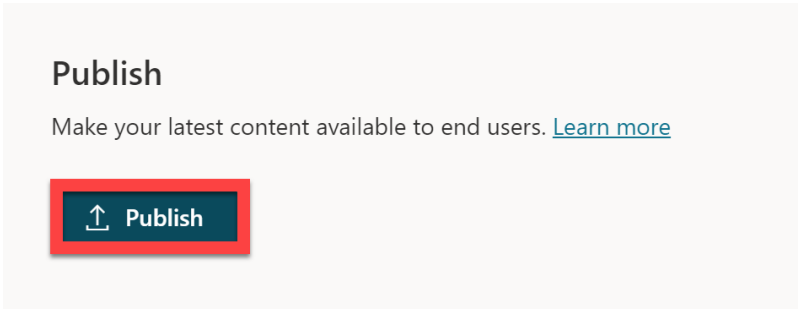
95) Now that you are back in the Virtual Agents topic editor, click **Save**

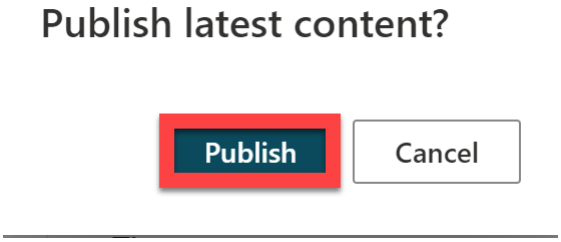


96) Expand the navigational menu and select **Publish**

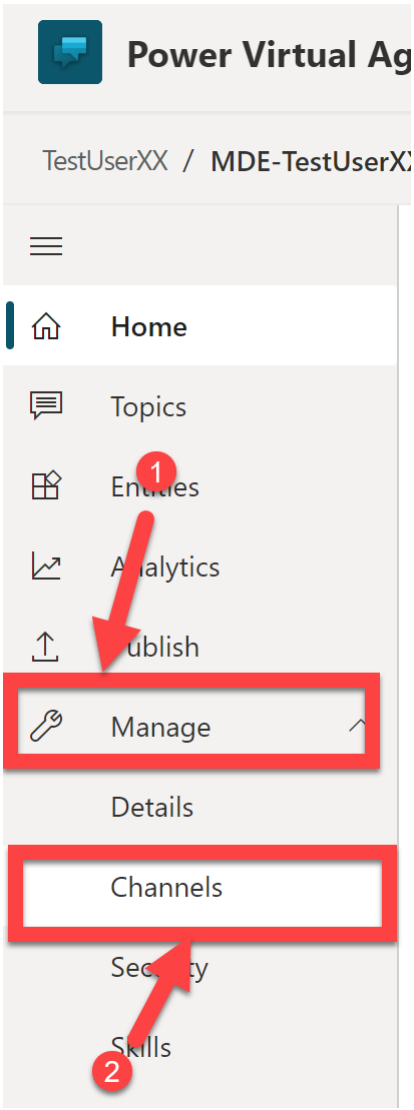


97) **Publish** your latest Bot Content

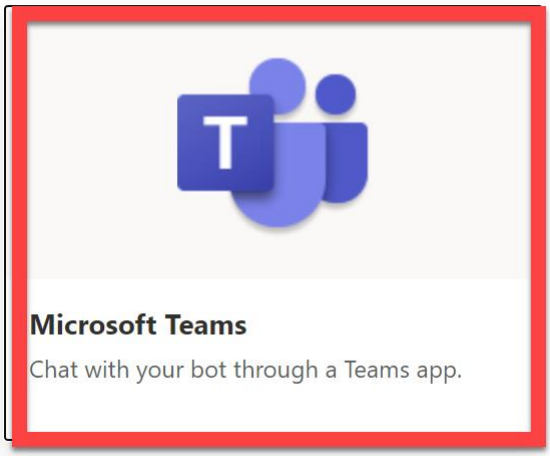




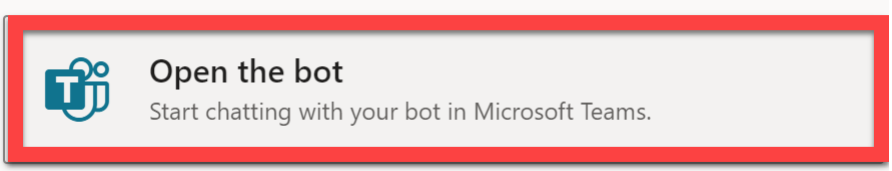
98) Select **Manage**, click **Channels**



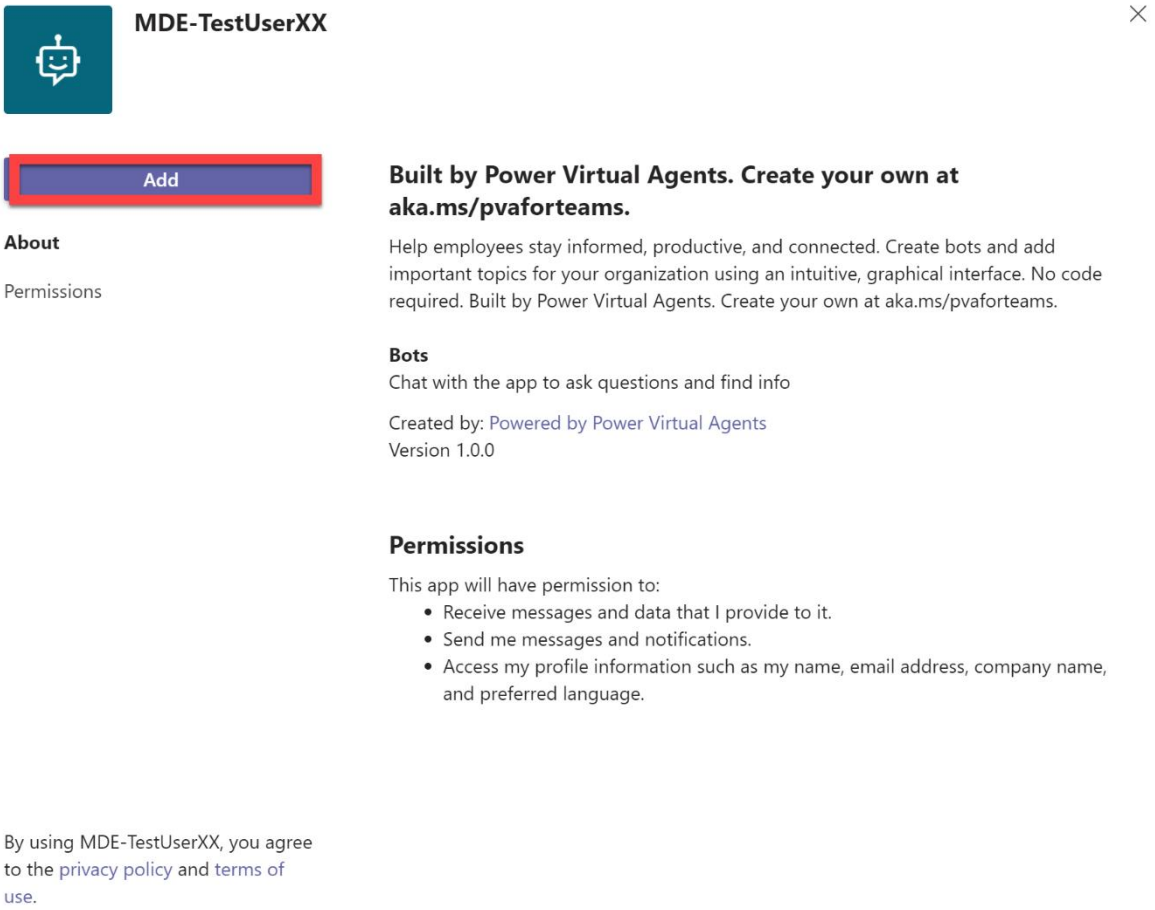
99) Select **Microsoft Teams**



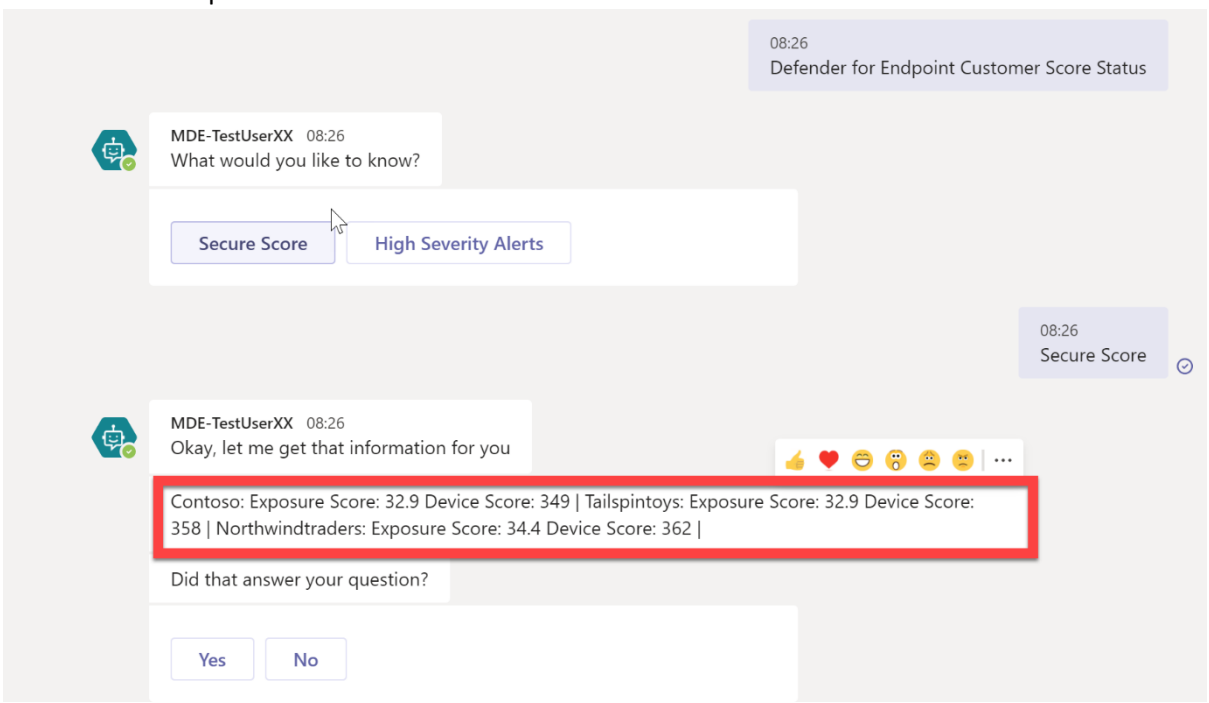
100) Click **Open the bot**

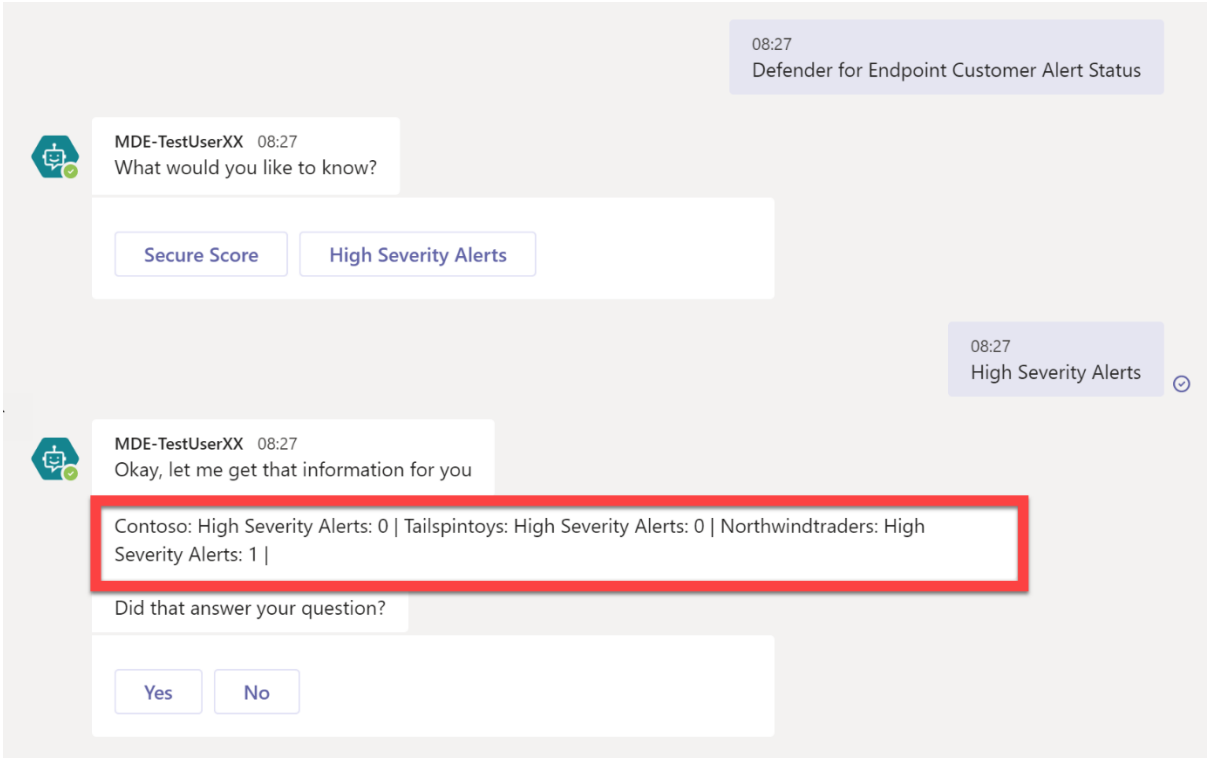


101) Click **Add**



102) Have a conversation with the bot, you should be able to interrogate the customer Defender for Endpoint status for Alerts and Score:





103) If your bot is not behaving as expected, send it a **start over** message to reset the state and for it to access the latest published content

Congratulations! You have now successfully completed this lab. We hope that you found this lab, and the associated lab materials useful. We look forward to seeing what you build as a result of attending this lab!

Lab Complete.