Lab Guide: Introduction to AI Governance in watsonx

Author: Elena Lowery, elowery@us.ibm.com

Contents

Overview

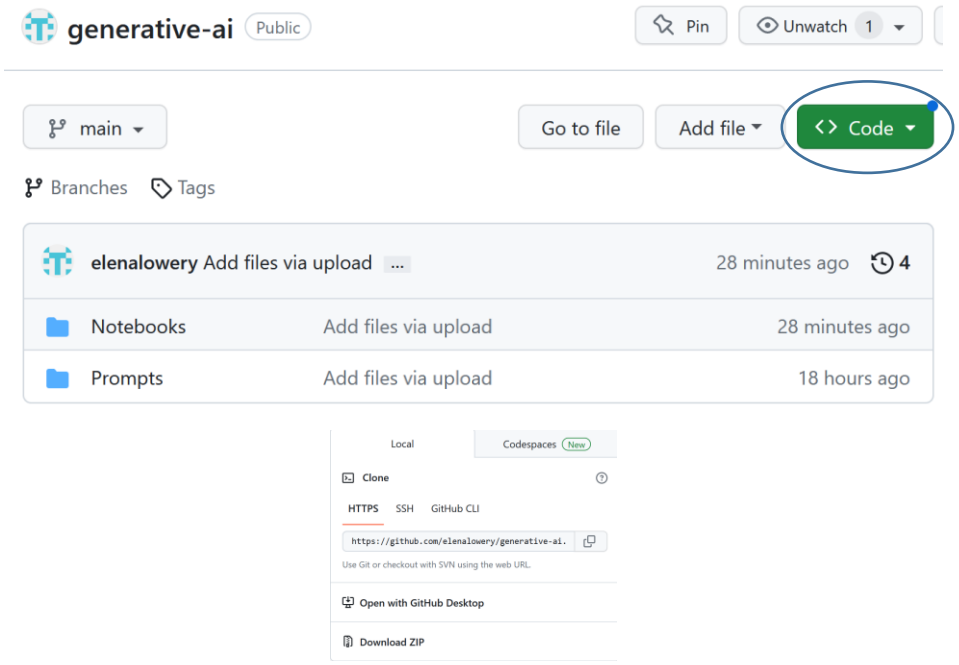
In this lab you will learn how to track and evaluate large language model (LLM) prompts in *watsonx.governance*. We will complete the following tasks:

- Create a model inventory
- Create a use case
- Test and evaluate a prompt
- Promote the prompt to production
- Use the prompt from a client application.

Required software, access, and files

Note: if you have completed these steps in the previous labs, you don't need to repeat them.

- To complete this lab, you will need access to *watsonx.ai* and *watsonx.governance*. You can get access by signing up for an IBM Cloud account and provisioning watsonx.ai and watsonx.governance services.
- Recommended, but not required: A Python IDE with Python 3.10 or 3.11 environment (*Visual Studio Code* or *PyCharm*). If you don't have the Python IDE, you can test integration with LLMs in watsonx.ai Notebooks.
- You will also need to download and unzip this GitHub repository: *https://github.com/elenalowery/generative-ai*
- Click the **Code** button and select **Download ZIP**



Unzip the downloaded zip file. In the lab, we will refer to this folder as the *git repo* folder.

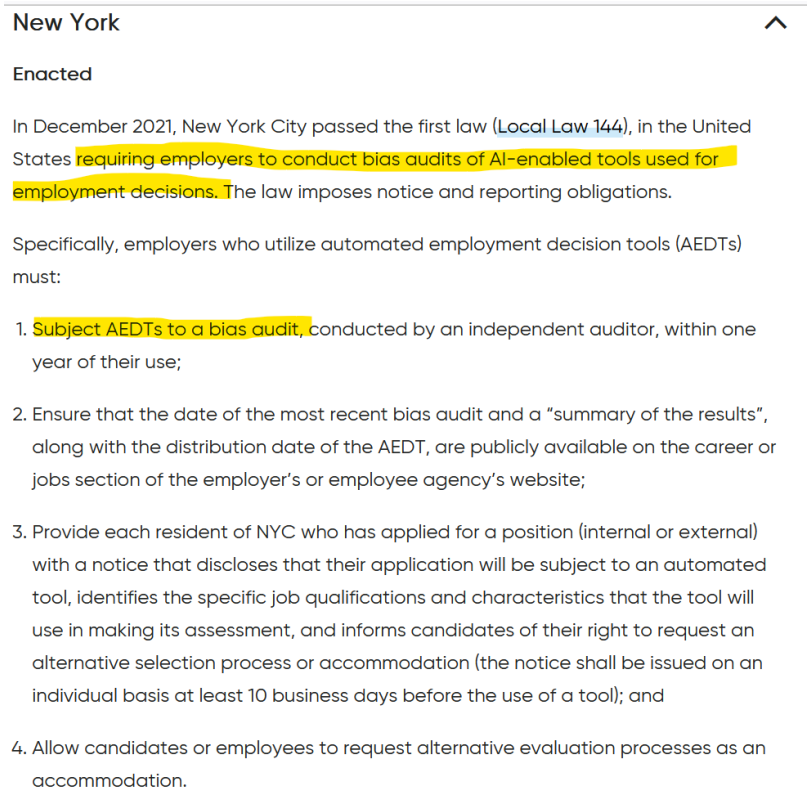
Important note: *Some screenshots in the lab may be slightly different from the product. If you have questions, please ask your workshop instructor.*

AI Governance and watsonx.governance

AI Governance is a discipline that includes business process and implementation of best practices for *creating*, *deploying*, *managing*, and *monitoring* AI artifacts, such as Machine Learning (ML) models and Large Language Models (LLMs).

In the past, components of *AI Governance* were implemented by companies in highly regulated industries or for specific use cases. For example, companies in the financial industry must have Model Risk Governance (MRG) practice to comply with regulatory requirements that were first published in 2012, and even as early as 2000. However, in some cases, the implementation of AI Governance was simply keeping track of models in a spreadsheet or a document.

During the past few years, various government organizations proposed and approved laws for governing AI. Here is an example of a law related to the use of AI:



New York

Enacted

In December 2021, New York City passed the first law (Local Law 144), in the United States requiring employers to conduct bias audits of AI-enabled tools used for employment decisions. The law imposes notice and reporting obligations.

Specifically, employers who utilize automated employment decision tools (AEDTs) must:

1. Subject AEDTs to a bias audit, conducted by an independent auditor, within one year of their use;
2. Ensure that the date of the most recent bias audit and a "summary of the results", along with the distribution date of the AEDT, are publicly available on the career or jobs section of the employer's or employee agency's website;
3. Provide each resident of NYC who has applied for a position (internal or external) with a notice that discloses that their application will be subject to an automated tool, identifies the specific job qualifications and characteristics that the tool will use in making its assessment, and informs candidates of their right to request an alternative selection process or accommodation (the notice shall be issued on an individual basis at least 10 business days before the use of a tool); and
4. Allow candidates or employees to request alternative evaluation processes as an accommodation.

Source

Regulatory landscape for AI continues to evolve, with important regulations such as the United States AI Bill of Rights and EU AI Act will likely accelerate development of guidelines and laws.

One of the best ways to prepare for upcoming regulations is to infuse AI Governance in several steps of the ML model and LLM lifecycle. *watsonx* includes multiple features that

will streamline not only the governance of models but also improve operationalization and simplify maintenance.

In this lab, we will focus on the *operational* aspects of implementing AI Governance. We will cover *Model Risk Governance (MRG*) and the automatic integration of MRG with the model lifecycle in a separate lab.

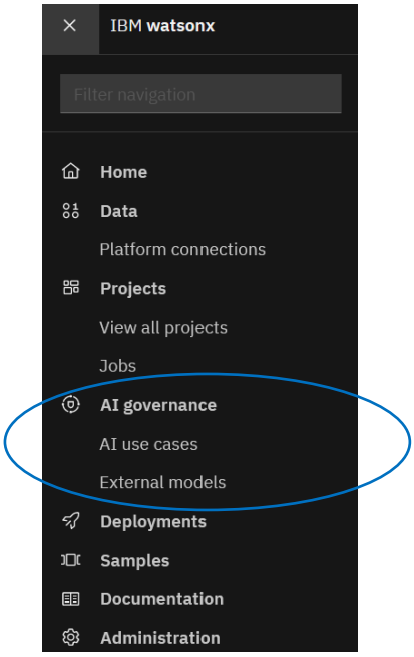
Watsonx.governance

Many software offerings today are packaged as individual services that include a set of features. Services provide granular capabilities that can be included in multiple offerings. IBM's *watsonx.governance* includes two services:

- **AI Factsheets**: used to capture model and prompt metadata as well as track model lifecycle
- **OpenScale**: used to monitor model performance, accuracy, provide explainability, and alert for bias.

It is common in microservice software design to have prerequisite services. In the case of *watsonx.governance*, the prerequisite services are *watsonx.ai* and *Watson Machine Learning*.

From the end-user perspective, all services are seamlessly integrated into *watsonx* UI. As additional services are provisioned, new actions (menus/buttons) become available in various components of *watsonx*. In other words, end users don't need to know the names of the services, but this information may be useful to administrators. For example, if a user doesn't see a certain action, the administrator can check if the required service was provisioned.



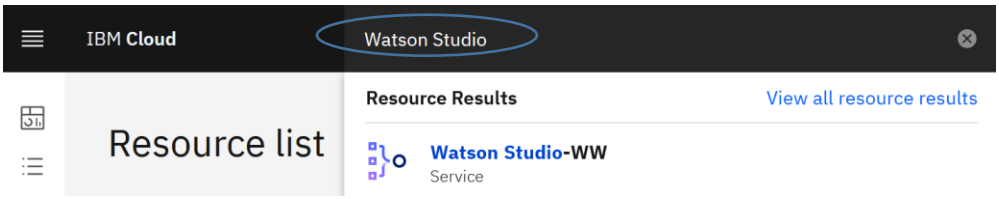
Part 1: Create a Model Inventory

Creating a *Model Inventory* is the first step in better organization of AI artifacts.

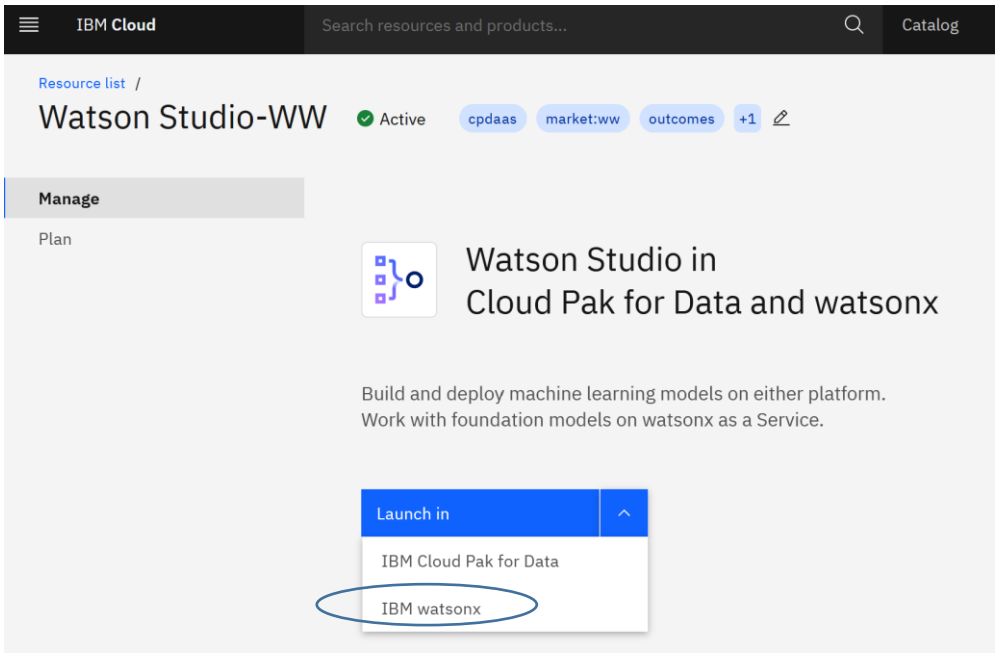
1. Log in to your **IBM Cloud account**. Your lab instructor will provide the URL and userid/password.

2. If you're not already in watsonx.ai service, you can navigate to it from the **IBM Cloud Resources** page.

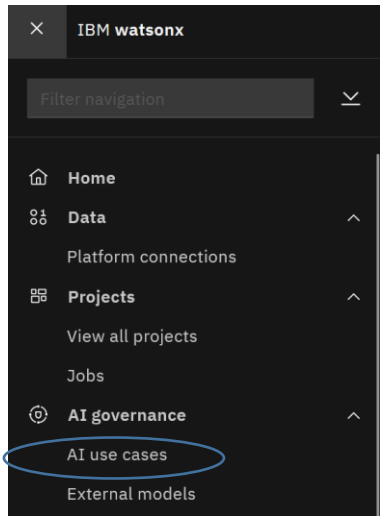
   - Search for *Watson Studio* in the search box on top of the page.



   - On the *Watson Studio* service page, make sure to select *Launch in IBM watsonx*.

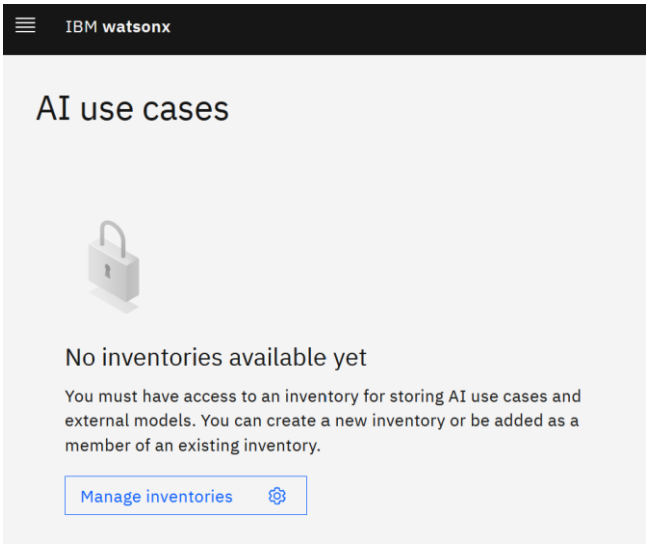


3. From the main menu in the top left corner select **AI Use Cases**



Note: If you don't see this menu, check that OpenScale service was provisioned in your IBM Cloud account.

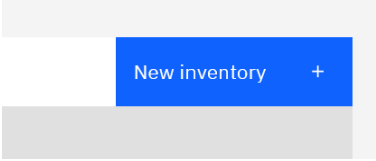
4. If you have not previously created a Model Inventory, you will see an option to create it. Click **Manage Inventories**.

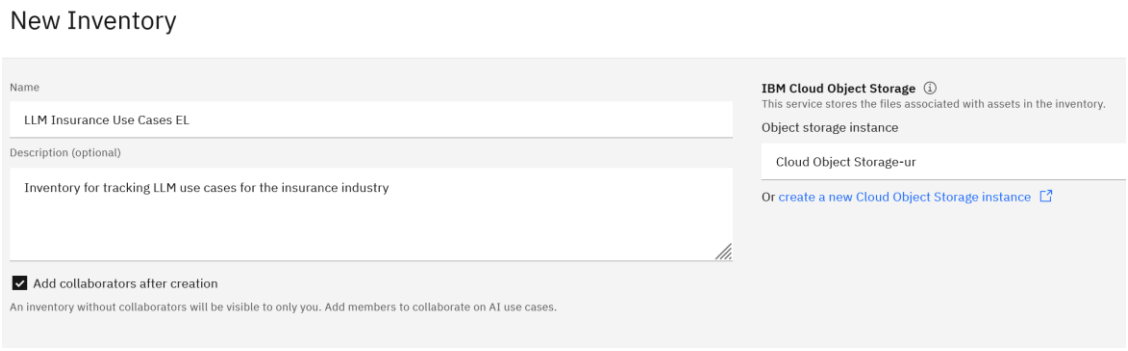


5. Click on the **New Inventory** button, then provide a unique name for your inventory, for example, *LLM Insurance Use Cases <your initials>*. Make sure to select *Cloud Object Storage* associated with your account id.

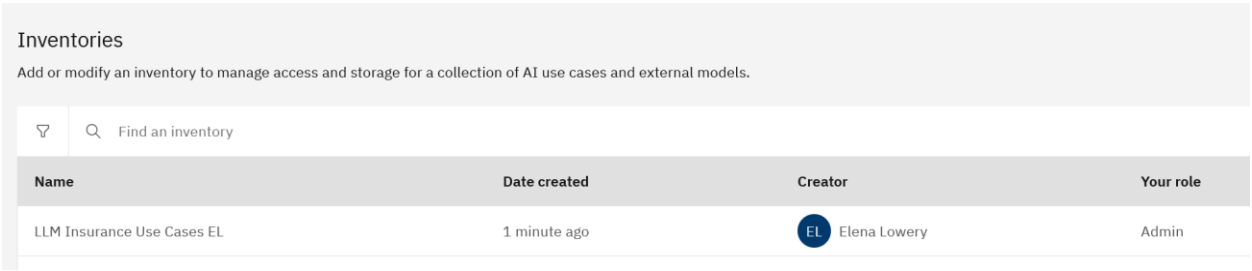
   Click **Create**.

   Optionally, add collaborators and close the screen.





Your inventory will look similar to the following screenshot:

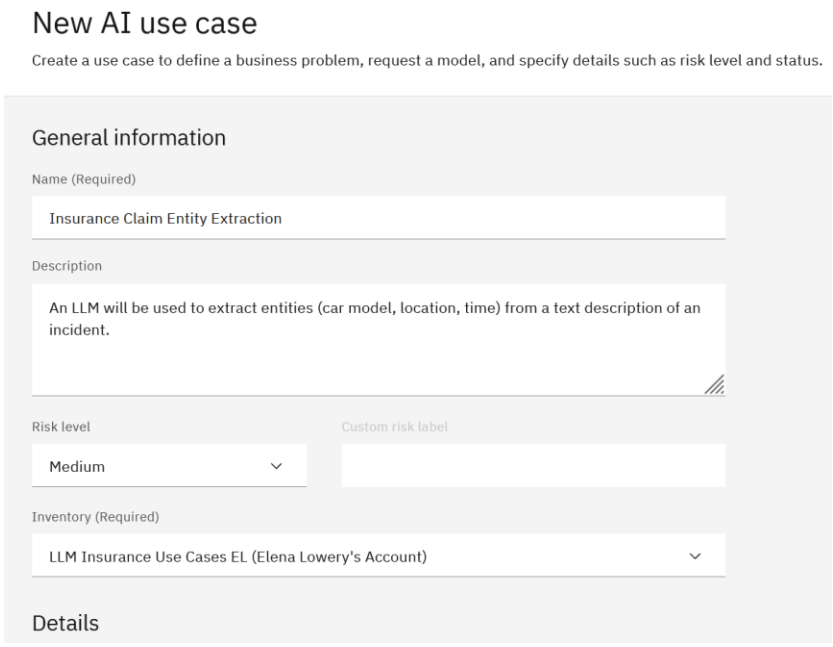


Next, we will create an AI use case.

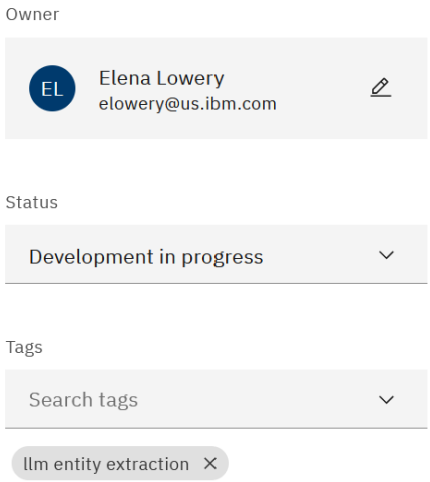
6. Navigate to the **AI Use case** menu again and click the **New AI use case** button.

7. Review metadata options available on this screen.

   First, we provide use case information, such as *name* and *description*, as well as *risk level* and *inventory name*. Risk level designation can be based on industry regulations or an organization's evaluation of risk. It can also help with prioritization of work if multiple models perform below specified thresholds.



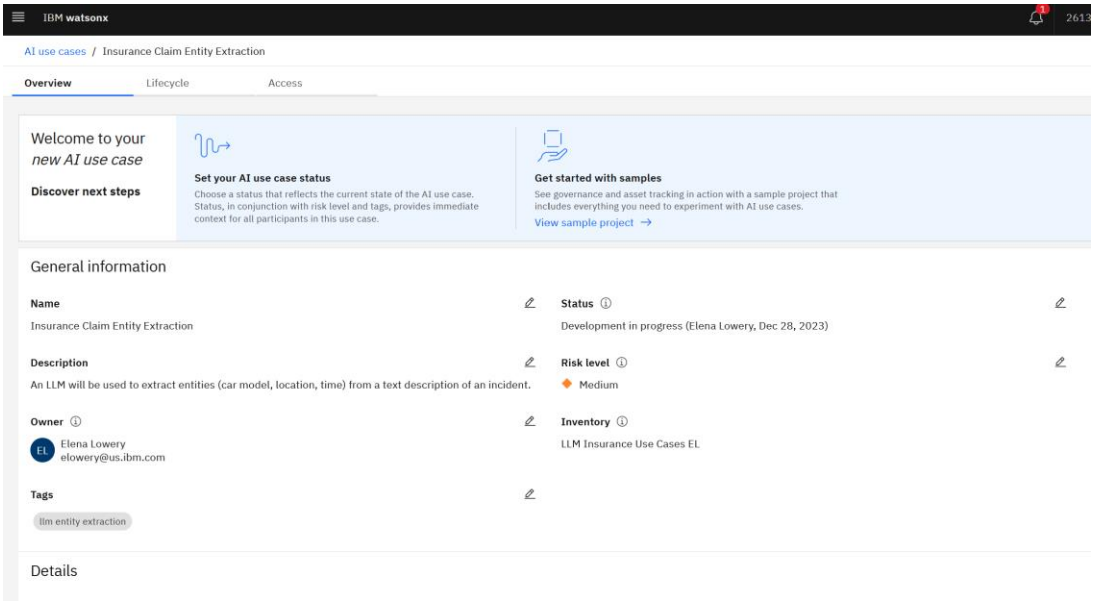
   On the right side, we can capture some aspects of model lifecycle process, specifically the status, which we decided to set to *Development in Progress*. Adding tags to the use case entry will help us easily find use cases in the inventory.



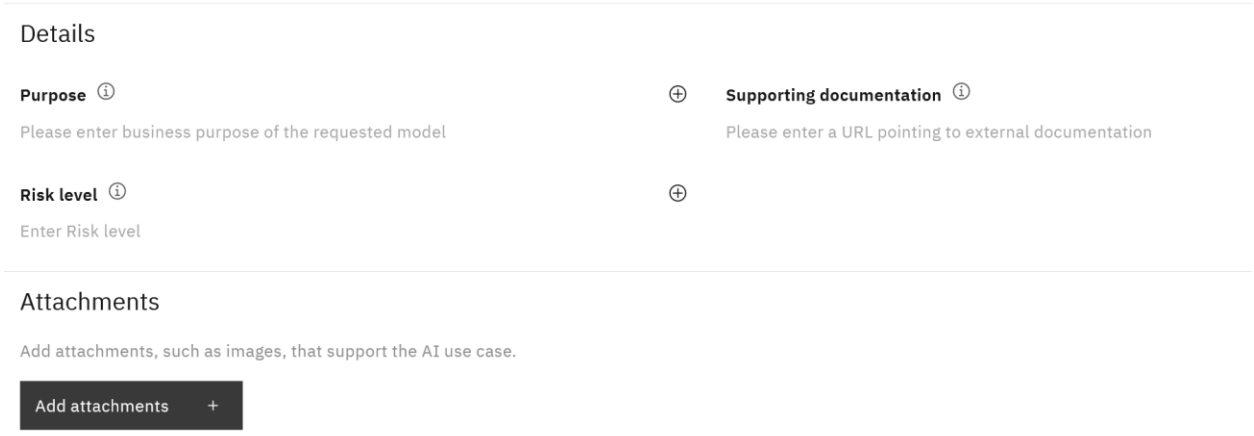
Create a use case record using the sample values in the screenshot or provide your own.

- Name: *Insurance Claim Entity Extraction <Your Initials>*
- Risk level: *Medium*
- Status: *Development in progress*

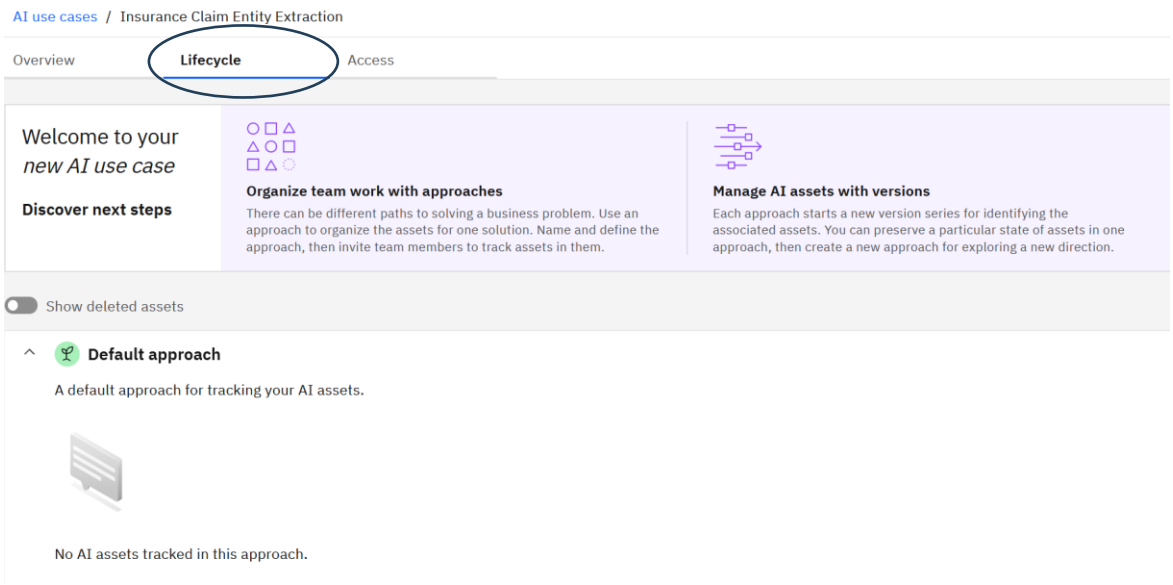
The detailed *Use Case* view provides the summary of details we entered.



Notice that we can add different types of supporting documentation to the use case, which is useful for both transparency and auditing of AI use case implementation.

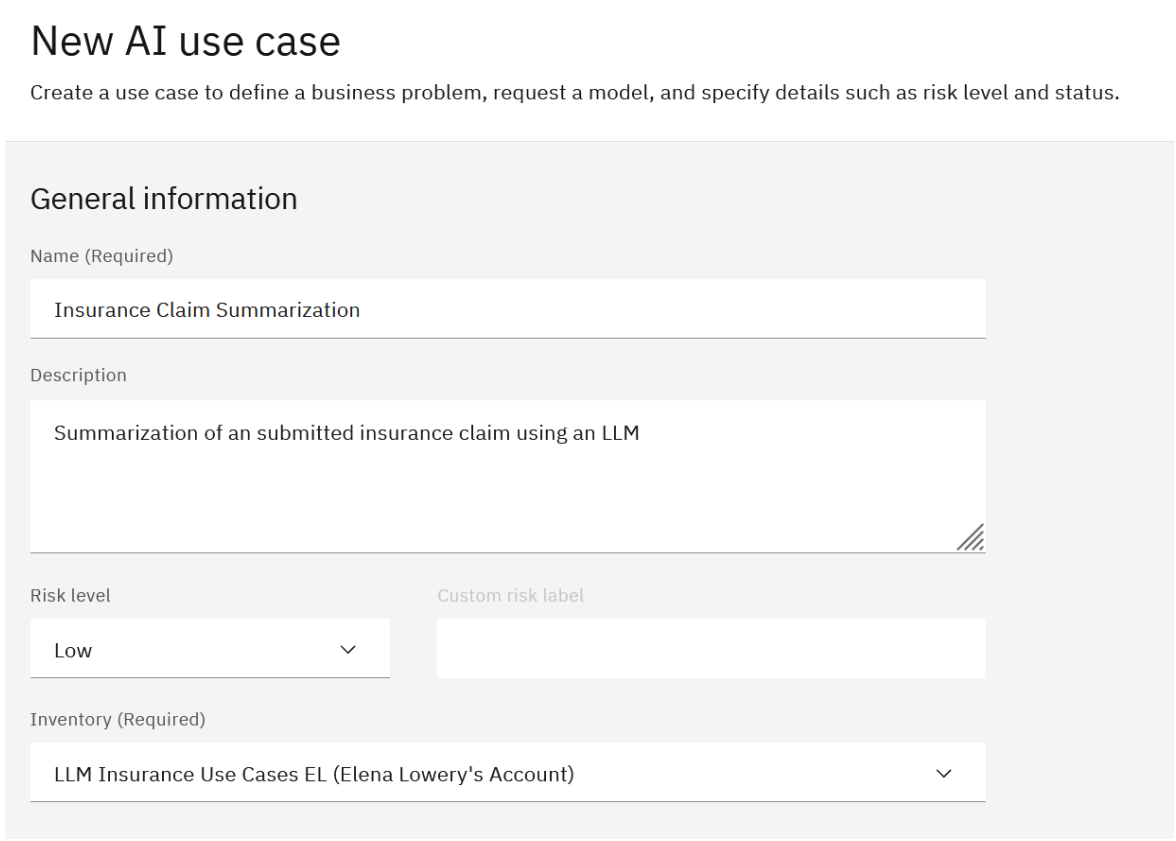


If you click on the *Lifecycle* tab, you will notice that we don't have any information yet because we have not added project assets that we want to track.

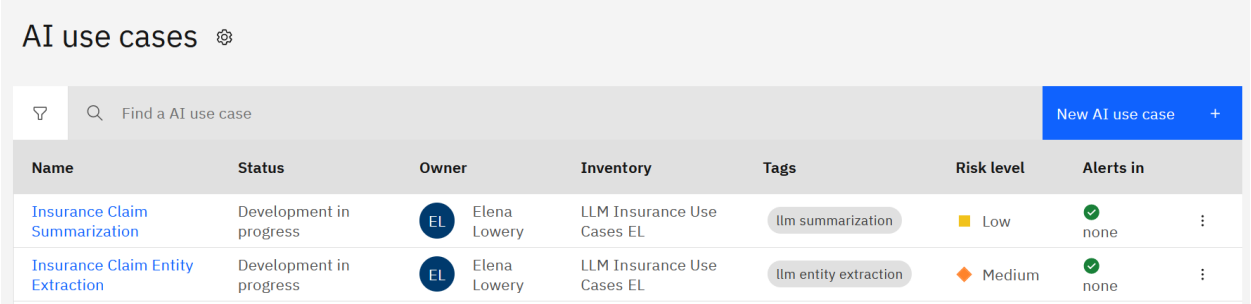


8. Create a second use case entry for the *Summarization* use case. You can use the values in the following screenshot or provide your own.

   - Name: *Insurance Claim Summarization <Your Initials>*
   - Risk level: *Low*
   - Status: *Development in progress*



When finished, you AI use case view should look similar to this screenshot.



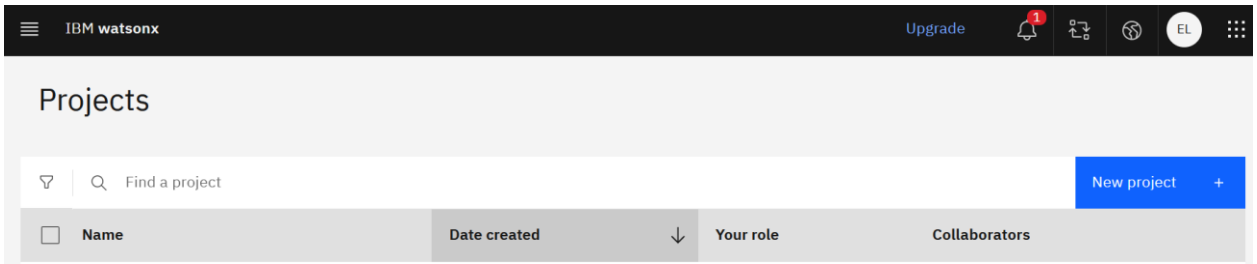
We have completed setting up a *Model Inventory* and creating metadata for AI use cases that we will implement with LLMs. In just a few minutes, we provided a better organization for auditing, tracking, and monitoring AI assets that will be used in various applications.

Part 2: Prompt Development and Evaluation

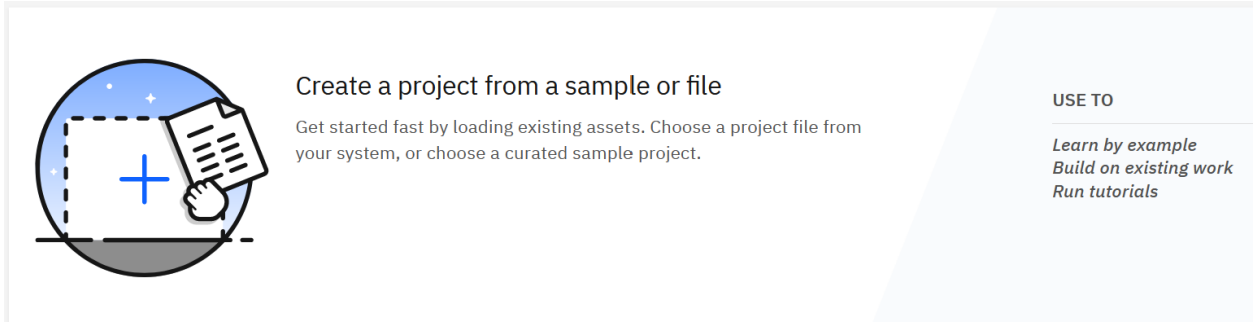
In this section, we will use provided prompts for the defined use cases to walk through the lifecycle of deploying a prompt/LLM for use by production applications. We will not cover the iterative process of prompt engineering and prompt tuning. Since we are focusing on governance, we are making the assumption that the development process for the prompts has been completed.

First, we will import the project that contains prompts into watsonx.

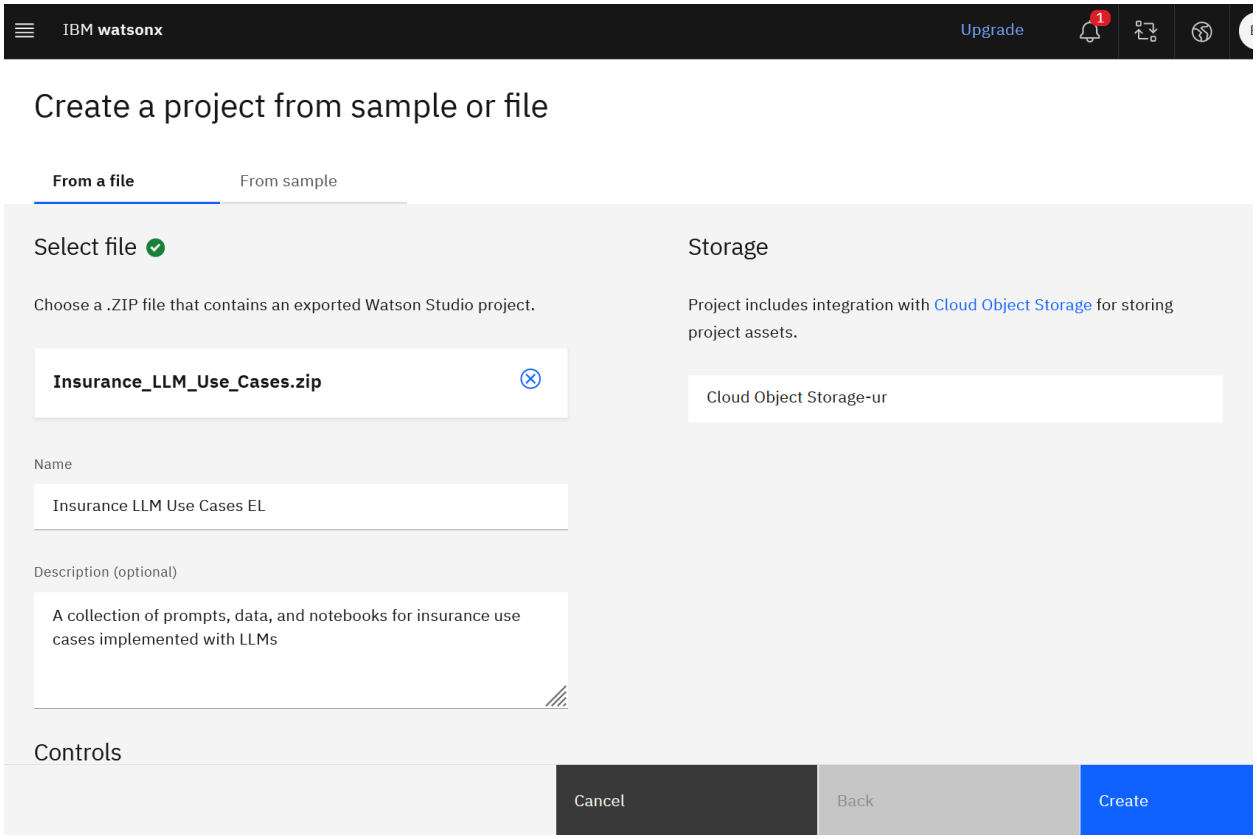
1. In *watsonx* navigate to the **Projects** view (from the main menu in the top left corner) and click the **New Project** button.



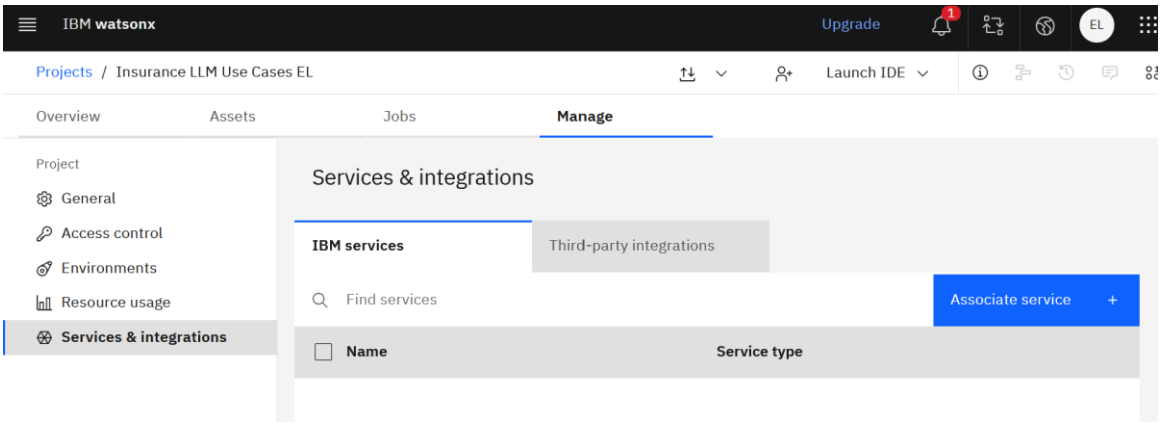
2. Click the **From file** option and navigate to the downloaded lab repo */Cloud Projects* folder to select *Insurance_LLM_Use_Cases.zip.*



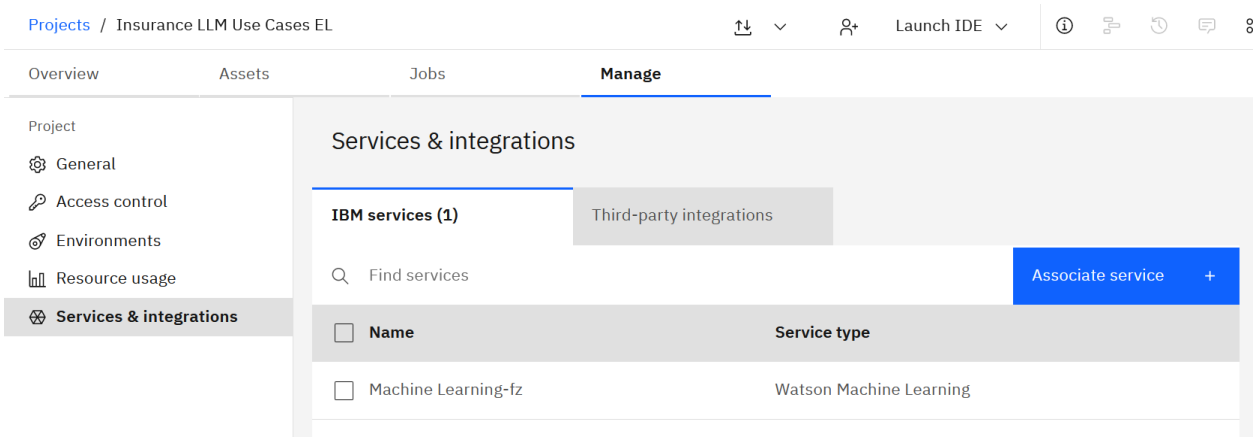
Provide a unique project name and click **Create**.



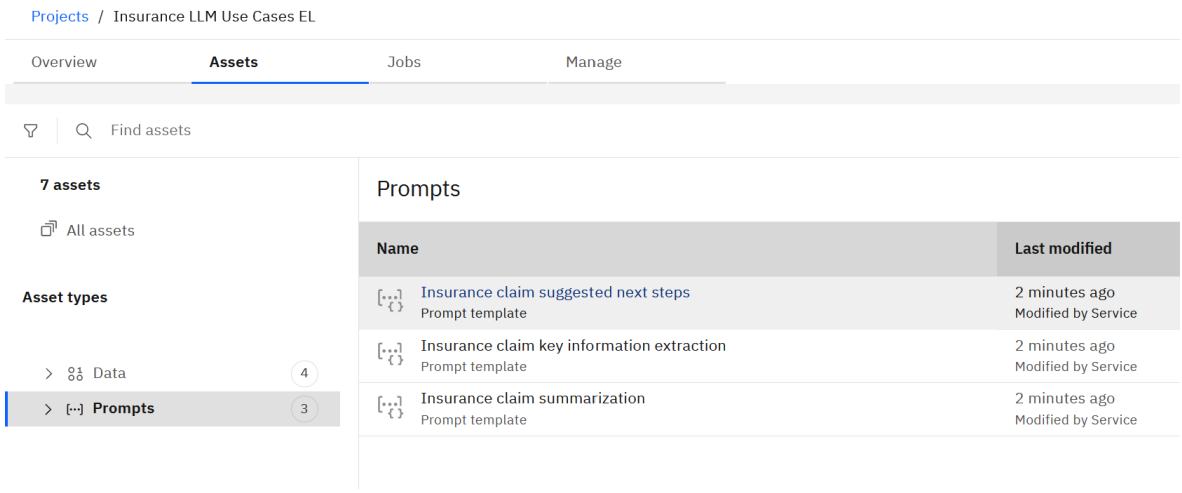
3. Switch to the **Manage** tab, then select **Services and Integrations** tab. Click **Associate Service**.



4. Select the displayed **Machine Learning** service and click **Associate**.



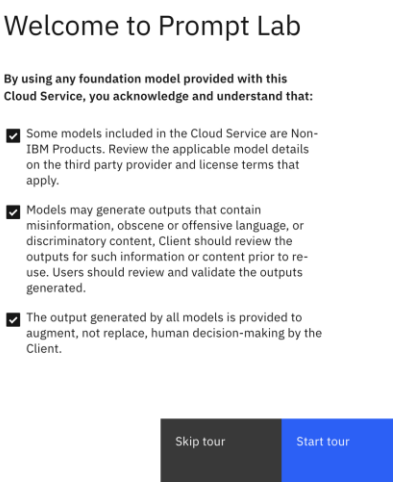
5. Switch to the **Assets** tab and expand **Prompts**.



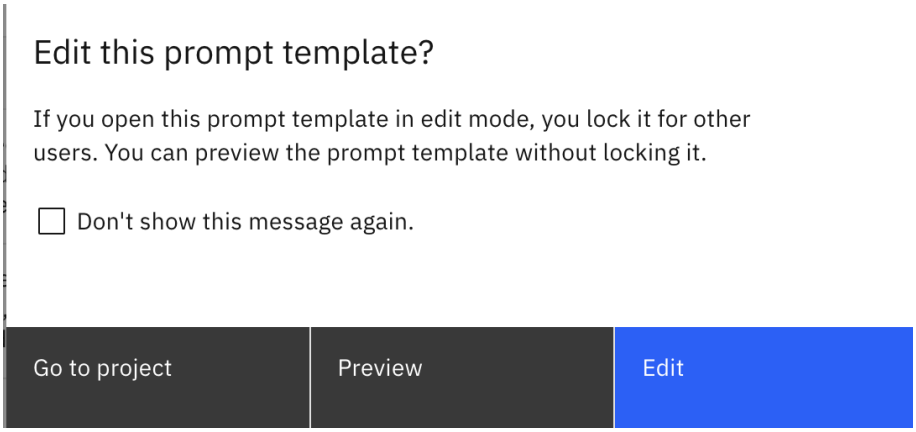
Let's review prompts for our AI use cases.

6. Click on the *Insurance claim key information extraction* prompt. We are making an assumption that you are familiar with prompt structure, but let's point out a few things that may be unique to prompt format in the *Prompt Lab*.

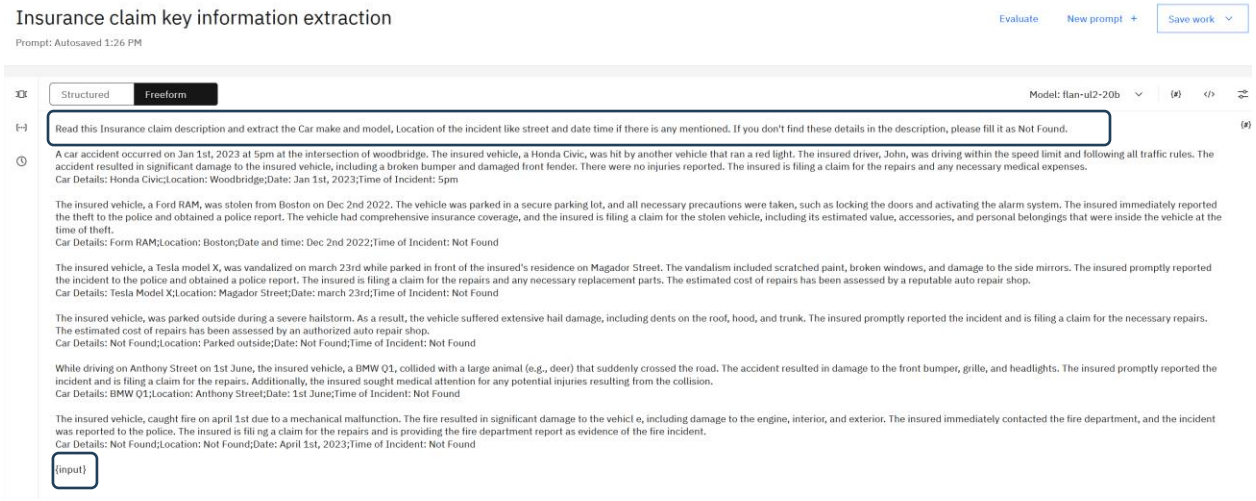
   Note: when prompted, check all the boxes to agree to terms and click *skip tour*.



When you open the prompt, you can choose to open in preview mode or edit mode. Edit mode will lock the prompt and allow you to make changes. Choose *Edit* for now.



The prompt is displayed in either *Structured* or *Freeform* view. Switch to the *Freeform* view.

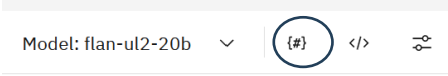


In the beginning of the prompt, we see the instruction for the LLM: *Read this Insurance claim description and extract the Car make and model, Location of the incident like street and date time if there is any mentioned. If you don't find these details in the description, please fill it as Not Found.*

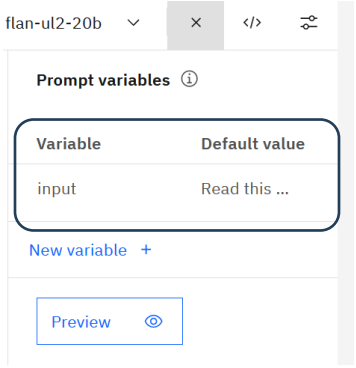
After that we have a few examples with text and extracted entities, which means that this is *a few-shot prompt*.

At the end of the prompt we have an *{input}* variable. This variable will be passed in during inference – it will contain the text (claim summary) from which the LLM will extract entities.

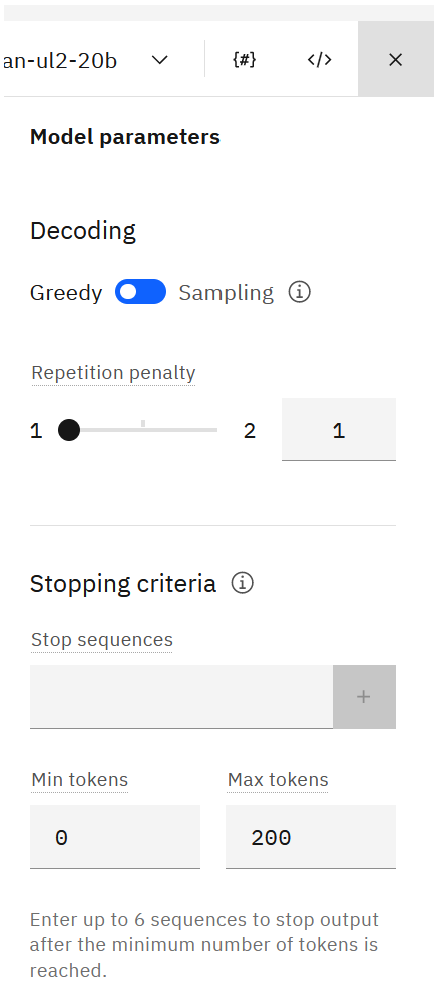
In the top right corner of the *Prompt Lab* we see that this prompt was created for the *flan-ul2-20b model.*



Next to the model we have the parameters button, where we can review and add parameters for the prompt. Notice that our parameter name is *input* (referenced in the prompt as {input})



Finally, we can view model parameters by clicking on the *Parameters* icon.

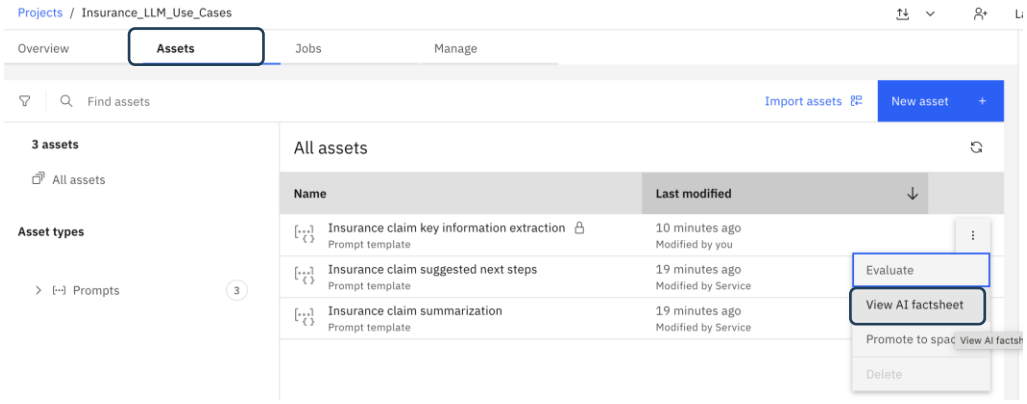


If you wish, test the prompt by clicking the **Generate** button. We get the results because the *{input}* variable has a default value (claim text).

{input}Car Details: Tesla Model S; Location: Parked outside; Date: April 15th, 2023

7. Return to the project view by clicking on the project name on the top of the screen.

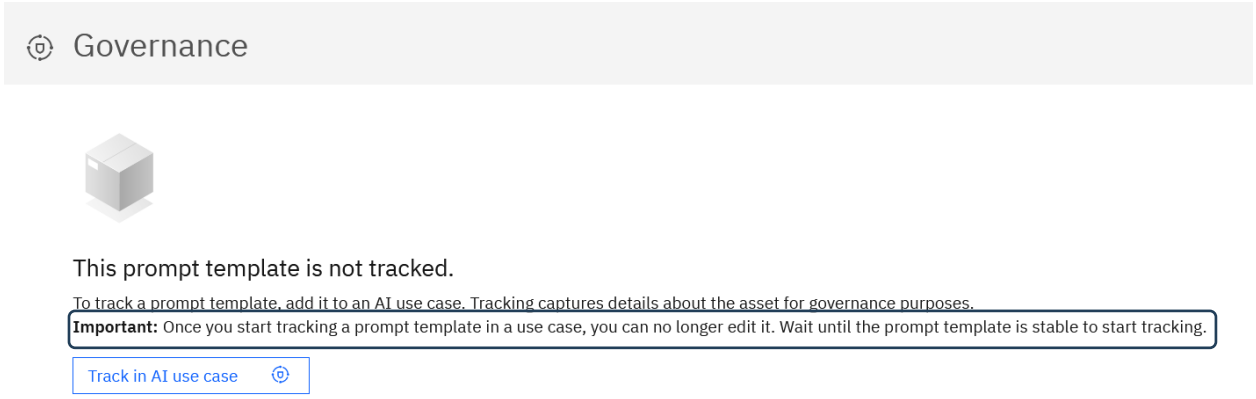
8. Expand **Prompts** on the **Assets** tab. Click on the vertical ellipses menu and select **View AI factsheet.**



Notice that the factsheet captures the details (prompt, model parameters) that you have used in the **Prompt Lab**. At this time, if you make a change in the prompt template using the **Prompt Lab**, it will be automatically reflected in the factsheet.

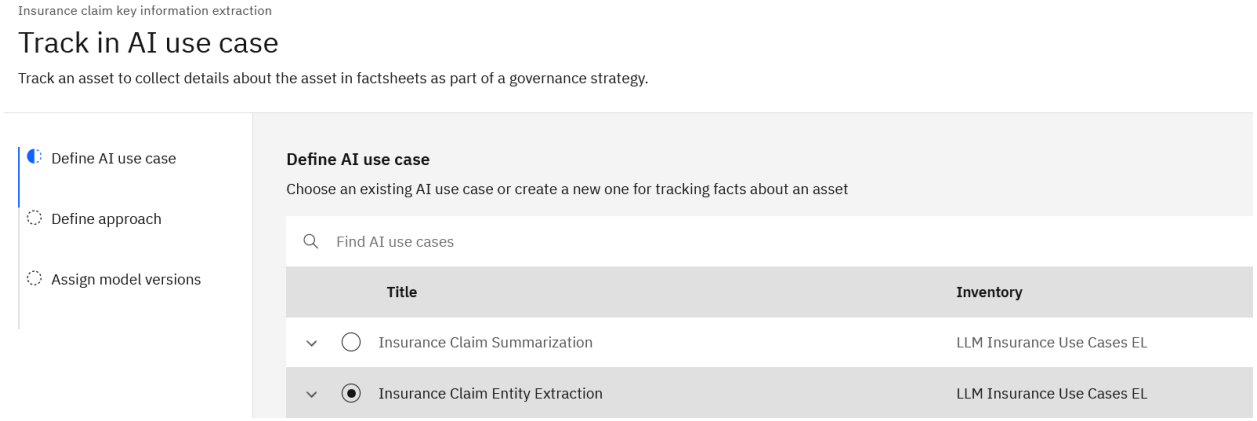
If you wish, in the Prompt Lab change the *max number of tokens* model parameter and verify that the changes are reflected in the factsheet.

Notice the comment on top of the factsheet: we can make changes to the prompt template until we start tracking it.



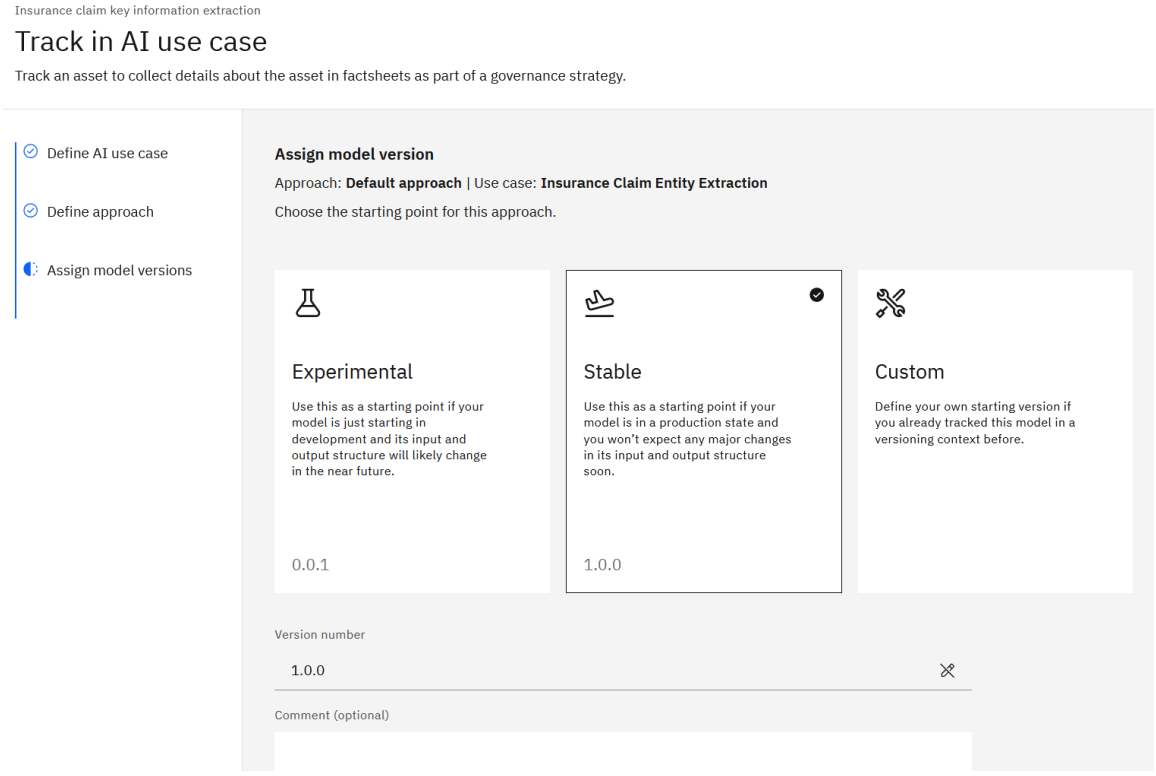
Since we made the assumption that our template is final, let's start tracking.

9. Click the **Track in AI use** case button and select the entity extraction use case you created earlier.



Click **Next** on **Define Approach** screen, then select **Stable** on the **Assign model version** screen. Click **Next**.

Note: we will review approaches later in the lab.



Click the **Track asset** button to finish this task.

Typically, a data scientist will perform testing prior to releasing the template for production. The *evaluation* process that we will complete in the next few steps is performed primarily for the purpose of documenting test results.

Evaluation metrics provided in watsonx are standard evaluation metrics that are used for various LLM tasks such as classification, summarization, and extractions. You can learn more about evaluation metrics in documentation.

Evaluation of LLMs requires "reference data", which is the "expected output" for the LLM. Reference data can be created manually or generated using various approaches. For generation use cases such as summarization and content generation, it is important to provide high quality reference data, which means that it may need to be created by experts.

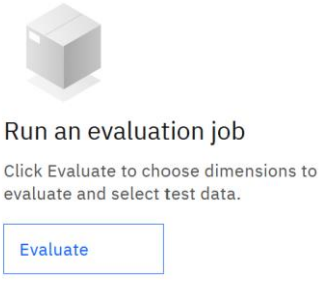
Example of reference data for information extraction use case:

Input Data	Reference Data
A car accident occurred on Jan 1st, 2023 at 5pm at the intersection of woodbridge. The insured vehicle, a Honda Civic, was hit by another vehicle that ran a red light. The insured driver, John, was driving within the speed limit and following all traffic rules. The accident resulted in significant damage to the insured vehicle, including a broken bumper and damaged front fender. There were no injuries reported. The insured is filing a claim for the repairs and any necessary medical expenses.	**Car Details:** *Honda Civic;Location: Woodbridge;Date: Jan 1st, 2023;Time of Incident: 5pm*

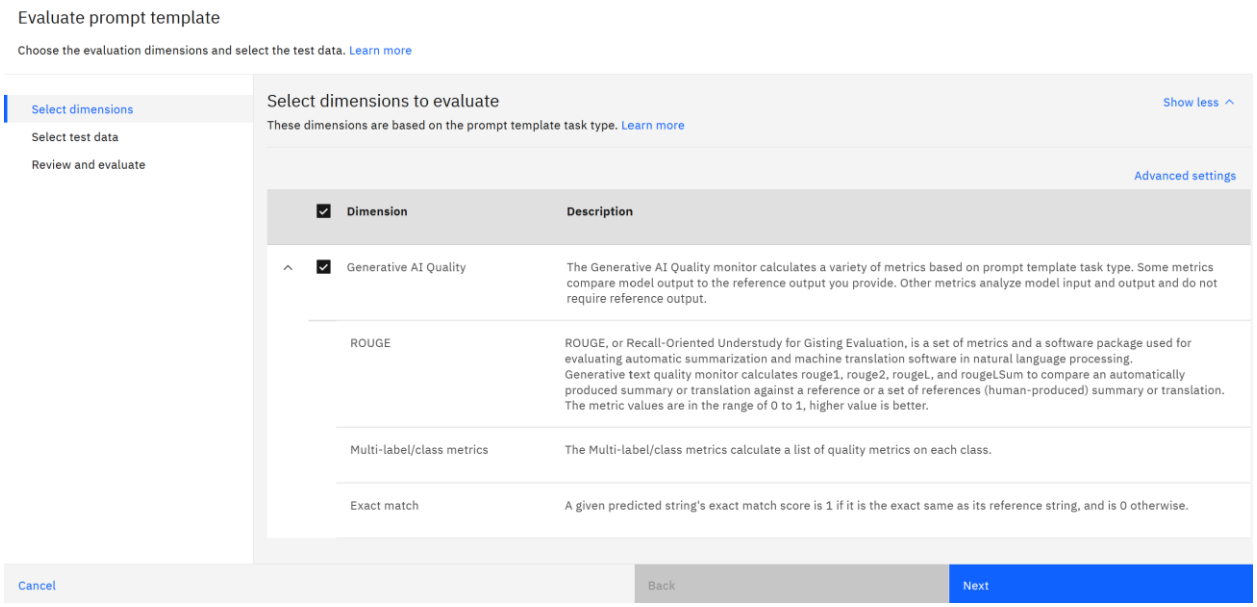
You can find reference data for our use cases in the *lab repo/Test Data* folder.

Evaluation can be done either from the Factsheet view (bottom of the page) or the ellipses menu next to the prompt in the project view.

10. Click **Evaluate** in the factsheet, then **Evaluate** on the **Run an evaluation job** screen.

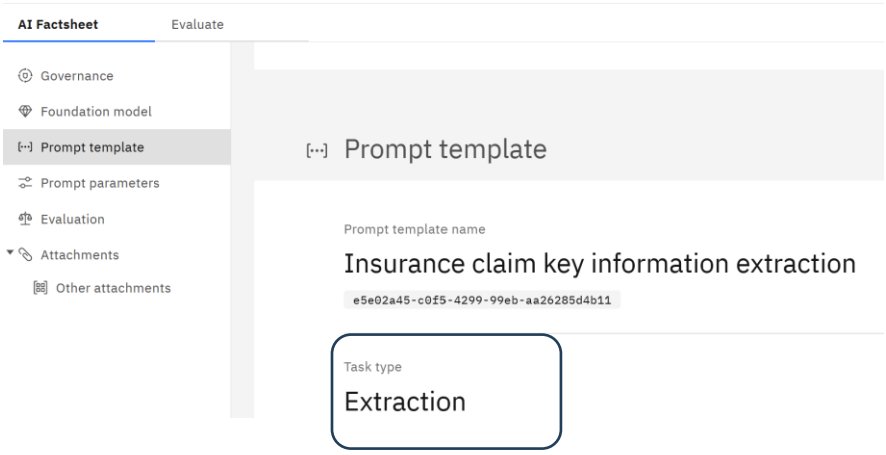


Notice that evaluation metrics relevant to our use case, *extraction*, were automatically selected.



As a reminder, the task type is captured in the factsheet.

Note: The task type is specified when we save a prompt. Since we imported a prompt, we did not need to complete this step.

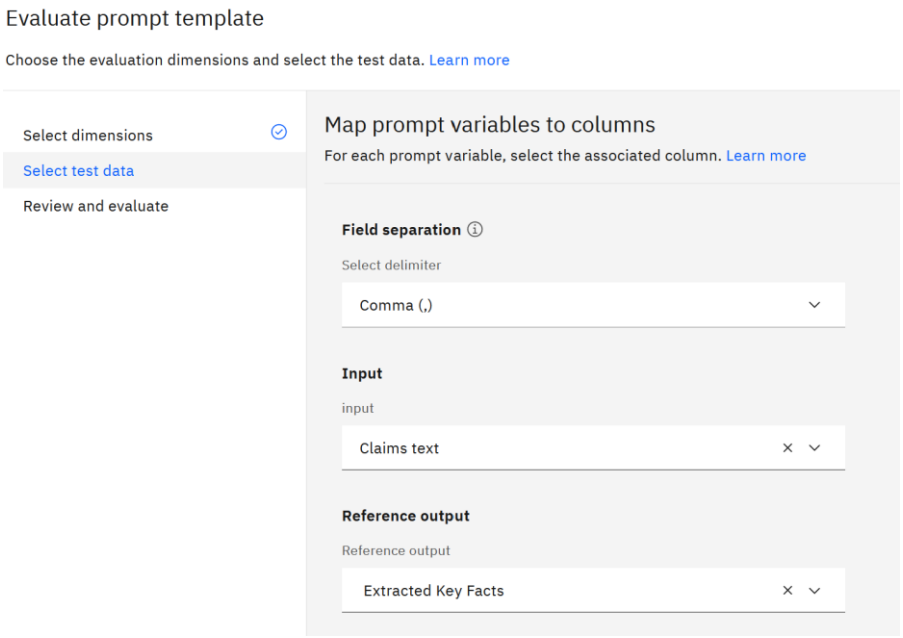


Click **Next** on the *Evaluate Prompt Template* screen.

11. On the select data screen, navigate to the *lab repo/Test Data* folder and select the *text_extraction_claims.csv* file.

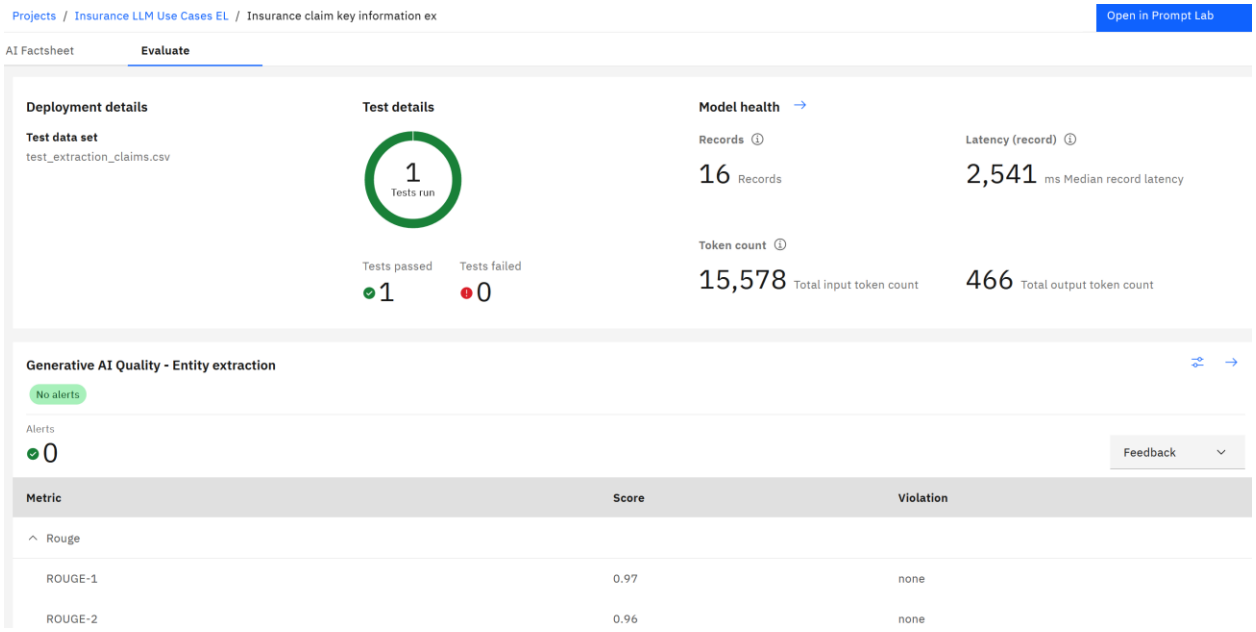
    If you wish, open this file to review its content. As we discussed earlier, this file provides "expected LLM output" for the instruction that we are asking it to perform.

    Select *Claims text* and the input and *Extracted Key Facts* as reference output.



Click **Next**.

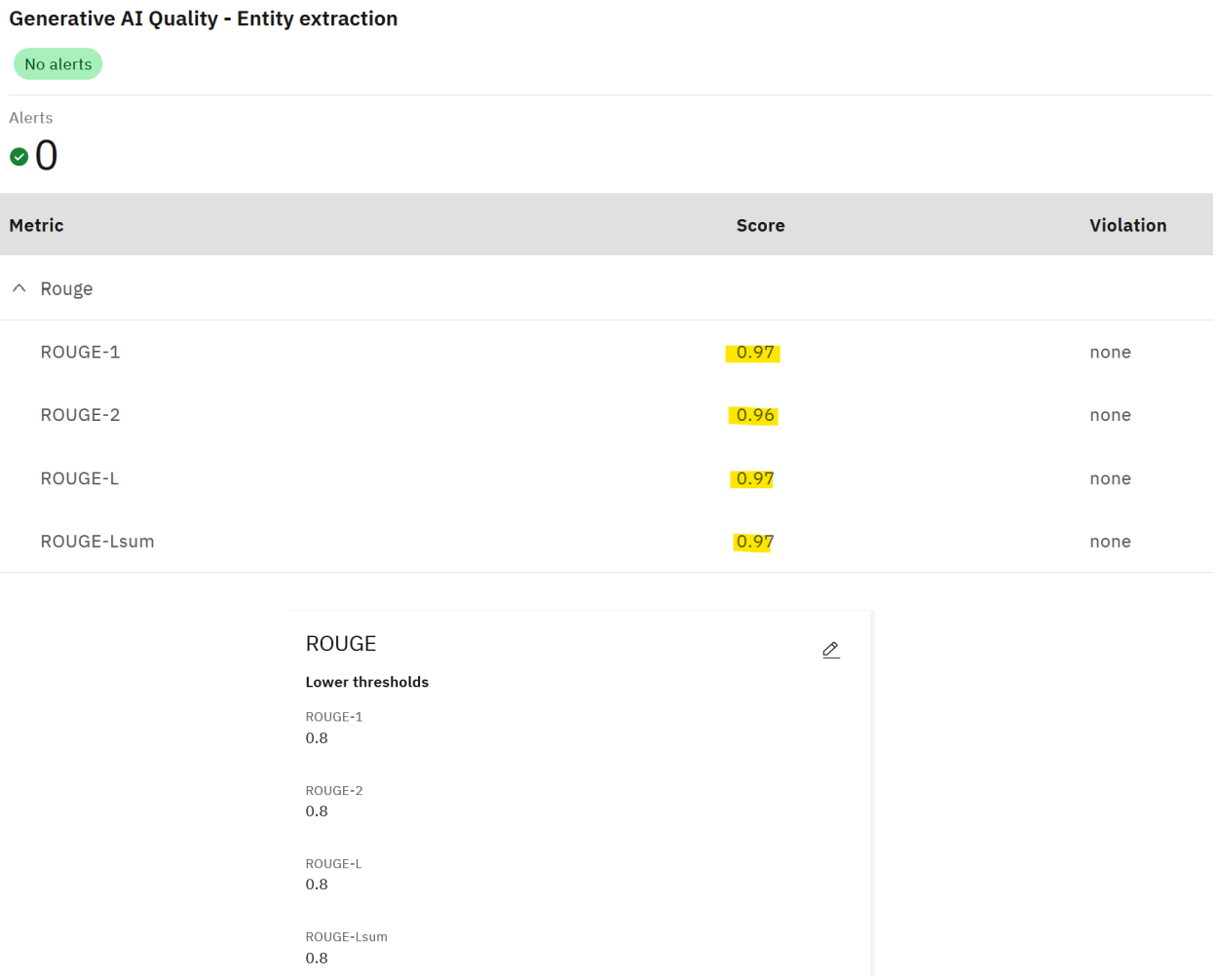
12. Click **Evaluate** to run evaluation.



Notice that the test result is *passed*, which means that the calculated evaluation metrics are above specified thresholds. The *passed/failed* metric helps data scientists and auditors make sure that only models that meet accuracy requirements are deployed into production.

To view the thresholds, click on the **Configuration** icon.

Compare the *rouge* metric thresholds defined on this screen with the values from our test run.



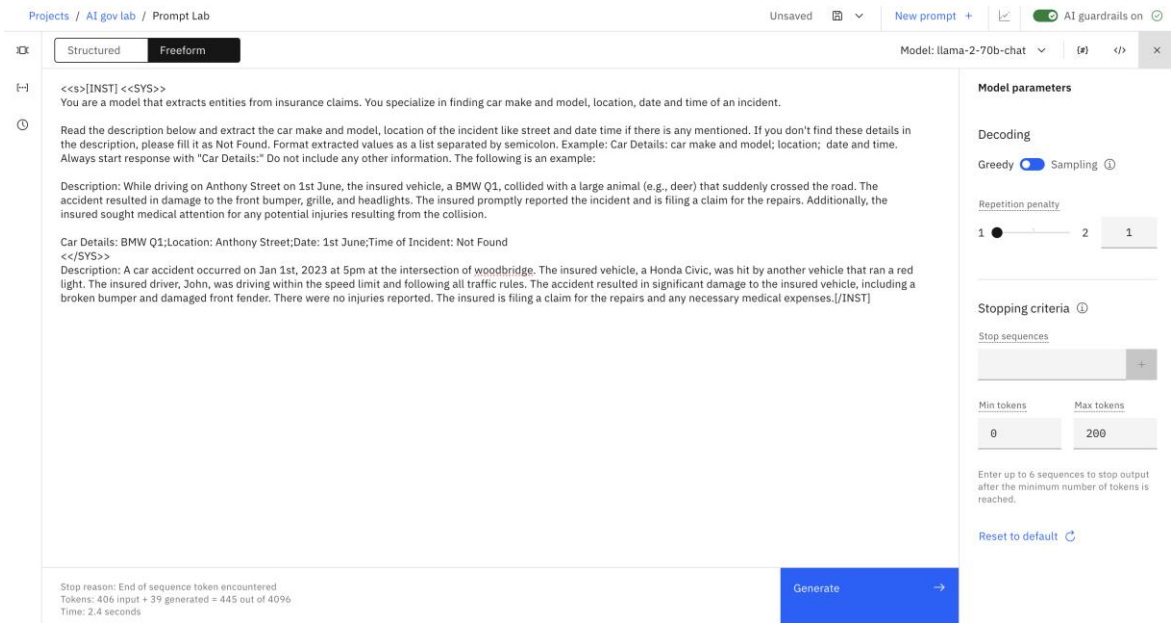
Our test is marked as passed because all values are above the thresholds.

Next, we will create another prompt template to understand prompt format that is required for *watsonx.governance* implementation.

13. Locate the sample prompt *Extract_info_insurance_claim_llama* in the *lab repo/Prompts folder.*

14. Navigate to your project and create a new prompt. In the **Prompt Lab** use the *Freeform* view to paste the sample prompt.

    Make sure to select the *llama-2-70b-chat* model and set max tokens to *200*.



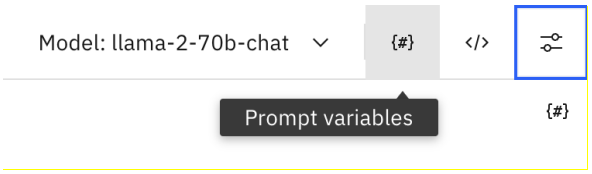
Notice that this prompt has been formatted specifically for *llama* (with *INST* and *SYS* tags). It is also a one-shot prompt, which, compared to the first example, will utilize less tokens for inference.

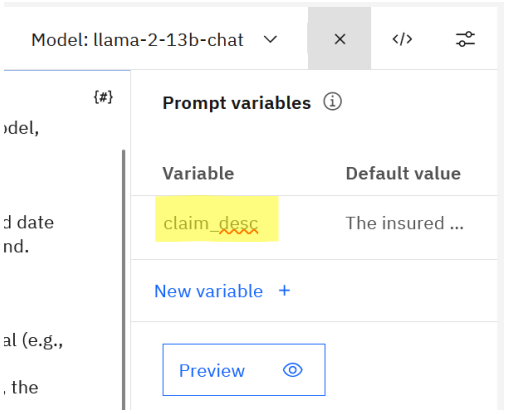
15. Click the **Generate** button to test the prompt.

    Notice that the **Evaluate** button (or icon) is disabled for this prompt. In order to evaluate, track, and deploy this prompt, we need to add a *parameter* – the input that will be passed to the prompt.

    Note: You will also not be able to track or deploy prompts without parameters.

16. In the **Prompt variables** view of the **Prompt Lab**, add the *claim_desc* parameter.



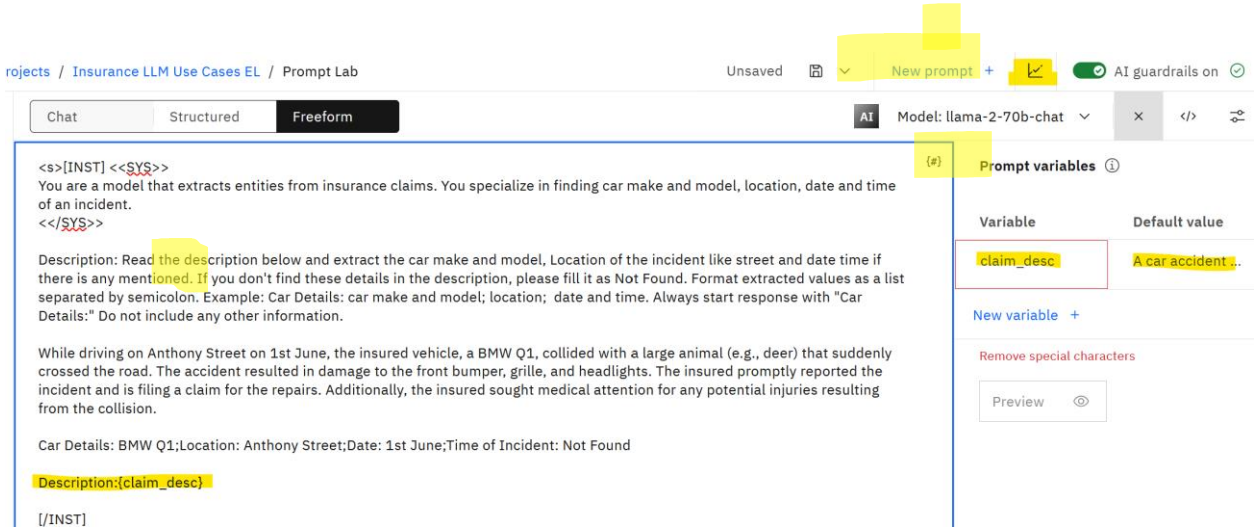


You can use the following value as the default value:

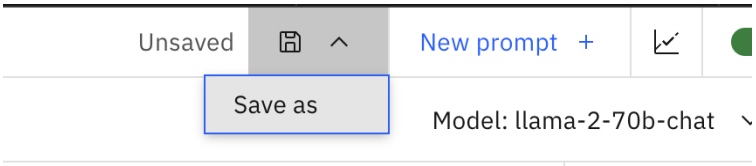
A car accident occurred on Jan 31st, 2023 at 5pm at the intersection of woodbridge. The insured vehicle, a Tesla Model Y, was hit by another vehicle that ran a red light. The insured driver, John, was driving within the speed limit and following all traffic rules. The accident resulted in significant damage to the insured vehicle, including a broken bumper and damaged front fender. There were no injuries reported. The insured is filing a claim for the repairs and any necessary medical expenses.

If you wish, you can provide a different claim description from the test file we used earlier.

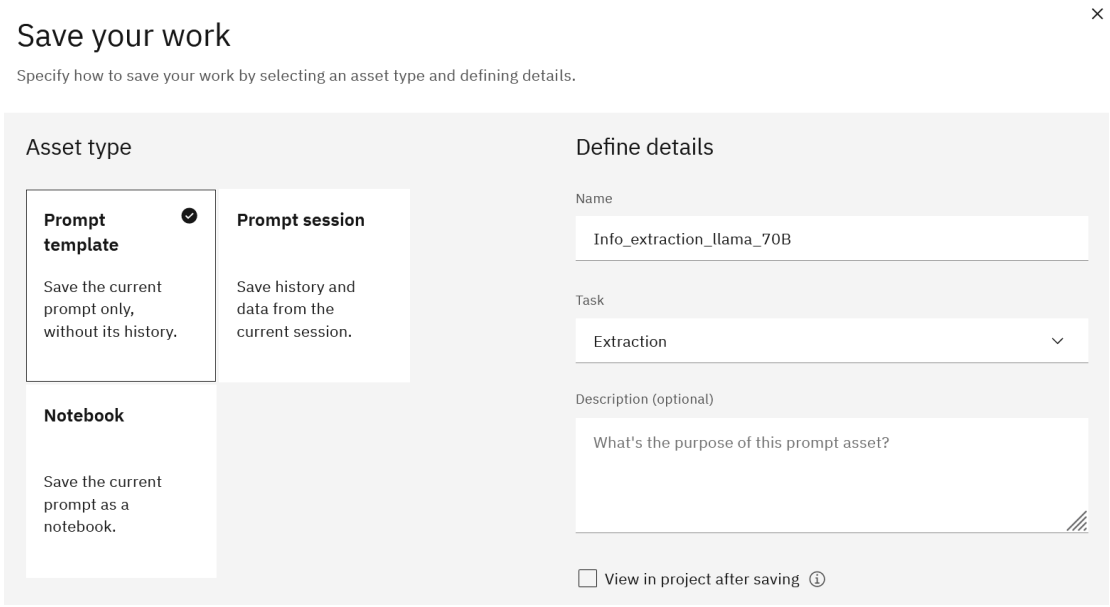
Next, add the word *Description* and the *parameter* enclosed in curly brackets to the prompt. Notice that now the **Evaluate** button is enabled.



17. Save the prompt template by clicking the save icon then *save as*. Make sure to select *Extraction* as the task type.

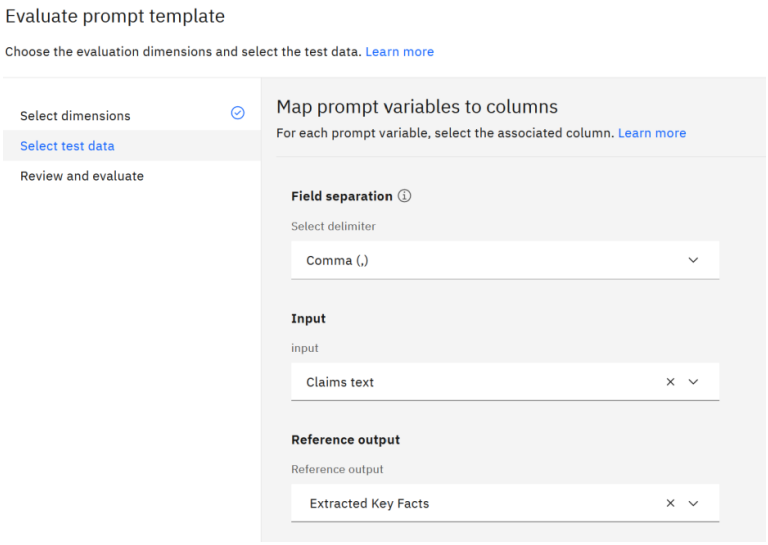


Important Note: At this time the task type will automatically determine which evaluation tests will be run. You will not be able to change the task type after saving the template.



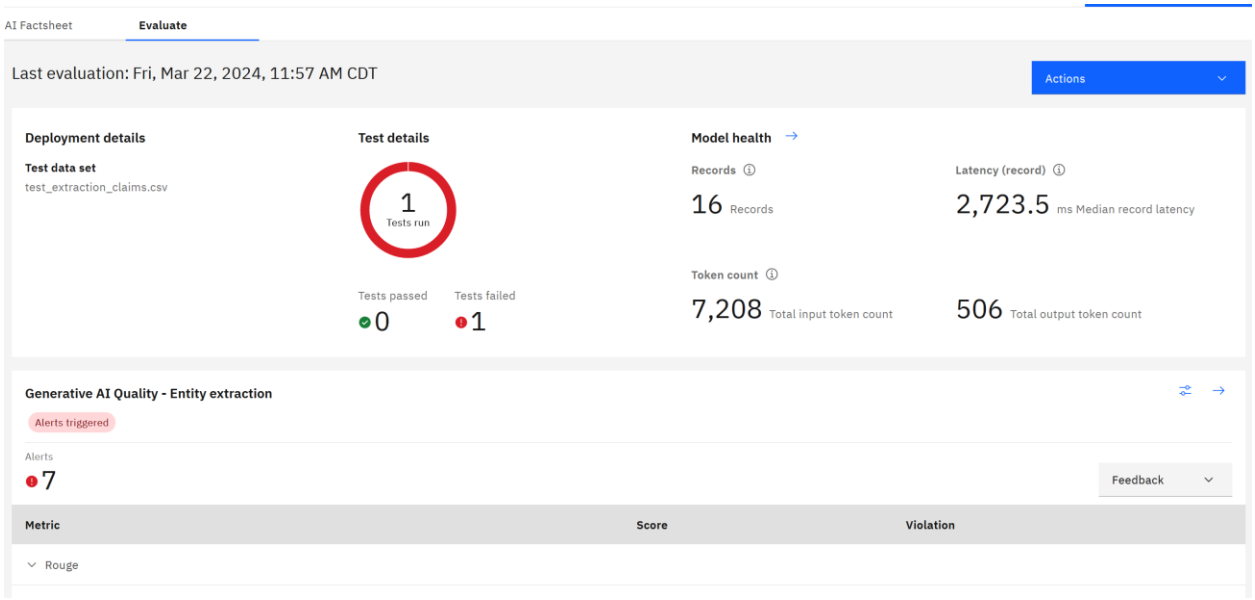
18. Click the **Evaluate** icon to run evaluation for this prompt using the same dataset and values as in the previous prompt evaluation.

    Test dataset: *text_extraction_claims.csv* file in the *lab repo/Test Data* folder



With this template, some of the test results are below the thresholds. If you wish, investigate individual metrics to understand which ones have failed the test.

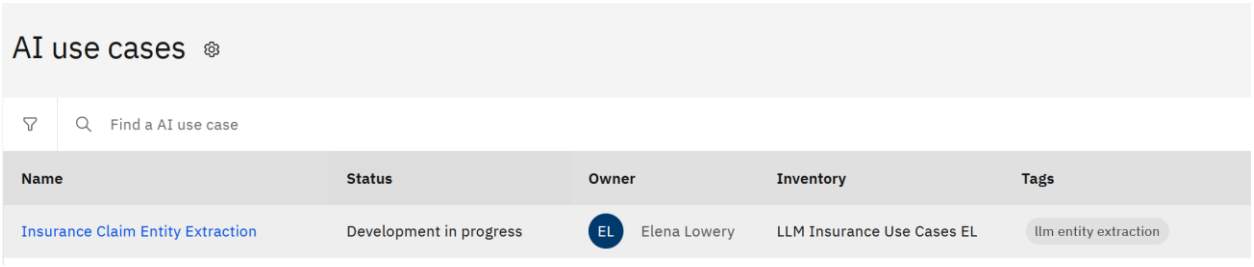
In general, the prompt template doesn't have to pass every test to be ready for production. Usually, a data scientist can decide which tests are applicable for their use case and set the appropriate thresholds.



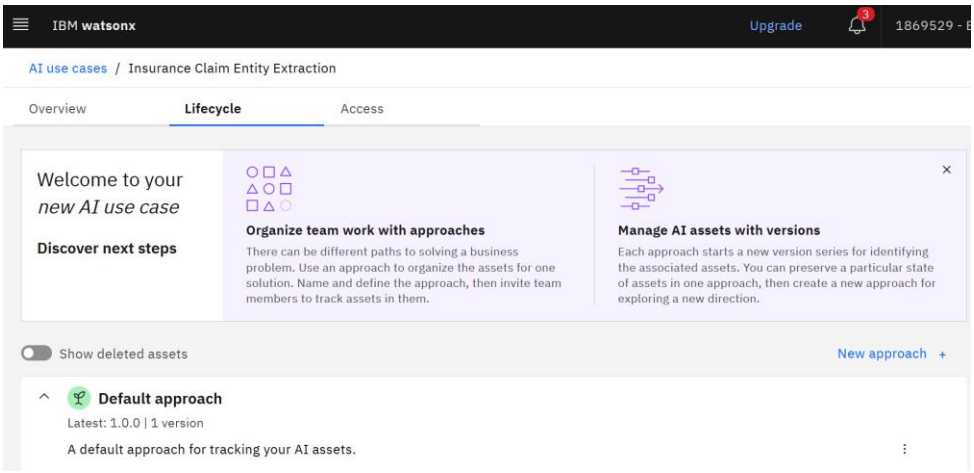
Next, we will enable tracking for this template following the same steps as you have done for the other prompt.

Before we do that, we will create a new *approach* for our use case. Approaches help us organize different *implementations* for the same use case. For example, some companies may do A/B testing to find out which prompt/model delivers the best performance.

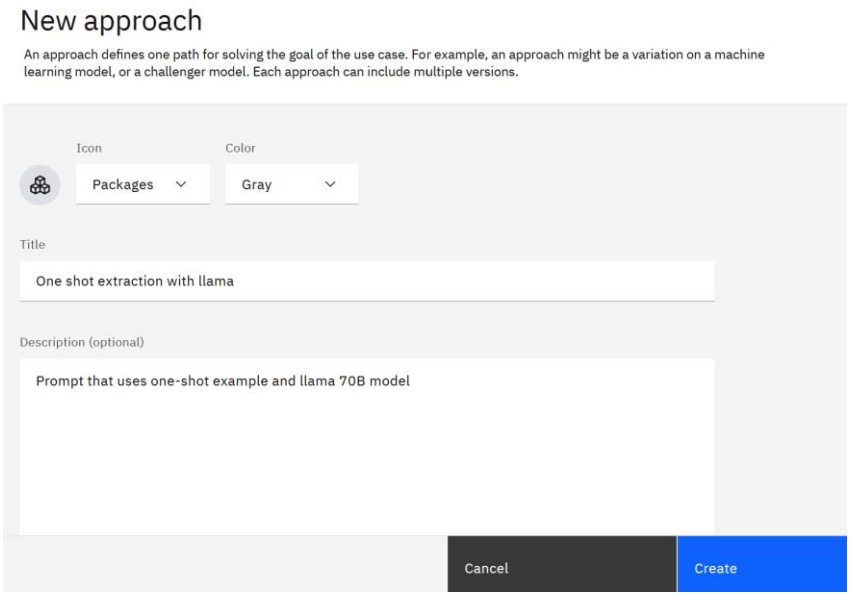
19. Navigate to the *AI use cases* view and select your use case.



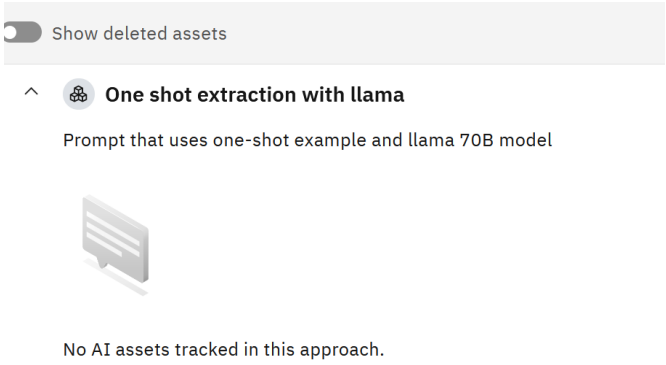
20. In the **Lifecycle** view click **New Approach**.



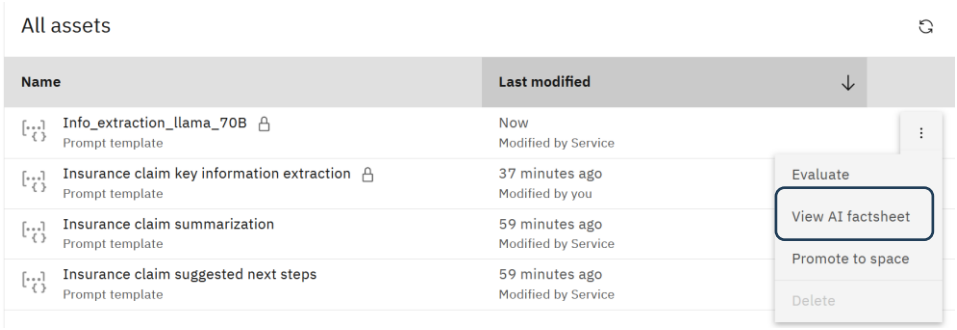
21. Create the new approach using values similar to the ones shown in the screenshot and click **Create**.



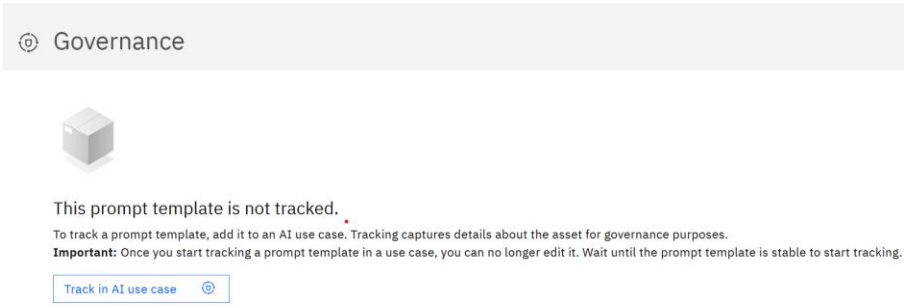
At this time, we do not have any assets tracked for this approach.

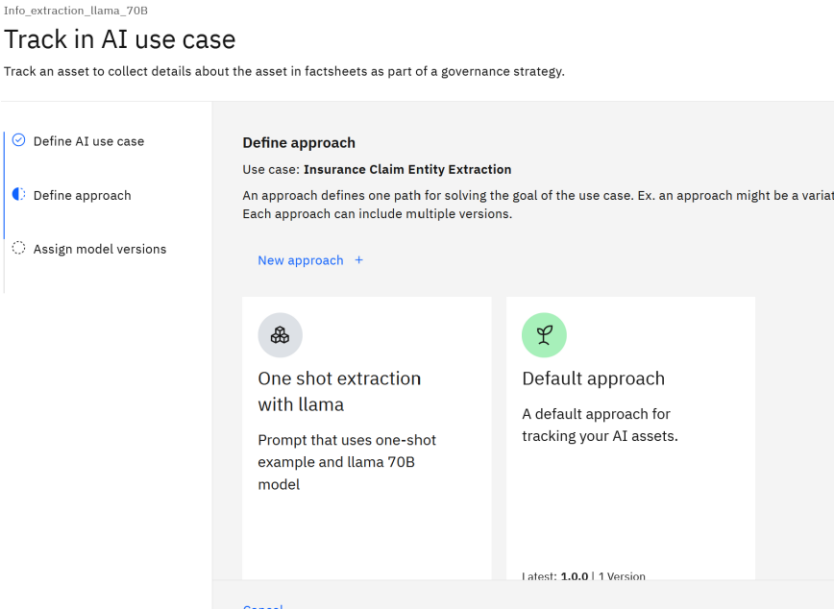


22. Navigate to the **Factsheet** view of your *llama* prompt (from the project **Assets** view).

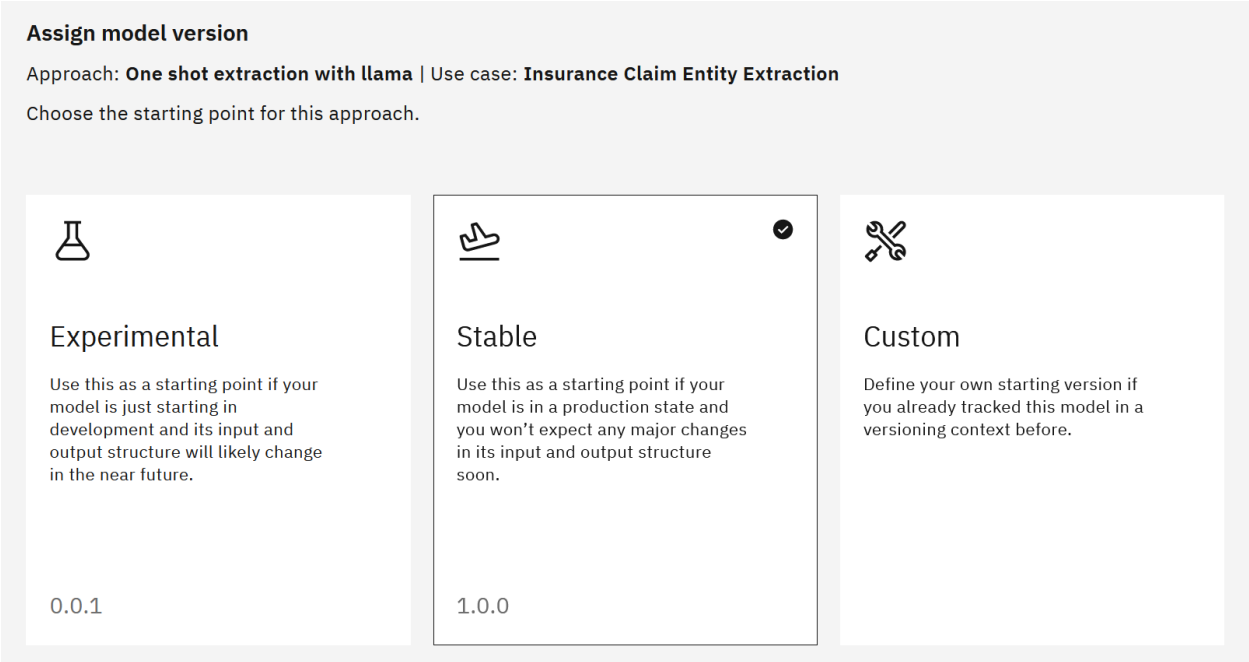


23. Click **Track in AI use case**. Select your use case and the newly created approach.





Select the *Stable* model version, click **Next**, then click **Track asset**.



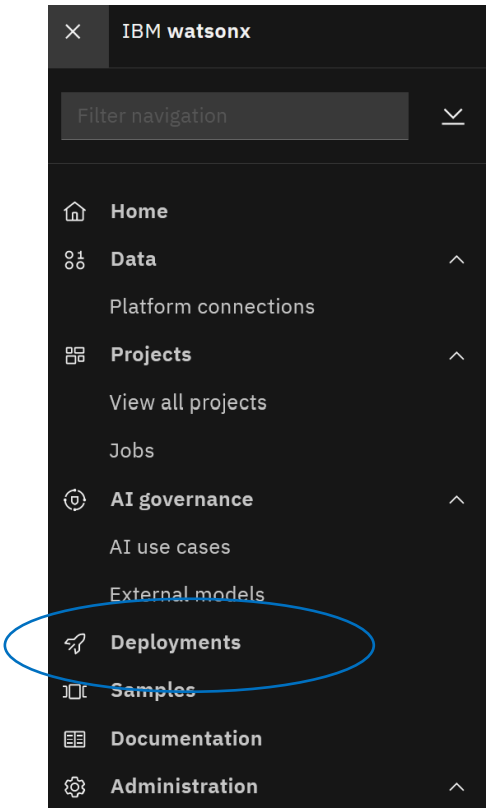
If you wish, come back to the **AI use case** view, and verify that both templates are now tracked – one under the *Default* approach, and one under the custom one that we created.

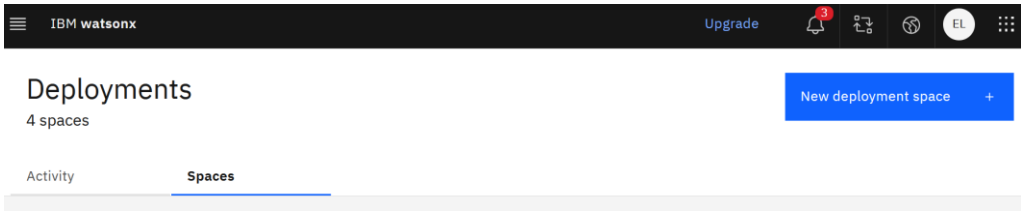
Next, we will promote both prompt templates to production. In this lab we will skip the validation step. Validation is similar to model evaluation that we have done in this lab, but it is usually done by a different team in a different *watsonx* project. If you would like to learn more about the validation step, you can review documentation.

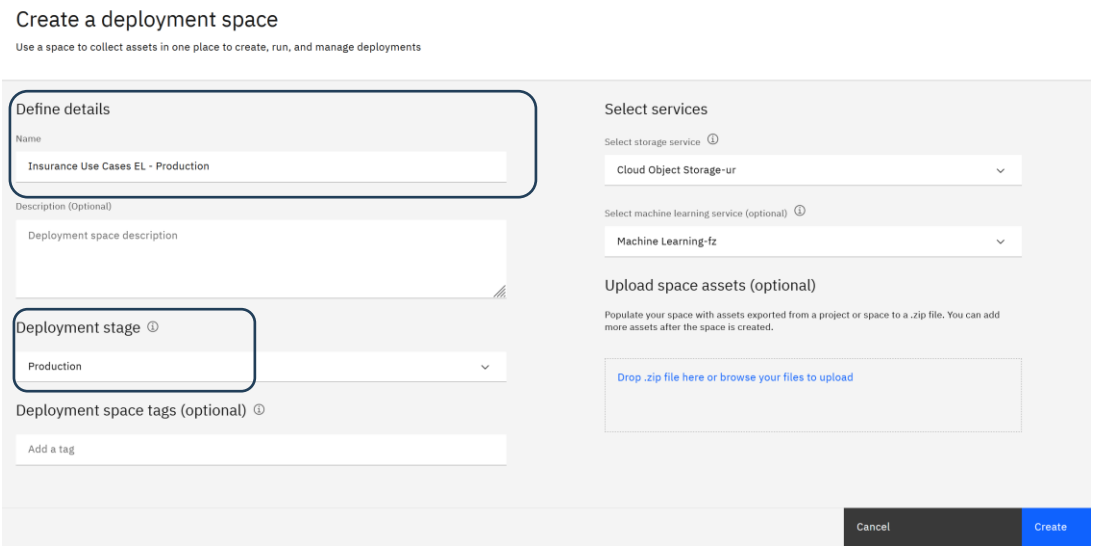
24. From the main menu navigate to **Deployments** and create a new deployment space using values similar to the following screenshot.

Notes:

- *Make your deployment space name unique by adding your initials.*
- *Your machine learning service name will be different.*

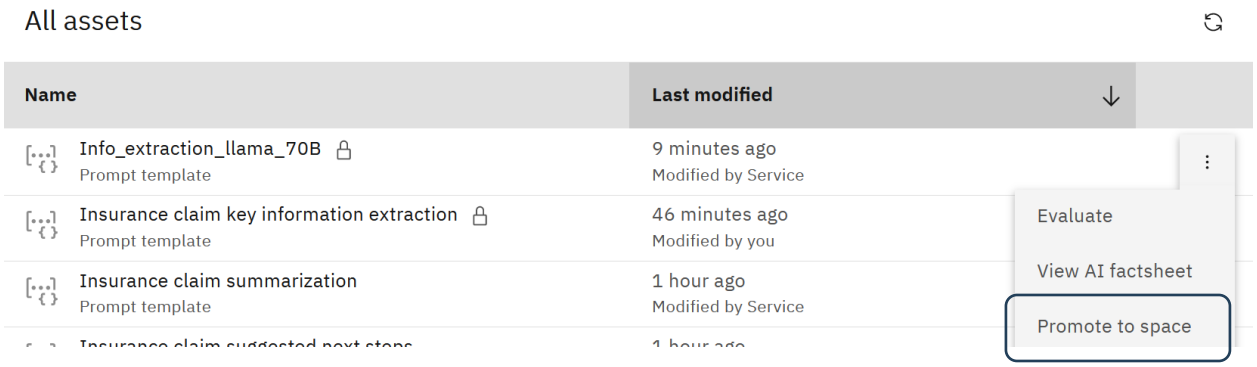


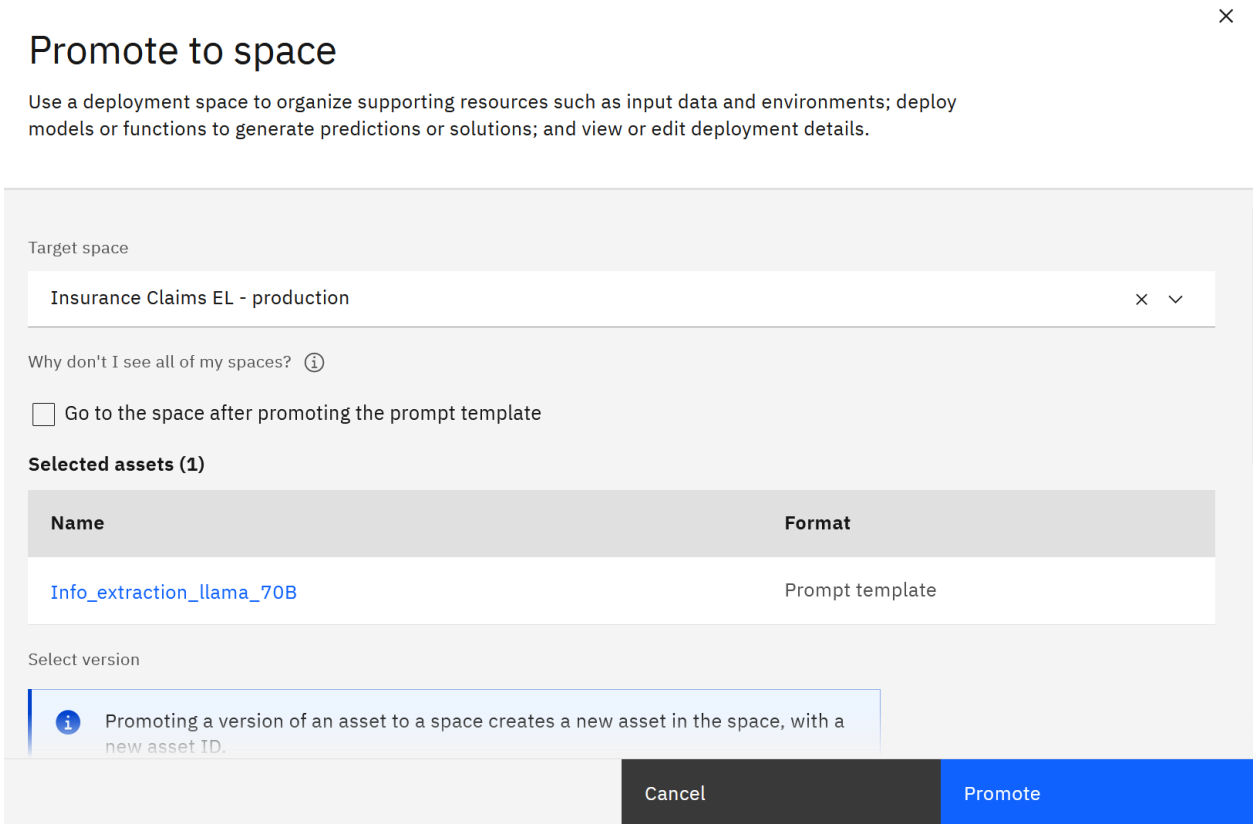




25. Navigate back to your projects. From the **Assets** view of the project, select **Promote to Space**.

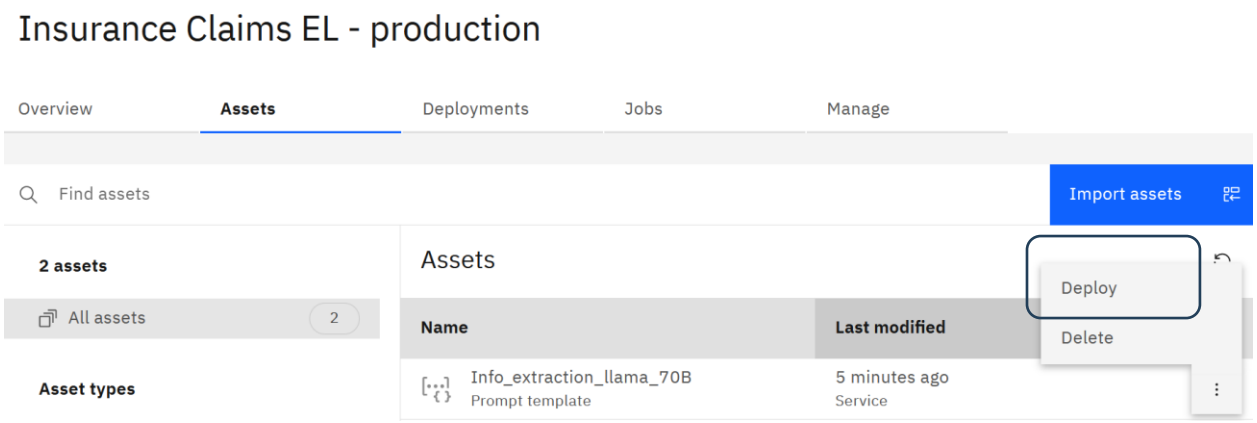
26. Select the space that you previously created and click **Promote**.



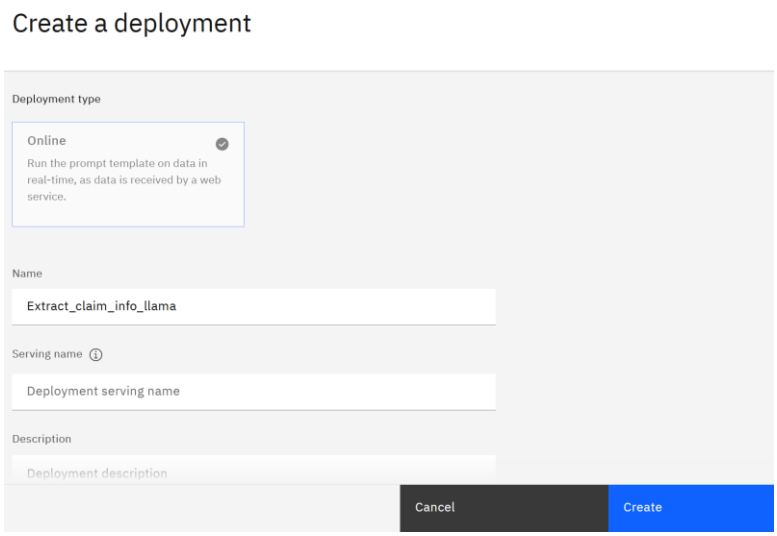


27. Navigate to the **Deployment space**. Next, we will deploy both templates.

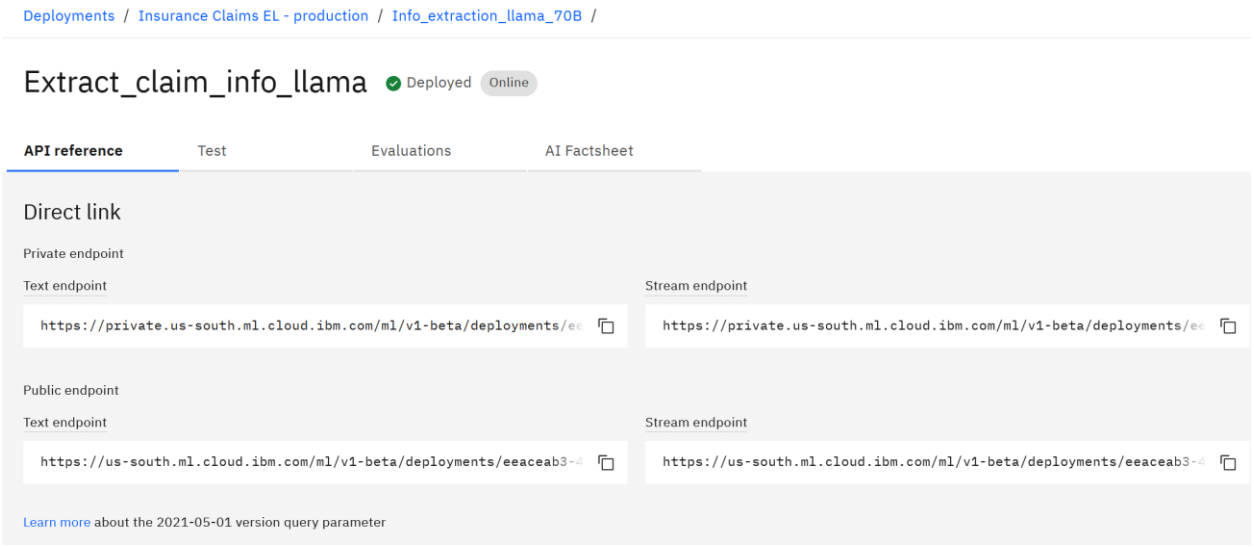
28. In the **Assets** view of the space, click on **Deploy** next to the template.



Provide a deployment name and click **Create**.

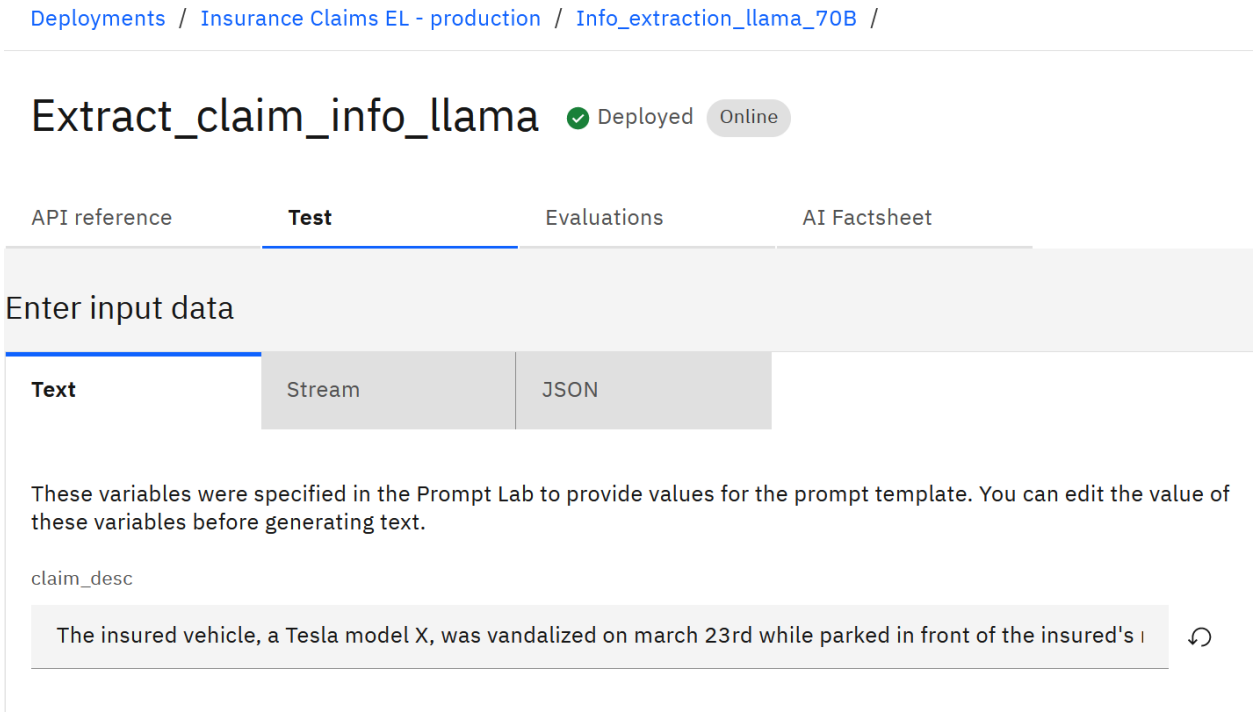


29. Click on the created deployment and explore the various tabs.



On the **Test** tab we can interactively test the prompt.

If you wish, you can copy one of the claim description values from the provided csv file that was used for evaluations and test the LLM.



The **AI Factsheet** tab shows the same information as the factsheet in the project.

30. If you wish, repeat the same steps to deploy the 2nd template.

Next, we will walk through the process of setting up *Evaluations*.

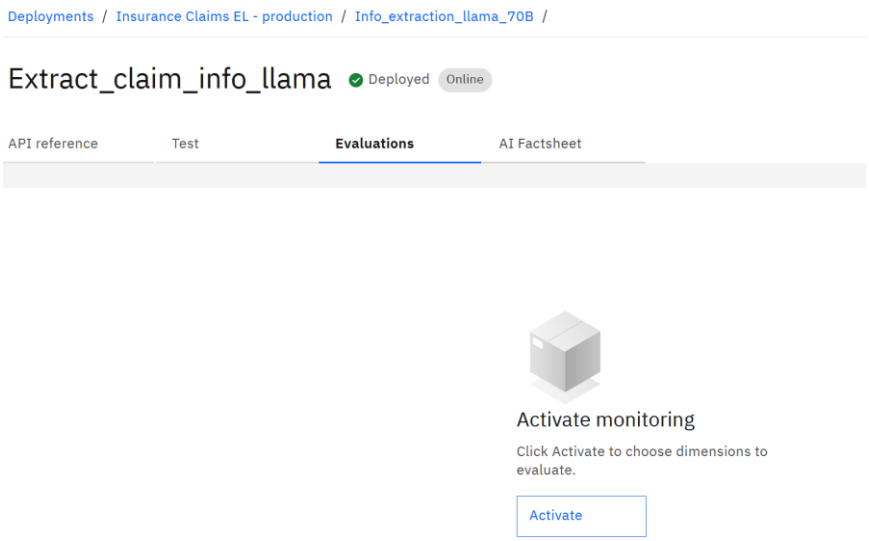
The process of evaluation, whether it is done manually in a project, manually in the *Deployment Space*, or automatically is always the same: generated output is compared with the reference data. The only difference is how generated and reference data are provided.

In **Deployment Spaces**, we can set up automatic payload logging (payload data is the data that is passed in to the LLM). You can find more information about setting up payload logging in documentation.

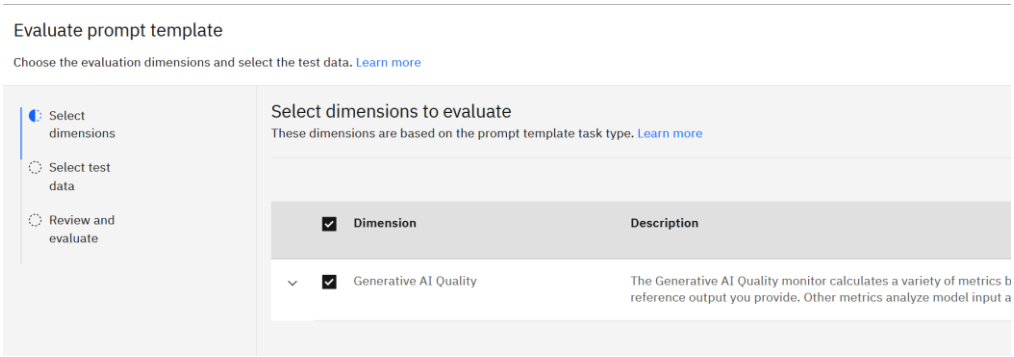
Payload logging is not in the scope of this lab. Since we do not have payload logging configured, we can go through the setup process and run manual evaluation.

You can run evaluation for one or both of the deployed templates.

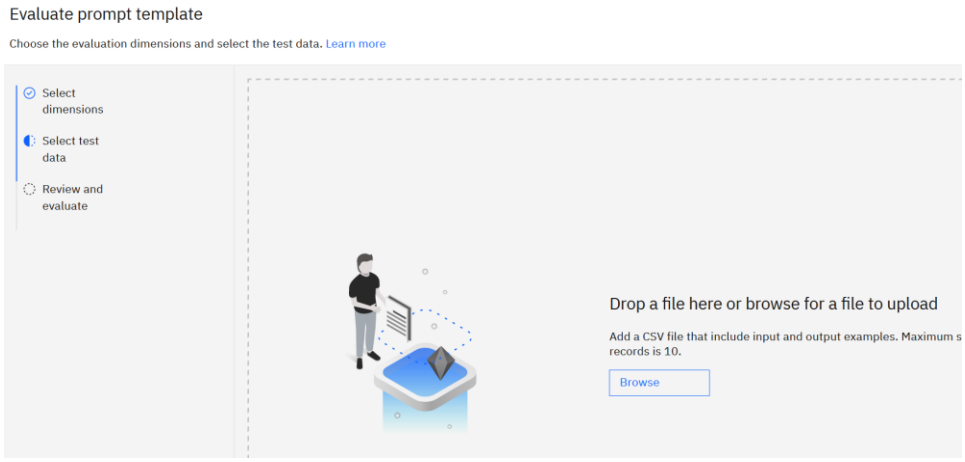
31. On the *Evaluations* tab click **Activate**.



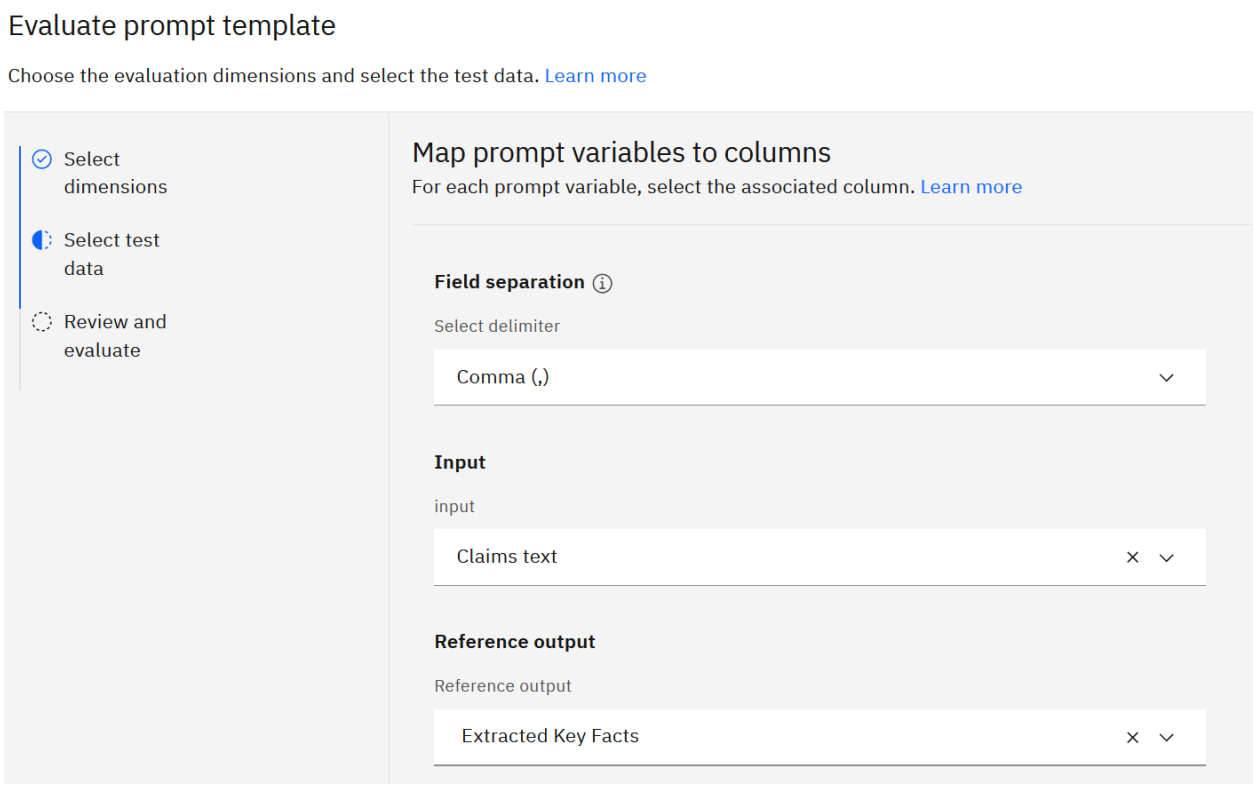
32. Click on **Next** on the **Select dimensions** screen (accept defaults).



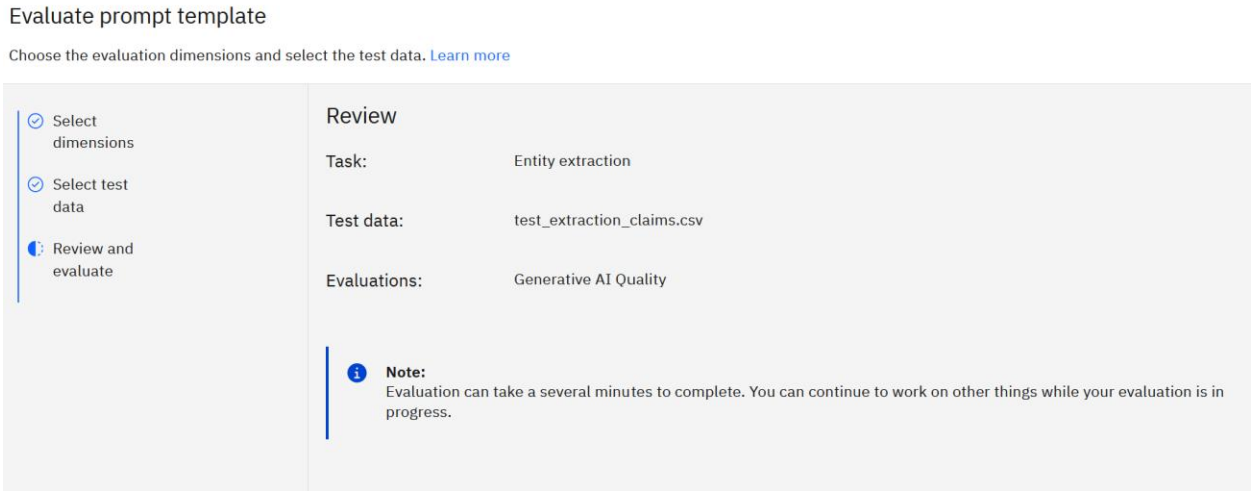
33. On the **Select test data** screen browse to the *lab repo/Test Data folder* and select the *test_extraction_claims.csv* file.



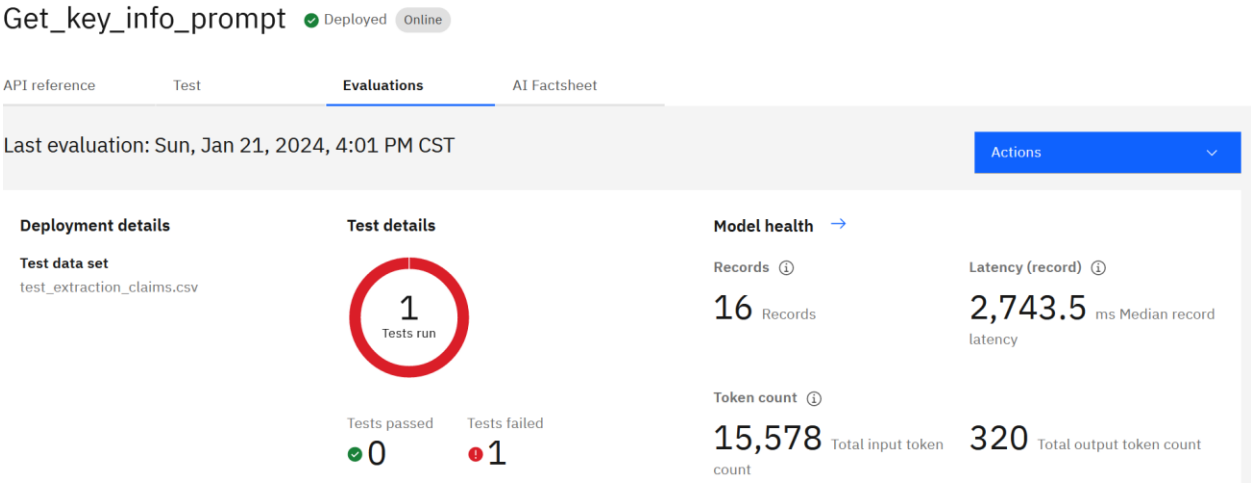
34. Similar to the evaluation configuration in the project, select the *Input* and *Reference* output fields as showing the following screenshot.



35. Click **Next**, then click **Evaluate**.



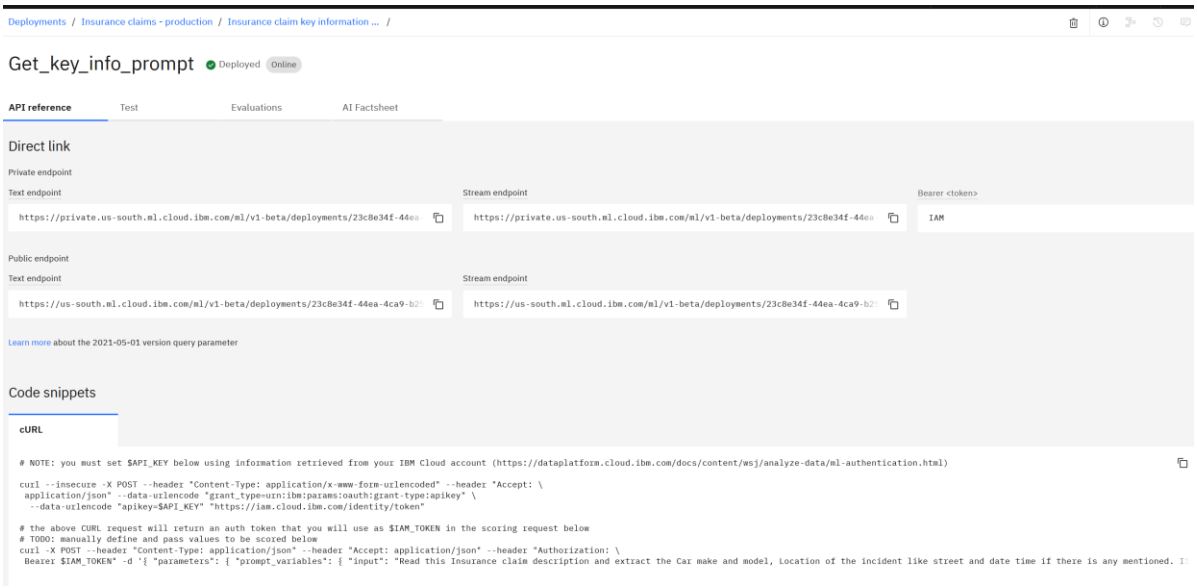
Review evaluation results.



While this evaluation process is similar to the process you completed in the project, it has a different purpose. *Deployment Space* is a production environment, and here the evaluations capture measurements for prompts that are currently in production, which is important for understanding model performance, transparency, and auditing.

Next, we will invoke the deployed template from a client application. If you would like to do a quick test, you can use a notebook that is running in *watsonx*. However, in production deployment, the client application will be running outside of watsonx. We will explain where to find and how to modify the REST call for invoking the template, and you can decide how you would like to test it.

36. In your *Deployment Space* project click on the **API reference** tab.

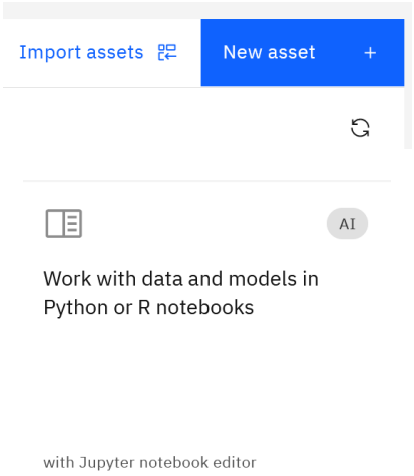


If you would like to do a quick test with a notebook, complete the next twos step. You can also skip these steps and go the step of testing prompt template invocation with a Python script.

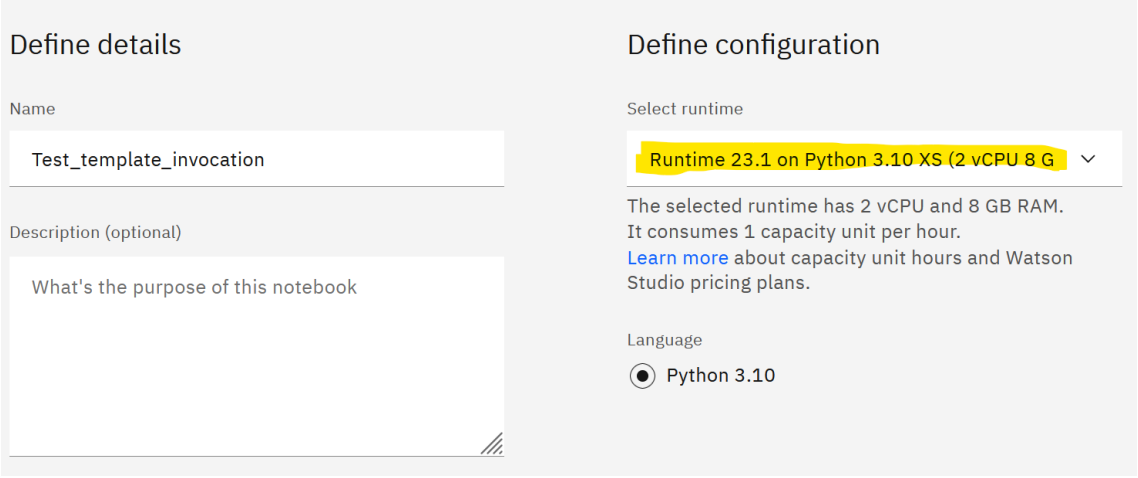
Note: if you typically don't work with the REST API, we recommend that you skip the next 2 steps.

37. Create a new notebook in your watsonx project (open a new browser tab).

    - In the **Project** view select **New Asset -> Work with data and models in Python or R notebooks**



    - Select the default Python environment.

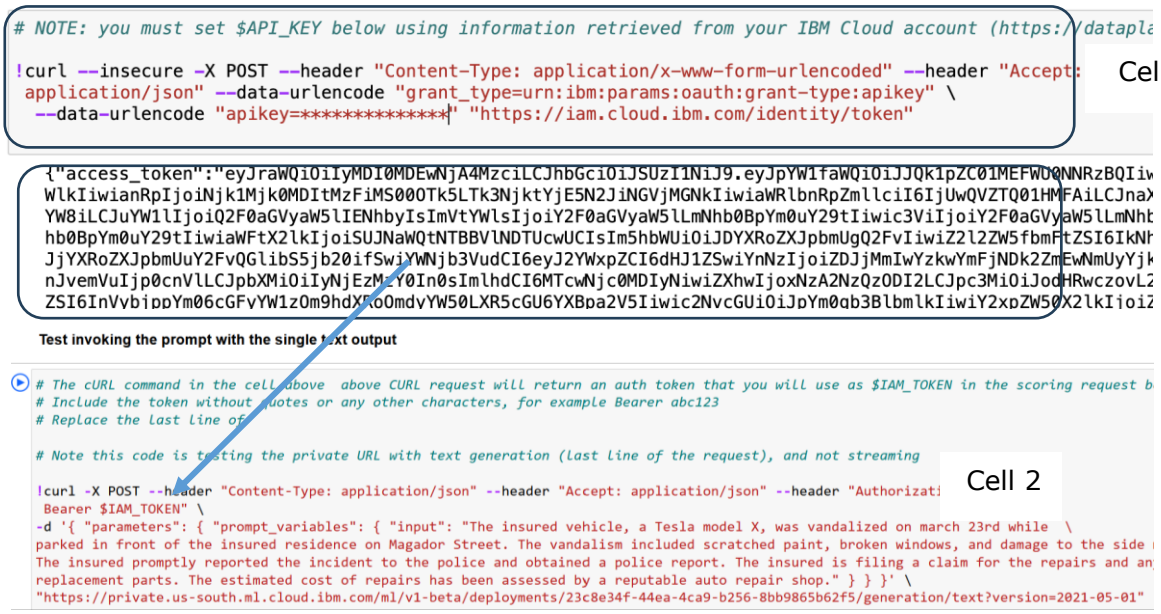


38. Copy the code from the *Code snippets* into the notebook.

    After the initial copy, break up the code for getting the token and invoking the prompt into 2 cells. We need to this to get the token value that we will paste into the invocation cURL command.

Update the first cell with your API key, run the cell, then copy the token into the cURL command in the 2nd cell.

Note that the sample code in deployment points to the streaming text URL. If you wish, you can replace it with the text endpoint URL, which you can find on the same **API reference** tab.



In the next step you will run a Python client script that invokes the template. This code invokes the same deployed template as the notebook, but we refactored the code to make it easier to understand and maintain.

To complete this step, you will need a Python IDE environment (VS Code, PyCharm, etc.). If you don't have a Python IDE, then you can watch a video in the *lab repo/ Reference folder.*

39. Find the following Python scripts in the downloaded lab repo */Scripts* folder:

    - *demo_invoke_template.py*

Let's update and review the script.

On top of the script replace variables with your IBM cloud API key and the public URL of your deployed template.

```
import requests, json

# Replace with your IBM Cloud API key
cloud_api_key = ''
# In most cases the URL for authentication should be this value.
# If you get an authentication error, check the URL in IBM Cloud
auth_url = 'https://iam.cloud.ibm.com/identity/token'
# Make sure to provide public, text URL (not private and not straming)
prompt_url = ''
```

The script has the following functions:

- *get_credentials():* generates the authentication token
- *invoke_prompt()*: invokes the prompt
- *demo_invoke_prompt():* invokes all other functions for testing

40. Run the script. The output will be shown in Python terminal.

```
C:\ProgramData\anaconda3\envs\Python310\python.exe C:\Users\1A3030897\PycharmProjects\LLM_Workshc
C:\Users\1A3030897\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py:1056:
  warnings.warn(
The access token is: eyJraWQiOiIyMDI0MDEwNjA4MzciLCJhbGciOiJSUzI1NiJ9.eyJpYW1faWQiOiJJQk1pZC0xMTA
The generated text is: Car Details: Tesla Model X;Location: Magador Street;Date:

Process finished with exit code 0
```

You have finished testing the deployed prompt template.

Conclusion

You have finished the **AI Governance in watsonx** lab. In this lab you learned:

- Best practices for organizing AI use cases
- Tracking prompt template lifecycle in watsonx
- Invoking the deployed prompt template from a client application.