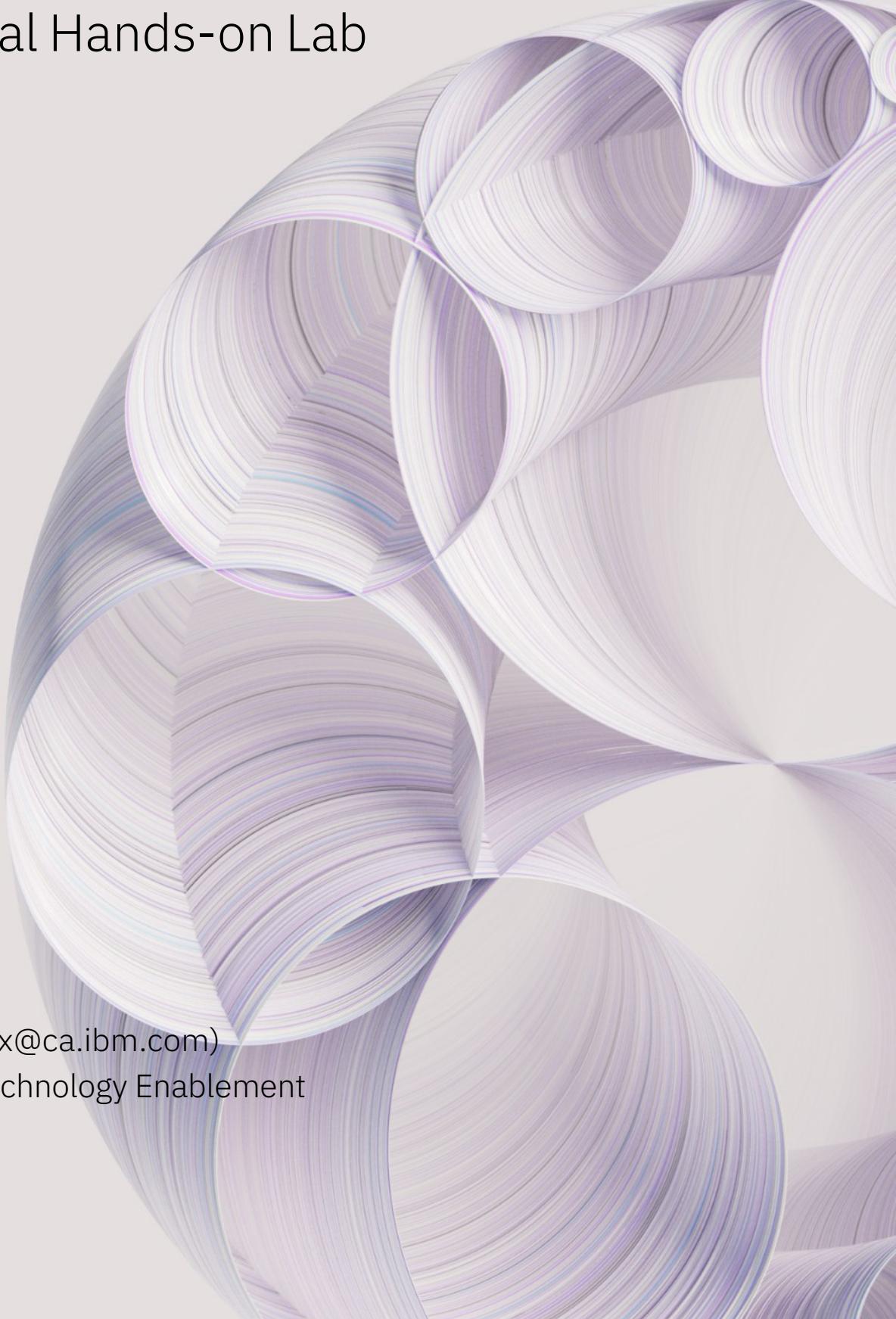


# IBM watsonx.ai

Technical Hands-on Lab



Felix Lee ([felix@ca.ibm.com](mailto:felix@ca.ibm.com))  
Worldwide Technology Enablement

# Contents

1. Introducing watsonx.ai .....	3
2. About this Lab .....	4
2.1 Disclaimer.....	4
3. Getting Help.....	6
4. Prerequisites & Getting Started .....	7
5. Watsonx.ai console .....	18
6. Watsonx.ai Prompt Lab .....	20
6.1 Basic Navigation .....	20
6.1.1 Open up a new Prompt Lab session .....	23
6.2 Exploring different foundation models and their characteristics.....	25
6.3 Prompt Engineering – updating a zero-shot prompt .....	39
6.4 Prompt Engineering - modifying parameters.....	47
6.5 Working with lists.....	58
6.6 Working with JSON .....	65
6.7 Working with one-shot prompting .....	71
6.8 Saving prompts and prompt sessions .....	77
6.9 Using Jupyter notebooks with prompts.....	92
6.9.1 Getting your API key.....	92
6.9.2 Creating a Jupyter notebook from a prompt .....	96
6.10 Further exploring llama-2 .....	104
6.11 Generating and working with code.....	108
7. Summary .....	122
Appendix A. Troubleshooting .....	124
Appendix B. Generative AI and Foundation Model concepts.....	125
Appendix C. Revision History.....	128

## 1. Introducing watsonx.ai

Watsonx.ai is a core component of watsonx, IBM's enterprise-ready AI and data platform designed to multiply the impact of AI across an enterprise's business.

The watsonx platform has three powerful components: watsonx.ai for new foundation models, generative AI and Machine Learning (traditional AI); and the watsonx.data fit-for-purpose data store that provides the flexibility of a data lake with the performance of a data warehouse; and the watsonx.governance toolkit, that enables AI workflows to be built with responsibility, transparency, and explainability.

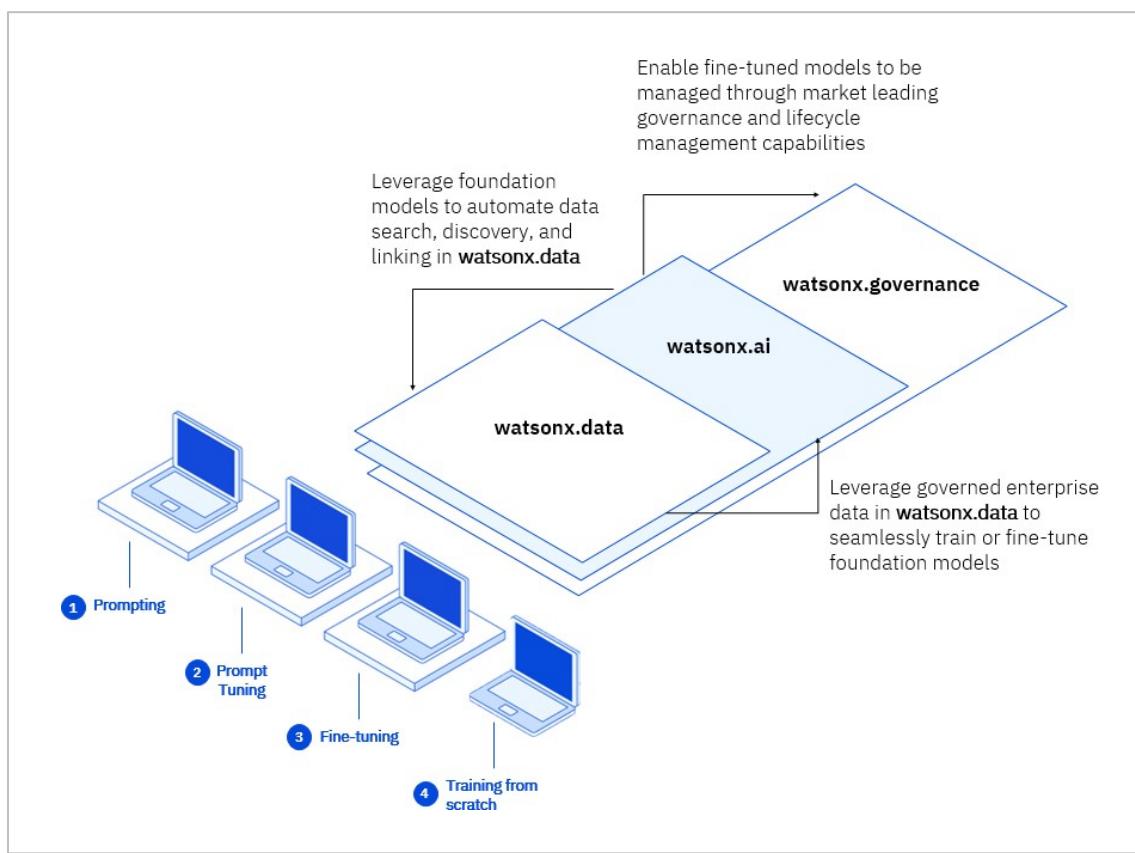


Figure 1: IBM watsonx platform

The watsonx.ai component (the focus of this lab) makes it possible for enterprises to train, validate, tune, and deploy AI models – both traditional AI and generative AI. With watsonx.ai, enterprises can leverage their existing traditional AI investment as well as exploit the innovations and potential of generative AI builds on foundation models to bring advanced automation and AI-infused applications to reduce cost, improve efficiency, scale, and accelerate the impact of AI across their organizations.

## 2. About this Lab

This IBM watsonx.ai hands-on lab introduces you to some of the core components and capabilities of IBM watsonx.ai. By completing this lab, you will gain the experience needed to demo these capabilities to clients.

Specifically, you will get hands-on experience in the following areas:

- The watsonx.ai web-based user interface (UI) which includes the Prompt Lab, the Structured and Freeform interface, model parameter configuration panels, and model information panels.
- Strengths and weaknesses of different models
- Meaning of different configuration parameters and how they influence a model in generating output
- Zero-shot and one-shot prompting (and by extension few-shot prompting)
- Generating list and JSON files with foundation models
- Code generation and translation
- Saving prompts as well as prompt sessions (to keep the history of all your changes)
- Restoring the prompt to an earlier state via prompt history
- Saving prompts to a Jupyter notebook and working with the Jupyter notebook

### 2.1 Disclaimer

Watsonx.ai is being developed and released in an agile manner. In addition to adding new capabilities, the web interface is likely to change over time. Therefore, the screenshots used in this lab may not always look exactly like what you see. You can expect to encounter some of the following:

- Additional foundation models in the library list.
- Changes in the user interface (location of buttons, text for various fields)
- Additional tabs/buttons (especially when the Tuning Studio is

rolled out) These should not affect how the labs work, but have

patience and explore.

There are three changes, however, that can affect the results

- Foundation models can be very sensitive to input. If you enter slightly different text than what the exercise is using (even if it is just one single word), the outcome can be different.
- There are ongoing tuning of the models. If the models themselves are updated, then some of the results may vary.
- The sample prompts may change (more can be added, or updated with new text). Since prompt examples are used heavily in this lab, this can affect what you might see. To mitigate that, the prompt example text used will be provided in a gray text box. If you select a prompt example and the text is different, simply copy the given text into various input fields.

Please raise questions on the [`#data-ai-demo-feedback`](#) Slack channel (IBMer only). IBM partners can request help at the [Partner Plus Support](#) website.

### 3. Getting Help

Lab guide help: If you require assistance in interpreting any of the steps in this lab, please post your questions to the [`#data-ai-demo-feedback`](#) Slack channel (IBMer only). Business Partners can request help at the [Partner Plus Support](#) website.

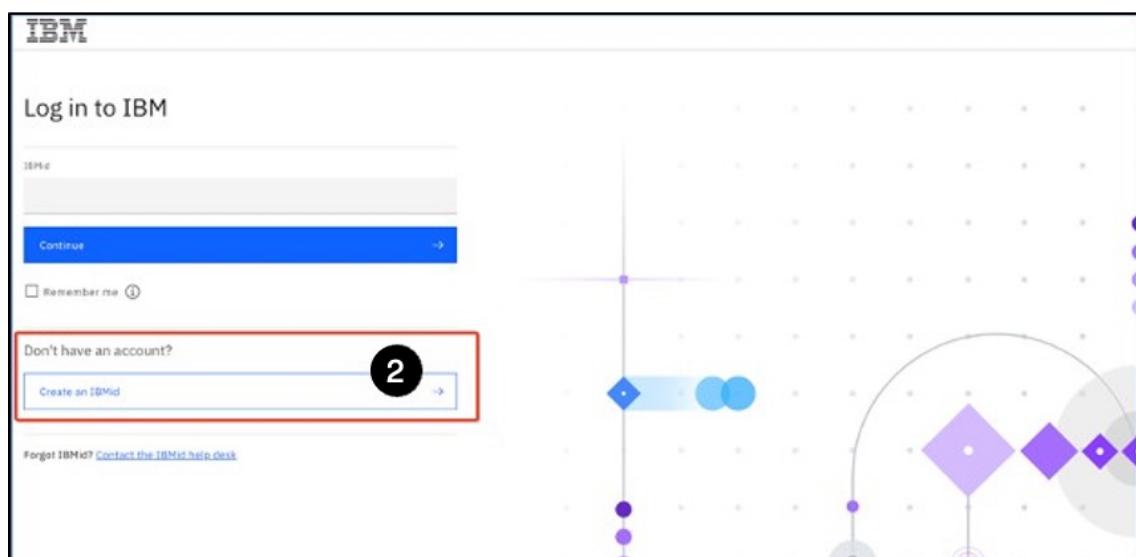
Watsonx.ai: Assistance with the watsonx.ai product itself is available in the [`#watsonx-ai- feedback`](#) (IBMer) and the [`#watsonx-ai-enablement`](#) Slack channels (IBMer only). Additionally, please refer to the [watsonx.ai documentation](#) as needed.

Additional troubleshooting: See [Appendix A. Troubleshooting](#) at the end of this lab guide for guidance on commonly encountered issues.

## 4. Prerequisites & Getting Started

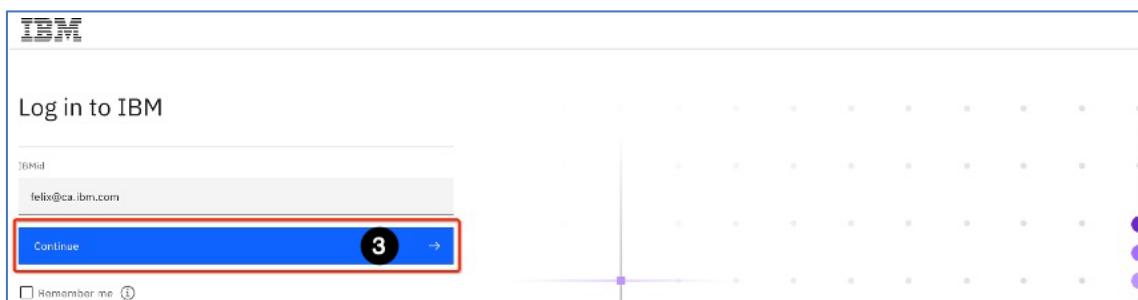
You can use IBM TechZone to perform the exercises in this document. The detailed setup instructions are available in this document:

1. Go to the [IBM TechZone](#) website.
2. If you already have an IBMid, skip to Step 3. If you do not have one, click Create an IBMid.

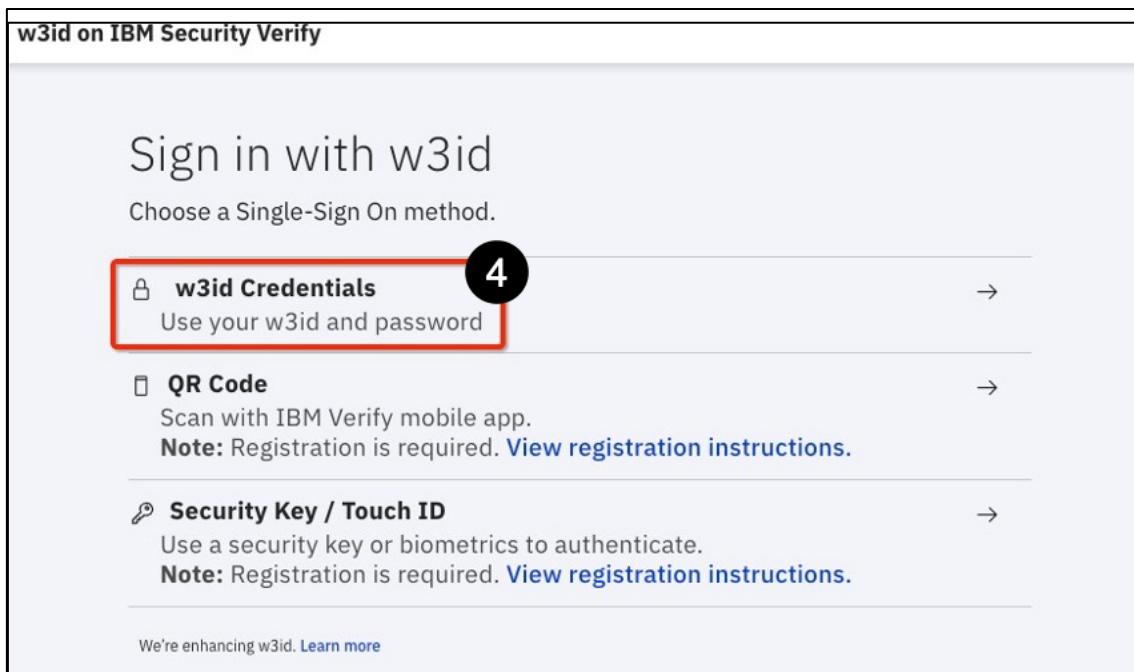


Simply follow the instructions from the subsequent panels to create your IBMid.

3. Provide your IBMid and Click on Continue.



4. Select your Single-Sign-On method. The w3id Credentials option is selected in this example:



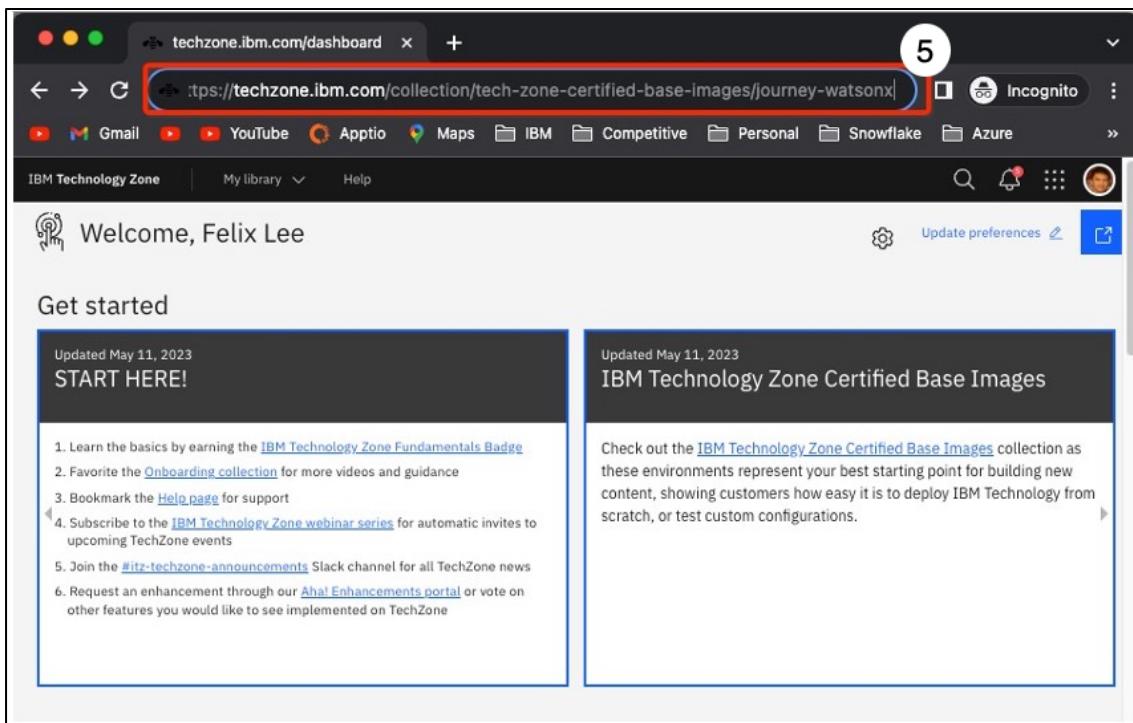
5. Provide the necessary information (w3id and password, QR code, etc.) and sign in.

If this is the very first time you are logging onto the TechZone, you will be brought to the TechZone dashboard. Copy and paste the following HTTP address in the browser's search field (do not use the magnifying glass icon) and press Enter:

<https://techzone.ibm.com/collection/tech-zone-certified-base-images/journey-watsonx>

to be taken to the TechZone Certified Base Images page.

If you have been to TechZone before, you might be taken directly to the TechZone Certified Base Images page already. In that case, proceed to Step 6.



6. Your browser will open to the TechZone Certified Base Images page. On the left-hand side, the watsonx section should be highlighted. If not, select it.

**TechZone Certified Base Images**  
watsonx

★★★★★ (25) Rate this resource

Developer Base Image - docker-based image best suited for education and test activities.

Stand Alone Base Image - OpenShift-based image best suited for education and demo activities.

SaaS Offerings - Available for Customer Demo and Proof-of-Technology purposes only.

Jul 6, 2023 watsonx.data Developer Base Image  This image contains the required software and prerequisites that are needed to run the IBM watsonx.data software.	Sep 21, 2023 Ibmcloud 2: us-east, us-east, eu-de, eu-gb, eu-gb, jp-tok  watsonx.data Standalone - OCP on VMWare  Self-Managed watsonx.data Standalone UPI (OpenShift cluster (VMware on IBM Cloud) with ODF (OCS) support.  watsonx.data: <ul style="list-style-type: none"><li>watsonx.data operator</li><li>GA Version 1.0.0</li><li>Deployed via OpenShift GitOps and OpenShift Pipelines</li></ul> Infrastructure details: <ul style="list-style-type: none"><li>UPI - User Provided Infrastructure</li><li>Cloud platform independent (Bare-metal)</li></ul>	Aug 11, 2023 watsonx.data standalone login and usage  This video will show you how to find your watsonx.data credentials, url login information and how to open the lakehouse portal.
---	---	--

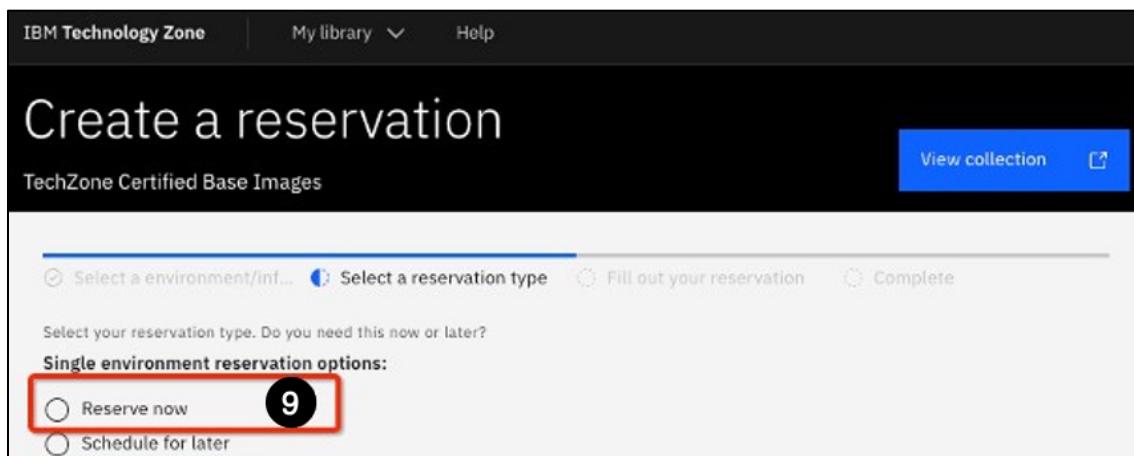
7. Scroll down until you find the watsonx.ai/.governance SaaS section (highlighted in red below).

The screenshot shows the IBM Technology Zone interface. On the left, there's a sidebar with categories like Overview, Pre-Installed Software, and watsonx. The main content area displays several cards representing different offerings. One card for 'watsonx.data Developer Base Image' is visible, along with three cards for 'watsonx.data Standalone - OCP on VMWare'. A fourth card, 'watsonx.ai/.governance SaaS', is highlighted with a red box and has the number '7' in a circle above it. This card includes details about its availability (Ibmcloud 2: eu-de, jp-tok, us-south, us-south, eu-de, jp-tok, us-south, eu-de, jp-tok), a video link for login information, and a note about accepting an invitation.

8. Click on Reserve in the watsonx.ai/.governance SaaS tile.



9. You will now be on the reservation site for a watsonx.ai environment. Select Reserve now. If you are not ready to start, you can select Schedule for later radio button – the steps are the same.



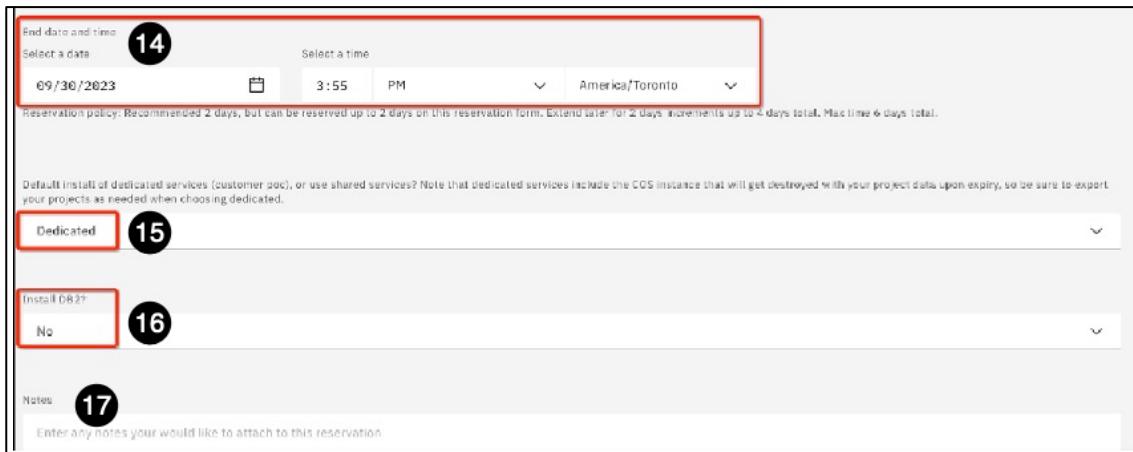
Now perform the following:

10. Change the Name of the reservation to <your id> watsonx.ai L3 SaaS
11. Select Practice/Self-Education
12. For the Purpose description, enter Completing L3 Lab
13. Select your preferred Geography – choose us-south or Frankfurt, do NOT use Tokyo.

The screenshot shows the 'Fill out your reservation' step. It includes fields for 'Name' (with value 'felix watsonx.ai SaaS' highlighted with a red box and number 10), 'Purpose' (with 'Customer Demo' and 'Proof-of-Technology' options), 'Test' (with 'Practice / Self-Education' checked and highlighted with a red box and number 11), 'Sales Opportunity number' (with placeholder 'Enter an opportunity number'), 'Purpose description' (with value 'Completing L3 Lab' highlighted with a red box and number 12), and 'Preferred Geography' (with value 'AMERICAS - us-south region - dal10 datacenter' highlighted with a red box and number 13). A note at the bottom states: 'Please ensure to select the correct purpose as this can NOT be updated or changed after this reservation has been created. Review the Reservation Duration Policy to understand default durations allowed for specific infrastructures based on purpose.'

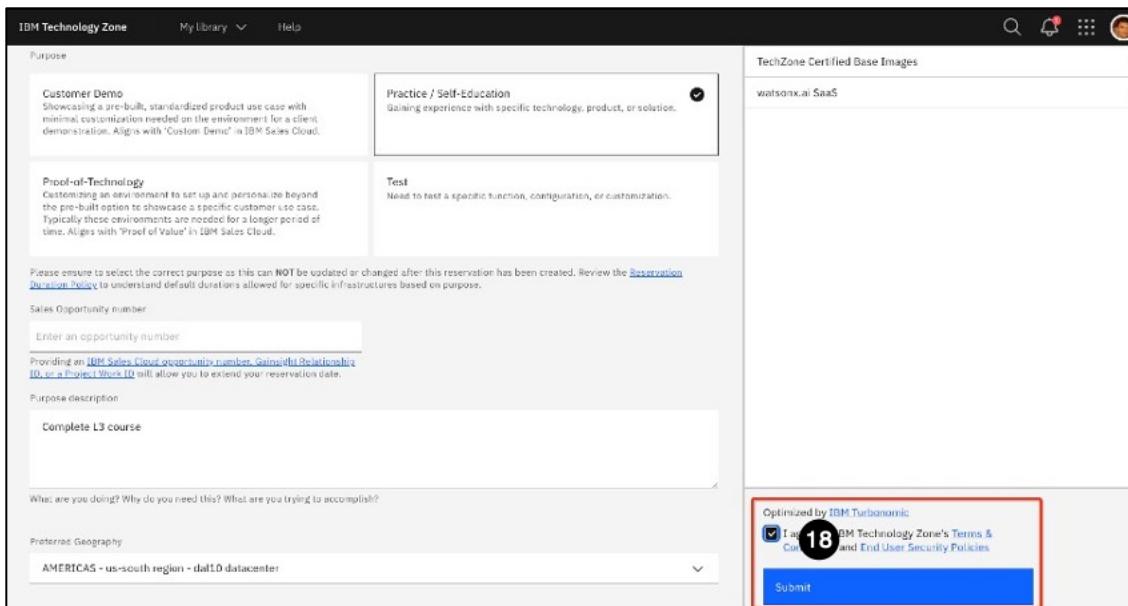
Continue to the bottom part of the reservation page:

14. Specify the end date and time (1). The maximum you can do is 2 days from the current time, but it can be extended incrementally up to 4 more days.
15. Select Dedicated (should be the default) service (2)
16. Select No (should be the default) for installing Db2 (3)
17. You can optionally provide some additional notes if you choose



The screenshot shows the 'End date and time' section with a red box around it, containing a date picker set to '09/30/2023', a time picker set to '3:55 PM', and a dropdown for 'America/Toronto'. Step 14 is circled in black at the top left of this box. Below this, there's a note about reservation policy. Step 15 is circled in black around the 'Dedicated' service option. Step 16 is circled in black around the 'No' selection for 'Install DB2?'. Step 17 is circled in black around the 'Notes' section with a placeholder 'Enter any notes you would like to attach to this reservation'.

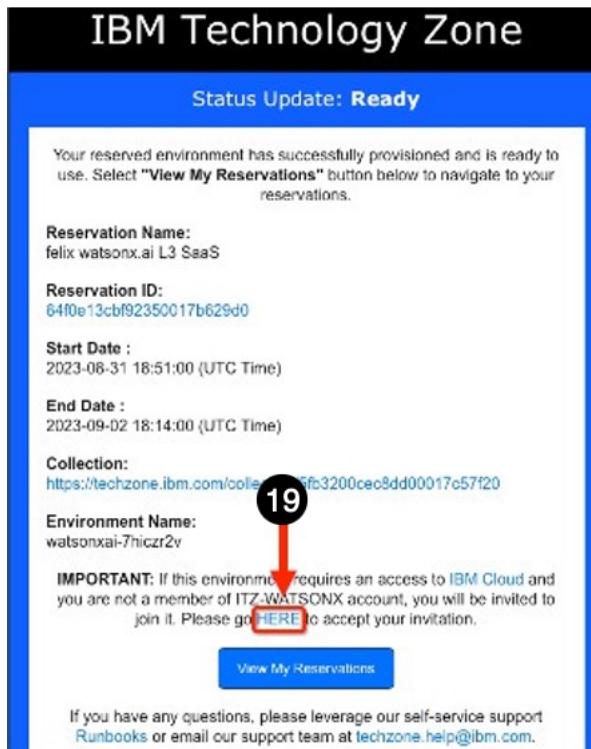
18. On the right-hand side at the bottom, accept the Terms and Conditions, then click Submit.



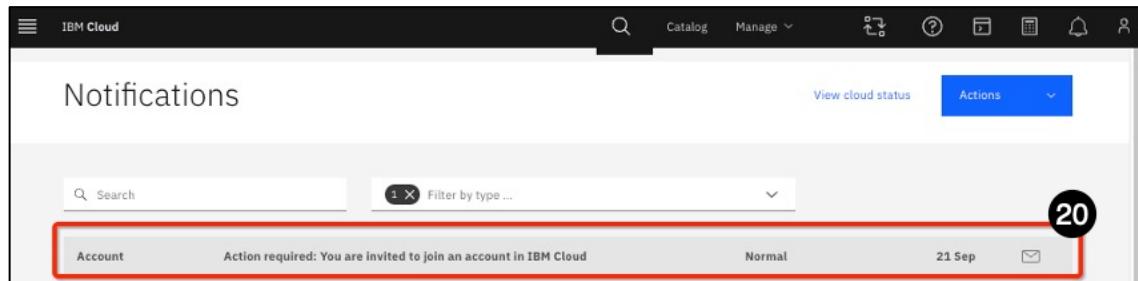
The screenshot shows the 'Purpose' section with three options: 'Customer Demo', 'Practice / Self-Education' (which is selected and has a checkmark), and 'Test'. It also shows fields for 'Sales Opportunity number' (with a note about providing an opportunity ID or Project ID) and 'Purpose description' (with a note about what the user is doing). At the bottom right, there's a red box around the 'Optimized by IBM Turbonomic' section, which includes a checked checkbox for accepting the 'IBM Technology Zone's Terms & Conditions' and a 'Submit' button. Step 18 is circled in black around the 'Accept' checkbox.

19. You will get an email sent to your IBMid email address letting you know that

your reservation is being provisioned (this should happen within 10-15 minutes). Once the watsonx.ai provisioning is completed, you will receive a second email telling you that it is ready for use. This email looks like the example below. You will need to click on the **HERE** link (as highlighted below) to go and accept the invitation.



20. Clicking on **HERE** brings you to a login page where you need to provide your IBMid (and may have to log in with your credentials if you have not already done so). Then you will see the Action required item. Click on the item.



21. The following page opens up. Click Join now.

Action required: You are invited to join an account in IBM Cloud

**Update time:** 21 Sep 2023, 4:26 PM local time

**Severity:** Normal

Hi felix@ca.ibm.com,

Data Steward invited you to join the following IBM Cloud account: Techzone Outcomes  
(Account ID: feb0a088323a45db90b8dd694b314c53).

**Join now.**

21

The link to join the account expires in 30 days.

Thank you,  
IBM Cloud

In case of any issues, please contact [IBM Cloud support](#).

22. You will now be sent back to the IBM Cloud login page. Use your IBMid to log in. You are asked to join the TechZone account. Accept the Terms and Conditions, then click Join Account.

The screenshot shows the 'Join Account' form on the IBM Cloud website. The 'Account notice' section is highlighted with a red box around the checkbox for accepting terms and conditions. A large blue checkmark graphic is visible on the right side of the page.

Join IBM Cloud  
You're almost there. Complete your IBM Cloud account details so that you can get started.

**Account information**  
felix@ca.ibm.com

**Personal information**  
Felix Lee  
United States

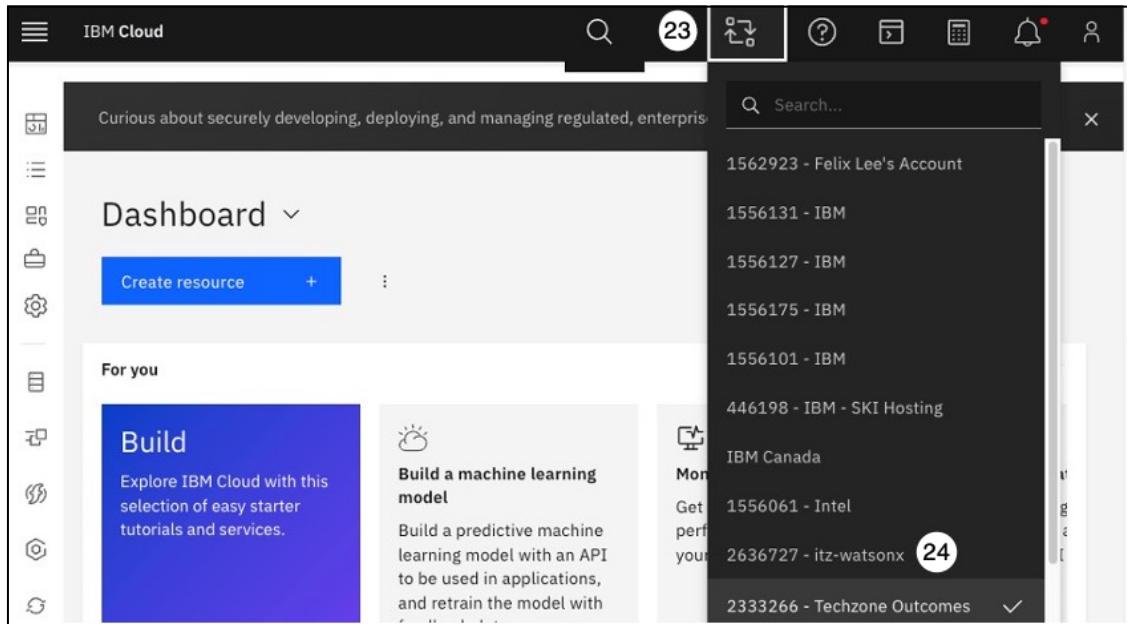
**Account notice**  
 I accept the product Terms and Conditions of this registration form.

**Join Account** 22

23. When you log in, ensure you are using the right account. Assuming that you have accepted the invitation, you should have an itz-watsonx account. Go to the

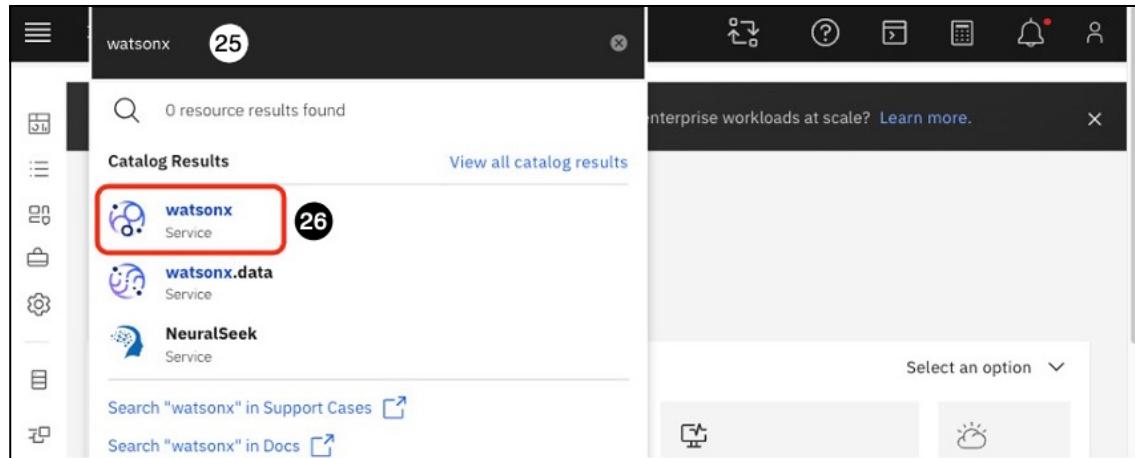
menu bar and click on the  icon to slide open the list of accounts available to you.

24. Ensure that you select an item that looks like <your id number> - itz-watsonx.

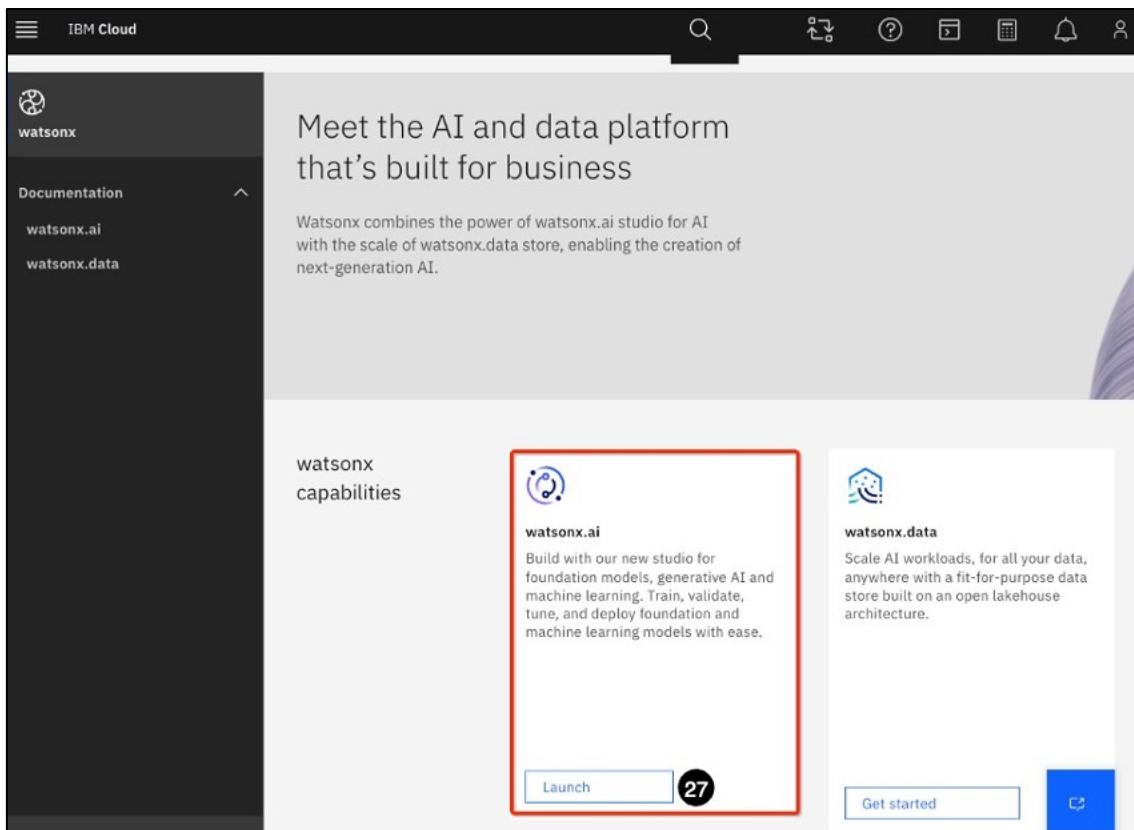


25. In the search field, enter watsonx.

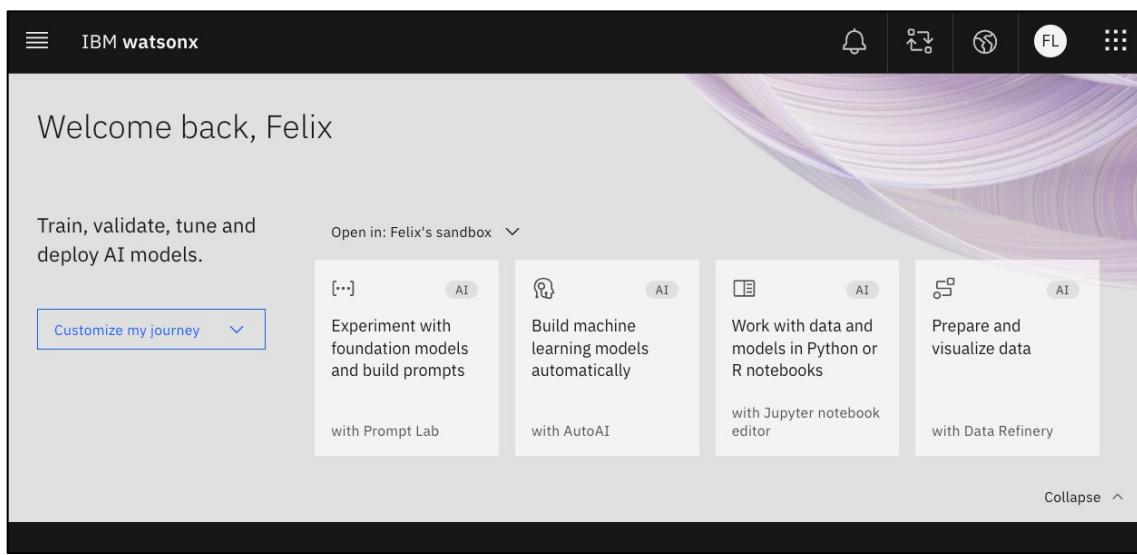
26. The search will return a list similar to the following. Click watsonx.



27. You will now see the AI and data platform page. Click Launch from the watsonx.ai tile.



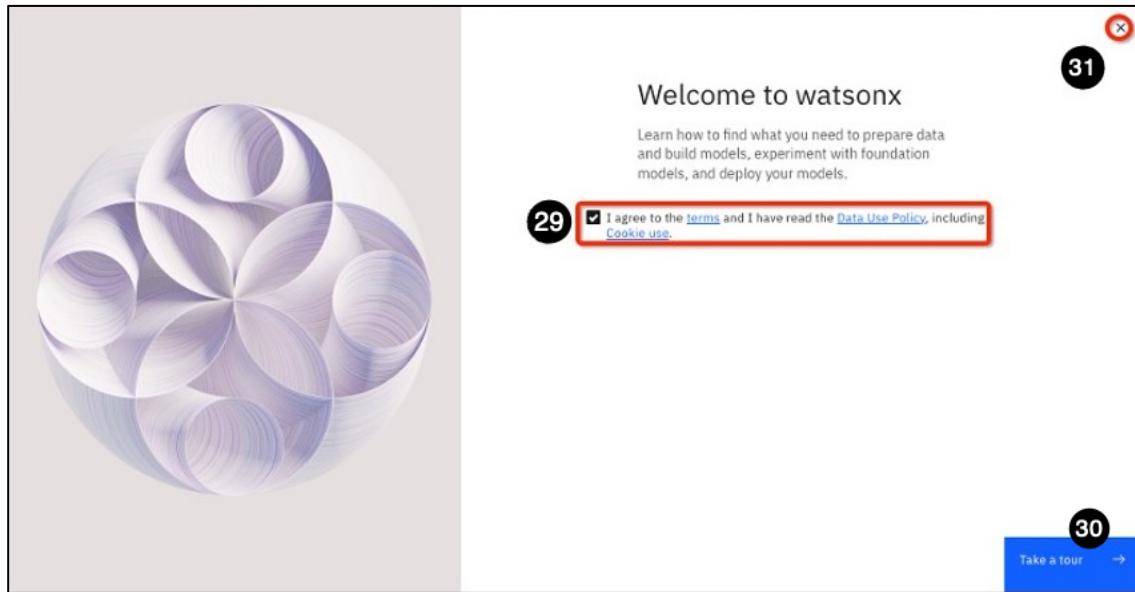
28. You are now in the watsonx.ai console and ready to begin this lab.



29. You might see the following page. Select the checkbox to agree to the terms.

30. You can click on the **Take a tour →** button on the bottom-right if you are interested in taking a tour of watsonx.ai.

31. For this lab, simply click on the “x” on the upper right to close this window.



You are now ready to explore the watsonx.ai console.

## 5. Watsonx.ai console

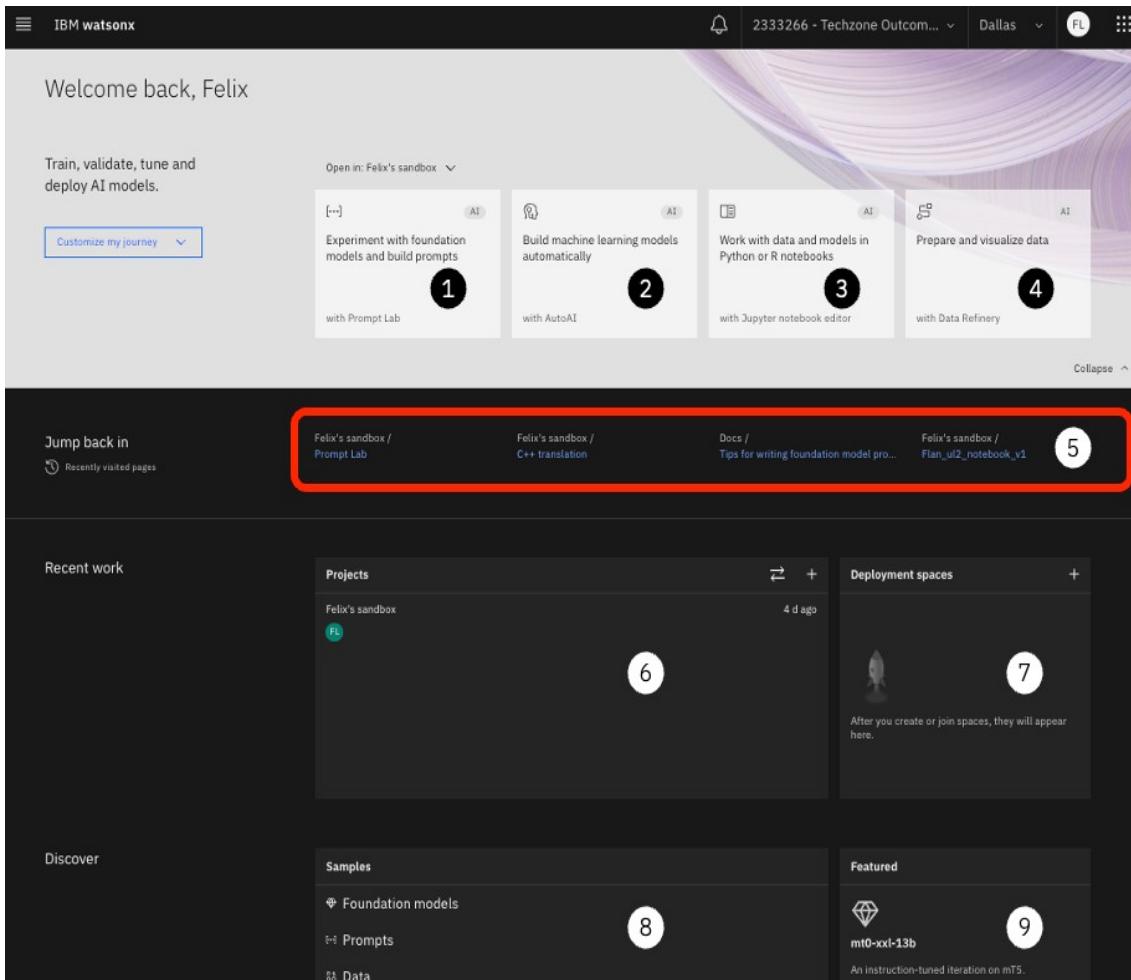
You will see the following console as you log onto watsonx.ai.

Here are the various regions of the console:

1. Takes you to the Prompt Lab console. You can experiment and work with different models, test your prompts, and work on zero-shot, one-shot, or few-shot prompting. You can also change various parameters to control how conservative or creative the generative AI model will be. This is the focus of this lab.
2. Lets you create AutoAI jobs to automatically build machine learning (ML) models.
3. Lets you work with Python or R notebooks. You can create new ones or use existing notebooks you have already created.
4. Lets you load data and then prepare it (using Data Refinery) for AI consumption.
5. Quick links to recently visited pages.
6. Shows a list of projects. For the watsonx.ai free tier, you will see a default project here called <username> sandbox.
7. Deployment space – this is where you can place assets in one place to create, run, and manage deployments.
8. Collection of samples – This is an important area if you are new to watsonx.ai and generative AI. Here you will find examples<sup>1</sup> and useful information on:
  - a. Foundation models that are available to use in watsonx.ai.
  - b. Prompts
  - c. Data
  - d. Projects
  - e. Notebooks
9. Model highlights – watsonx.ai will highlight various foundation models. It also provides details of the intended usage of the model. For open models from Hugging Face, you can also link to additional information from Hugging Face.

---

<sup>1</sup> The first 2 sets of examples are generative AI-related, the other sections contain examples of traditional AI. These are available to provide a better progression into generative AI.



In this lab, you will focus on the prompt lab (item 1 above). The Prompt Lab provides rich capabilities for working with prompts and different foundation models. This lab assumes that you are familiar with various generative AI and foundation model concepts covered in the following Your Learning assets:

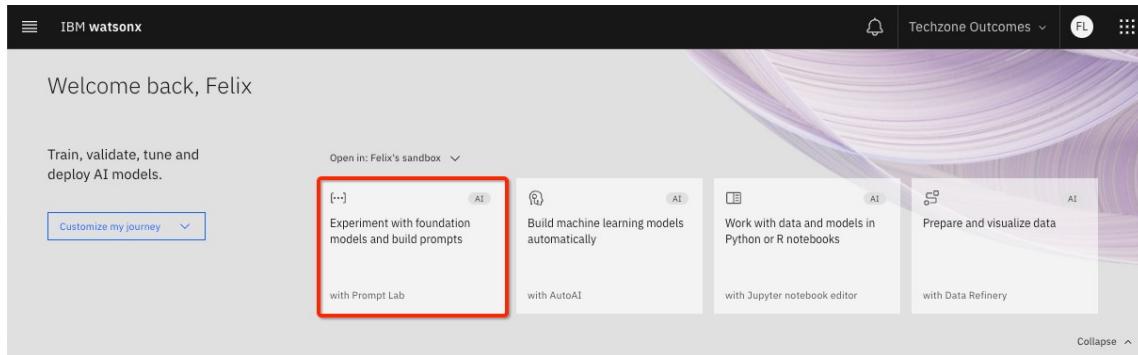
- [AI Fundamentals Level 1](#)
- [AI for Business Level 1](#)

Topics that will be covered include:

- Basic navigation of the Prompt Lab user interface
- Exploring different foundation models and their strengths/weaknesses
- Exploring simple zero-shot prompting and one-shot prompting
- Perform prompt engineering – changing the prompt input, or various parameters
- Create text and formatted output
- Various ways of saving your prompt information
- Creating a Jupyter notebook and running it

## 6. Watsonx.ai Prompt Lab

From the watsonx.ai console, click on the first tile.



This brings you to the Prompt Lab.

### 6.1 Basic Navigation

The Prompt Lab provides many capabilities. This lab only examines a subset.

The window that opens is the default view of the Prompt Lab. There are 3 icons on the left. You can hover over each one to get some information about each:

- ⌚ Provides a list of prompt samples – most useful for beginners
- 💬 List of saved prompts
- 🕒 History of prompts used in the current session

The other important areas of the Prompt Lab (looking at a subset for now) include:

1. Watsonx.ai provides AI guardrails. By default, it is off. You turn this on to prevent potential harmful input and output text (such as hate, abusive, or prejudiced wordings). You might have to start working with watsonx.ai before you can turn this on.
2. You might have to start working with watsonx.ai before you can turn this on.  
The ability to toggle between the Structured prompt or Freeform prompt editors.
  - a. Structured prompt is the default and provides guidelines for the prompt. This is recommended for beginners.

- b. Experienced users may opt for Freeform prompting.
3. You can change to different foundation models by clicking on the drop-down menu.
4. Allows users to provide an instruction which will become part of your prompt.  
While it is optional, you should be careful what you place here because it consumes tokens.

Keep in mind that prompting is not the same as asking a question. A foundation model is not “answering” but rather generating output based on the input. Anything that is included can be used by the model.

5. You have the option to provide a sample input here.
6. This is where you provide the sample output (corresponding to the input from item 4).

Foundation models can sometimes be thought of as probability machines – they generate output based on what might be the best extension of the given input (prompt). However, unless a user is an expert in writing prompts, a foundation model may not understand the intent of the prompt or the type/structure desired for the output.

A user can “teach” a foundation model by providing a sample input and output (referred to as a “shot”). These instruct the model on how to best respond to a query/prompt. Types of shots include:

- Zero-shot prompting – no input/output sample provided
- One-shot prompting – a single input/output sample provided
- Few-shot prompting – multiple input/output samples provided

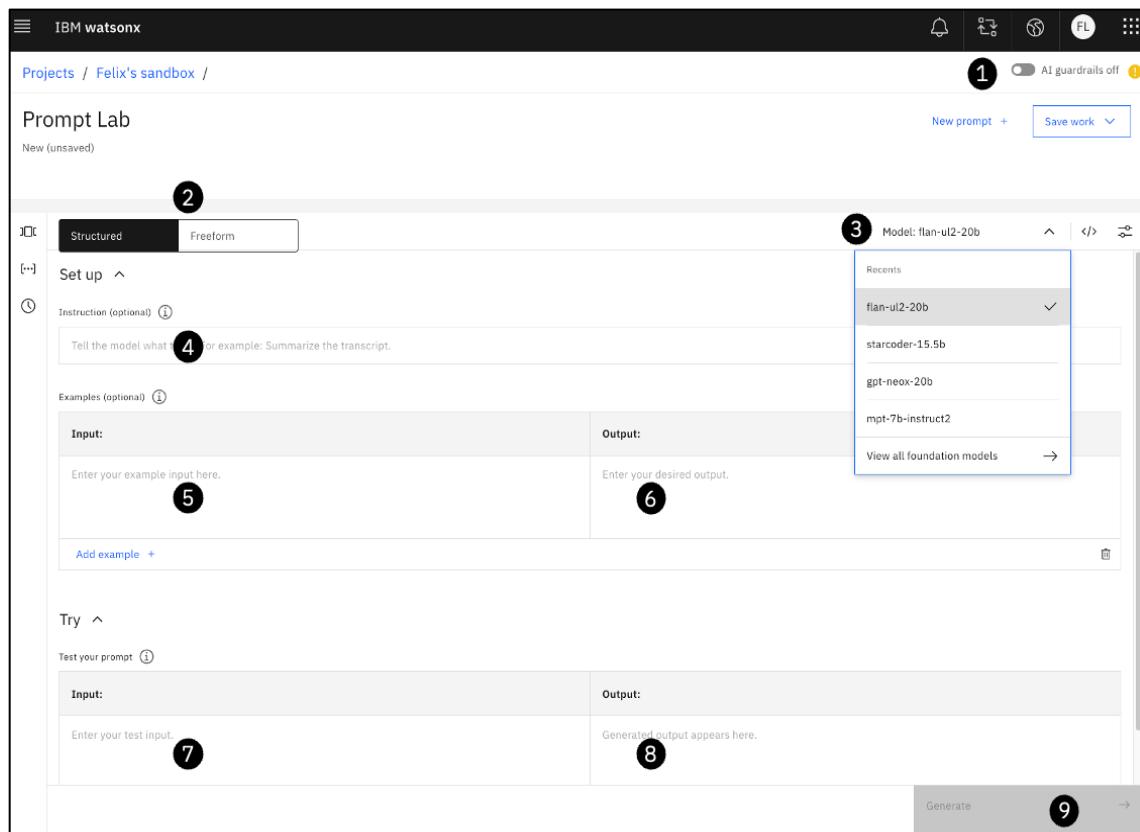
One-shot or few-shot prompting makes it easier to teach a model how to optimally respond. Zero-shot can still be effective when one has good skills in prompt engineering.

Notes:

- Prompts are tokenized before being passed into a model, and typically foundation model usage costs are calculated based on the number of tokens. Using few-shot prompting frequently can become expensive.
- As a related concept, the larger the foundation model the more resources (cores and memory) and costs are required to operate them.

7. The Try section is where you enter your prompt/query into the foundation model.
8. This is where the generated output from a foundation model will be displayed.
9. Click the Generate button for the model to generate the output.

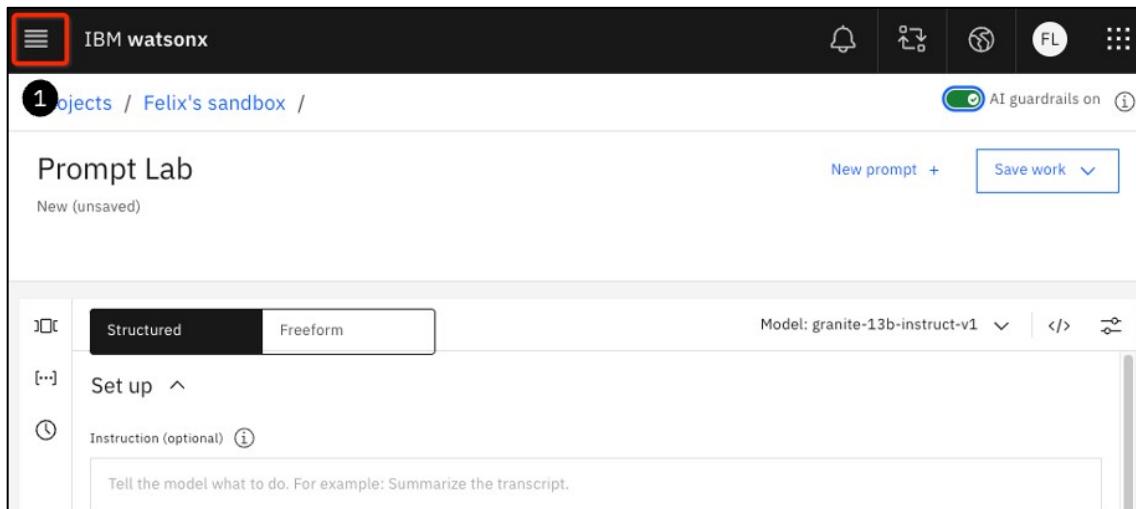
Other controls available will be discussed later in the lab.



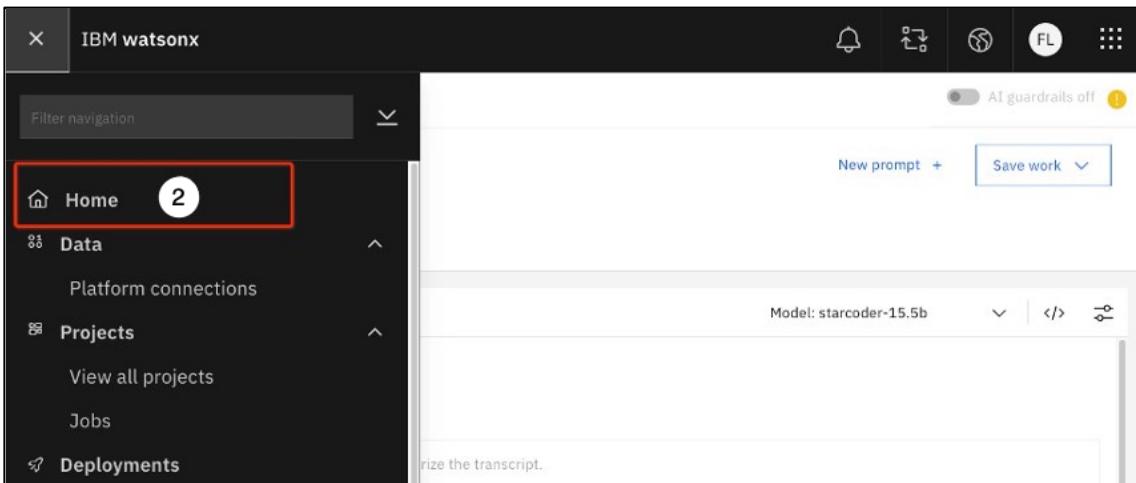
### 6.1.1 Open up a new Prompt Lab session

Many of the exercises in this lab begin with a step: Open a new Prompt Lab session. Here are the steps to do so.

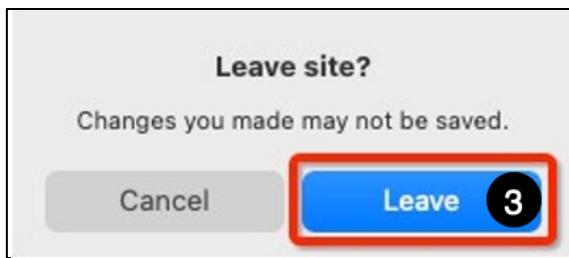
1. Click on the  icon in the upper left corner of the console.



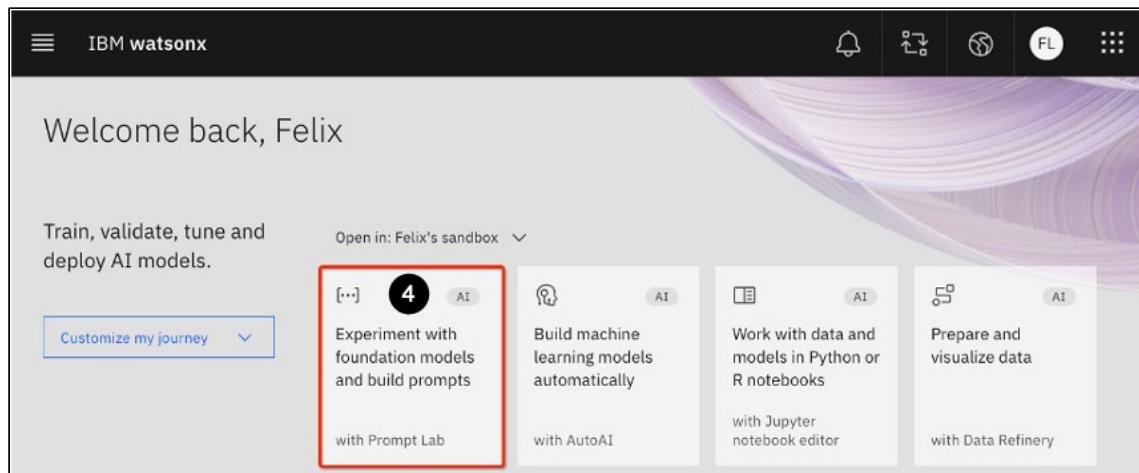
6. On the slide-out panel, click Home.



7. Depending on what you have been doing, you might see this message. If so, click Leave.



8. You are back to the watsonx.ai welcome screen. Select the Experiment with foundation models and build prompts tile (1).



This takes you back to the Prompt Lab.

Note: you can certainly explore the Prompt Lab on your own; however, since foundation models do “remember”, exploring and testing outside of the scope of the lab might impact your prompt output. You can always do a reset by opening up a new Prompt Lab session.

## 6.2 Exploring different foundation models and their characteristics

There are 6 open-source foundation models available in watsonx.ai.

Model	Num of parameters	Trained by	Usage
<b>flan-ul2-20b</b> Encoder-decoder	20 billion	Google	Generation, Extraction Summarization, Classification Question Answering, RAG <sup>2</sup>
<b>Starcoder-15.5b</b> Decoder-only	15.5 billion	BigCode	Code generation, Code conversion
<b>mt0-xxl-13b</b> Encoder-decoder	13 billion	BigScience	Generation, Summarization, Classification, Question Answering
<b>gpt-neox-20b</b> Decoder-only	20 billion	EleutherAI	Generation, Summarization, Classification
<b>flan-t5-xxl-11b</b> Encoder-decoder	11 billion	Google	Generation, Extraction Summarization, Classification Question Answering, RAG
<b>mpt-7b-instruct2</b> Decoder-only	7 billion	IBM	Generation, Extraction Summarization, Classification
<b>Llama-2-70b-chat</b> Decoder-only	70 billion	Meta	Generation, Extraction Summarization, Classification Question Answering, RAG, Code generation, Code conversion

At the closing of the current version of this lab guide, there are also 2 IBM models available: the granite-13b-chat-v1 model and the granite-13b-instruct-v1 model. These will be covered in more detail in a coming module of this L3 course.

There is also an IBM-trained Slate model which is available from the Natural Language Library from Watson Studio. It will not be covered in this lab as the focus is on the watsonx.ai console.

The encoder/decoder designation signifies the foundation model's architecture. See [Appendix](#)

---

<sup>2</sup> Retrieval-Augmented Generation (RAG)

[B. Generative AI and Foundation Model concepts](#) for more information.

---

You will now work with various models and prompts.

1. Click the  icon to slide open a list of sample prompts on the left (1).

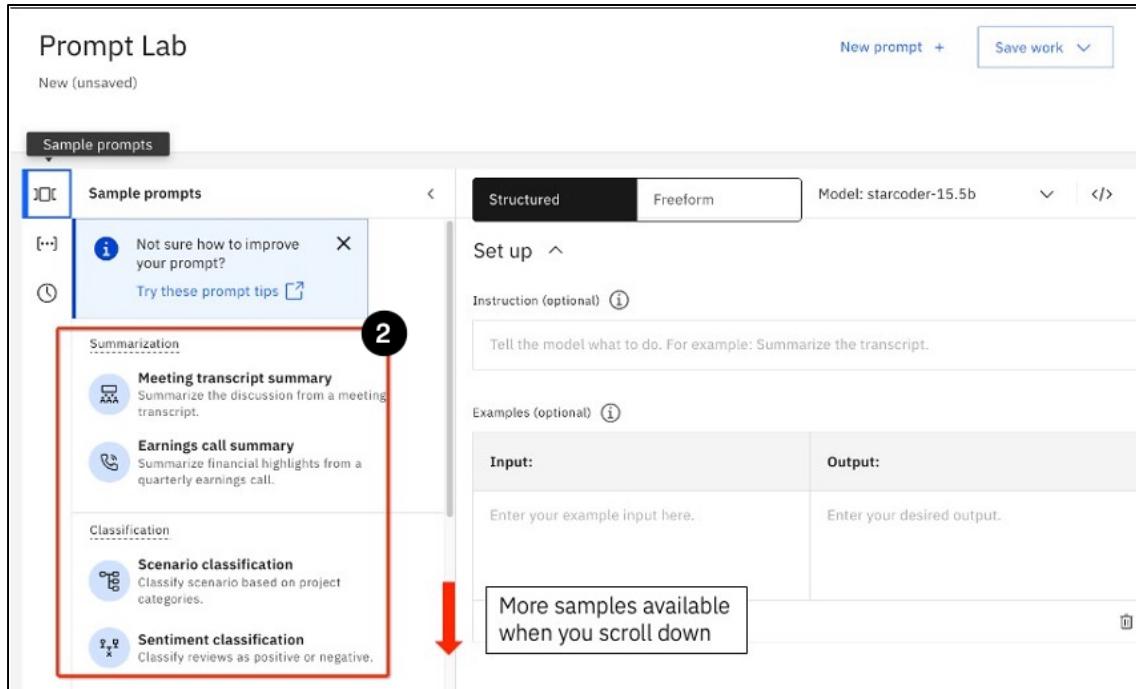


Watsonx.ai provides sample prompts grouped into categories like:

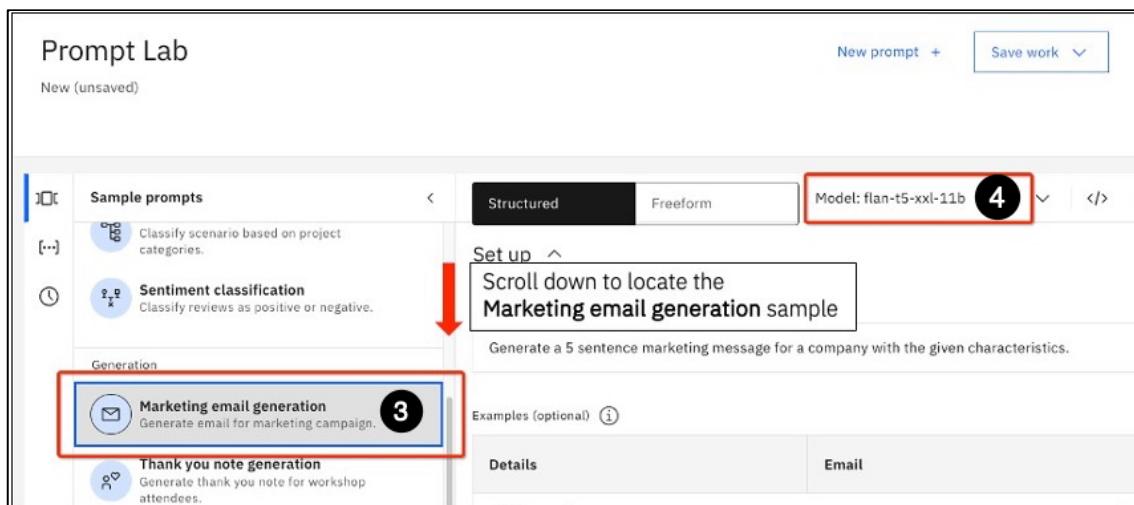
- Summarization
- Classification
- Generation
- Extraction
- Question Answering
- Code

These are the 6 main use cases for generative AI (there is no sample prompt yet for the Retrieval Augmented Generation use case). For this first exercise, you will look at a Marketing email generation example in the Generation section.

2. Select the Marketing email generation from the list of examples on the left. You should see the following sample prompts:



3. Once you select a sample prompt, the selected sample is highlighted in a blue rectangle. Scroll down to find the Marketing email generation example and click it.
4. Notice how the model flan-t5-xxl-11b was automatically selected. Watsonx.ai selects the model that is most likely to provide the best performance. However, this is not a guarantee, and in this part of the lab, you will explore different models and observe their completion.



This prompt asks the foundation model to generate a 5-sentences marketing message based on the following characteristics (click on the Details field in the

Try section to expand the box and see the full text of this example).

Details
<p>Characteristics:</p> <p>Company - Golden Bank</p> <p>Offer includes - no fees, 2% interest rate, no minimum balance</p> <p>Tone - informative</p> <p>Response requested - click the link</p> <p>End date - July 15</p>

If you cannot find this prompt example, or if the contents have changed, you can enter:

- For Instruction

Generate a 5 sentence marketing message for a company with the given characteristics.

- For Details under the Try section

Characteristics:

Company - Golden Bank

Offer includes - no fees, 2% interest rate, no minimum balance

Tone - informative

Response requested - click the link

End date - July 15

5. Click Generate to see the email this foundation model will generate based on the prompt and its details. The following output is generated:

Email
<p>We've got great news for you! Golden Bank is now offering Online Banking -- a better tool to manage your money. It's no-fee, and it offers 2% interest on all your accounts. You can now monitor all of your Golden Bank accounts from home. If you're interested, click on the link below to sign up. The promotion ends on July 15.</p>

This seems to be a reasonable draft as it hits all the characteristics specified in the

details. There is one line “You can now monitor all of your Golden Bank accounts from home” that is not provided from your input. This is not an entirely unreasonable sentence as this is quite common in today’s banking sector. Moreover, Golden Bank is the name of an actual financial institution. The model may have had some information regarding it.

This is a reasonable output – but perhaps not yet ideal.

Note: In this exercise, you did not pass in any sample input/output. This is an example of zero-shot prompting.

6. All the text in the Details section is part of the prompt. Look at the bottom of the console (to the left of the Generate button you will see the following text:

```
Stop reason: End of sequence token encountered
Tokens: 63 input + 81 generated = 144 out of 4096 | Seed: 111
Time: 3.6 seconds
```

Note the following:

- You might see a different execution time than 3.6 seconds. These are not dedicated environments, so performance will likely vary.
  - In this exercise, the input (including the instruction) consumed 63 tokens while the output consumed another 81 generated tokens.
  - A token is an abstract entity that the model uses to represent input and output. In this example, the instructions and details add up to fewer than 40 words. This means that **the number of words does not map 1-1 to the number of tokens**. The ratio of words: tokens varies anywhere from 1:1.3 to roughly 1:3<sup>3</sup> (depends on length of words, keep in mind that punctuations etc. also consume tokens). It is also model-dependent.
  - The information says that you have consumed 144 out of 4096 tokens. The limiting value of 4096 is dependent on the model. For example, the limit for the mpt-7b-instruct2 model is 2048 tokens. For watsonx.ai GA the input and output tokens total is limited. Among other things, you would not use it to generate a 10-page essay. These model limits will be raised in the future.
7. There is a monthly limit on the total number of tokens available to you, depending on your subscription.

---

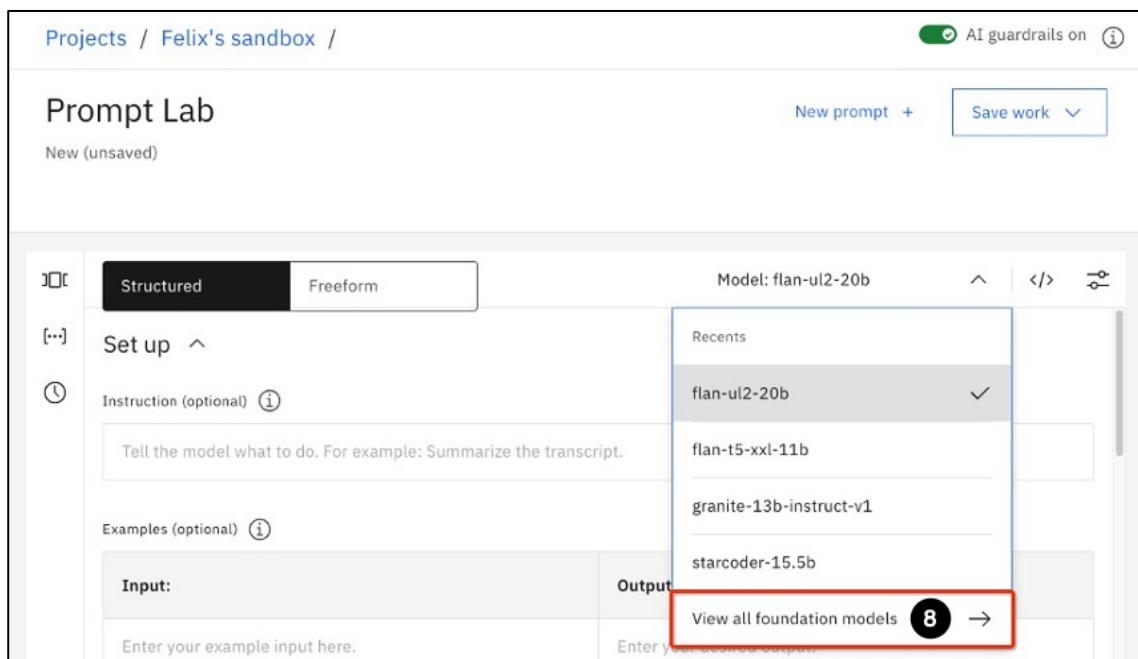
<sup>3</sup> See <https://ec.yourlearning.ibm.com/w3/join/event/10388387> (IBMer).

Now you will try a different foundation model. Click the model drop-down box on the right.



If this is your first time using the watsonx.ai Prompt Lab, you will see only the flan-t5- xxl-11b model listed. If you have used other models, you will see them in the drop-down.

8. Regardless of what is shown in the drop-down, click the View all foundation models option.



This will bring up a list of models as shown below:

Select a foundation model

Select a model that best fits your needs. All models support English text. Check the model information for other supported languages.

<b>flan-ul2-20b</b> ⓘ flan-ul2 is an encoder decoder model based on the T5 architecture and instruction-tuned using the Fine-tuned Language Net.  Provider: Google Source: Hugging Face	<b>starcoder-15.5b</b> The StarCoder models are 15.5B parameter models that can generate code from natural language descriptions  Provider: BigCode Source: Hugging Face	<b>mt0-xxl-13b</b> An instruction-tuned iteration on mT5.  Provider: BigScience Source: Hugging Face	<b>gpt-neox-20b</b> A 20 billion parameter autoregressive language model trained on the Pile.  Provider: EleutherAI Source: Hugging Face
<b>flan-t5-xxl-11b</b> flan-t5-xxl is an 11 billion parameter model based on the Flan-T5 family.  Provider: Google Source: Hugging Face	<b>granite-13b-chat-v1</b> The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative... tasks.  Provider: IBM Source: IBM	<b>granite-13b-instruct-v1</b> The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative... tasks.  Provider: IBM Source: IBM	<b>mpt-7b-instruct2</b> MPT-7B is a decoder-style transformer pretrained from scratch on 1T tokens of English text and code. This model was... trained by IBM.  Provider: Mosaic, tuned by IBM Source: Hugging Face
<b>llama-2-70b-chat</b> Llama-2-70b-chat is an auto-regressive language model that uses an optimized transformer architecture.  Provider: Meta Source: Hugging Face			

- In the previous example, you were using the flan-t5-xxl-11b model. This time, select the flan-ul2-20b model. Note that this is a bigger model with 20 billion parameters versus the flan-t5-xxl-11b model which has 11 billion parameters.

Select a foundation model

Select a model that best fits your needs. All models support English text. Check the model information for other supported languages.

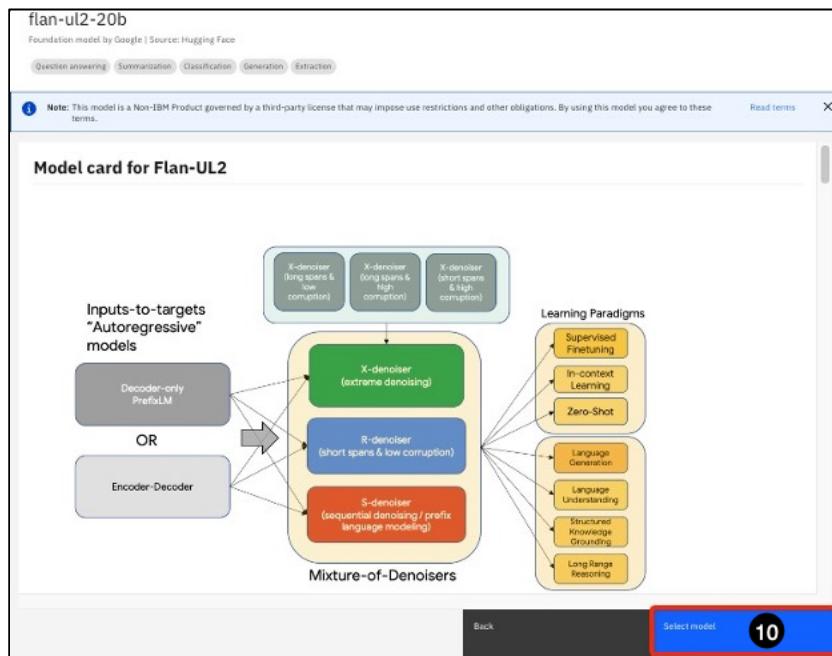
<b>flan-ul2-20b</b> ⓘ flan-ul2 is an encoder decoder model based on the T5 architecture and instruction-tuned using the Fine-tuned Language Net.  Provider: Google Source: Hugging Face	<b>starcoder-15.5b</b> The StarCoder models are 15.5B parameter models that can generate code from natural language descriptions  Provider: BigCode Source: Hugging Face	<b>mt0-xxl-13b</b> An instruction-tuned iteration on mT5.  Provider: BigScience Source: Hugging Face	<b>gpt-neox-20b</b> A 20 billion parameter autoregressive language model trained on the Pile.  Provider: EleutherAI Source: Hugging Face
<b>flan-t5-xxl-11b</b> flan-t5-xxl is an 11 billion parameter model based on the Flan-T5 family.  Provider: Google Source: Hugging Face	<b>granite-13b-chat-v1</b> The Granite model series is a family of IBM-trained, dense decoder-only models, which are... particularly well-suited for generative tasks.  Provider: IBM Source: IBM	<b>granite-13b-instruct-v1</b> The Granite model series is a family of IBM-trained, dense decoder-only models, which are... particularly well-suited for generative tasks.  Provider: IBM Source: IBM	<b>mpt-7b-instruct2</b> MPT-7B is a decoder-style transformer pretrained from scratch on 1T tokens of English... text and code. This model was... trained by IBM.  Provider: Mosaic, tuned by IBM Source: Hugging Face
<b>llama-2-70b-chat</b> Llama-2-70b-chat is an auto-regressive language model that uses an optimized transformer... architecture.  Provider: Meta Source: Hugging Face			

When you select a model from this screen, watsonx.ai provides additional information

about the model through a model card.

Browse the page to learn more about this model. Notice that this is a 3<sup>rd</sup> party model. You can scroll down to find out how it was trained, and how it might have already been tuned. This information may be of more use and interest to experienced data scientists.

10. Click Select model to use this model.



11. Verify you are now using the flan-ul2-20b model, then click Generate. You will see the following:

The screenshot shows a user interface for generating email templates. At the top, there are two tabs: "Structured" (selected) and "Freeform". Below them is a button "Add example +". On the right, a status bar indicates "Model: flan-ul2-20b" with a circled "11" icon. A red arrow points from the text "Ensure that you are using the flan-ul2-20b model" down to the "Model" text.

Try ^

Test your prompt ⓘ

Details	Email
Characteristics: Company - Golden Bank ... Offer includes - no fees, 2% interest rate, no minimum balance	We'd like to tell you more about Golden Bank and our great new checking account. It's free, requires no minimum balance and has no monthly maintenance fees. Currently we're offering a 2% interest rate on all balances. To take advantage of this offer, click the link below or call us at 713-852-1411.

New test +

The completion is as follows:

We'd like to tell you more about Golden Bank and our great new checking account. Its free, requires no minimum balance and has no monthly maintenance fees. Currently we're offering a 2% interest rate on all balances. To take advantage of this offer, click the link below or call us at 713-852-1411.

This is an interesting completion. While most of the email is similar to the output of flan-t5-xxl-11b, this model has shown a common flaw of generative AI: hallucination. Note that the input does not have the phone number 713-852-1411 – the model has completely made up that part of the email!

Here is a simple example where a larger foundation model performs worse than a smaller one.

12. Now repeat Steps 5-7, this time selecting the mpt-7b-instruct-2 model. Then repeat Step 9 to ask the Prompt Lab to generate a new response using this new model. You will see the following completion:

**Email**

Hi there! We're glad you found this offer today from our friends at "Golden bank".  
They wanted us to reach out and let people know about their new account offering  
that has 0 fee's on anything in it (including overdrafts), gets them instant access via  
card or phone app where they can view all of thier transactions through 24/7 live  
banking available anywhere as long-as internet is avaivable & gives customers up to  
3x more money just by paying off an existing loan wth Them before Dec 12 2021 so  
don't miss your first opportunity ever like these when offered.. Take care :)

This model also hallucinated in its completion – coming up with dates like Dec 12, 2021. The interesting part is the year 2021. This is probably due to the model using data up to 2021.

13. Try the prompt with one more model – Llama-2-70b-chat (the largest model in watsonx.ai as of 4Q23). Follow Steps 7-10 to select the model, and then click Generate. You will see the following output:

## Email

Subject Line : No More Hidden Fees! Get Your Money's Worth With Our Account Services Today at 07/\*/\*<http://www.\*bankgolden\*.com> Unlock your financial freedom and start earning on every dollar you save now- Click this Link Now Before Time Runs Out On This Offer End Date Is July \*\*\*\*!! Don;t miss out let us help put more money in ypur pocket today .

This model hallucinated a URL (bankgolden.com), it generated a July time frame (as in 07/\*/\* and July \*\*\*\*) but is missing the all-important detail of July 15.

### Section summary

- In this part of the lab, you became familiar with picking different models.
- You learned how to find out more detailed information about various models.
- Depending on the task, different models provide different results. It is important to experiment with different models.
- Larger models do not necessarily perform better than smaller models. In fact, larger models may be more inclined to be “creative” (if the underlying conditions and parameter settings like temperature and greedy/sampling mode remain the same).
- This is a zero-shot exercise (you provide no input/output example) with default settings.
- One should always test before using the foundation model too quickly.

## 6.3 Prompt Engineering – updating a zero-shot prompt

The Prompt Lab is the place for you to try out prompt engineering – the “art” of modifying your input to produce the best result.

The last section demonstrated that given a list of characteristics, different models will generate different marketing emails with various degrees of hallucination. In this part of the lab, you will modify the prompt and look at how different models perform the completion.

Note: you will be looking into model operations which is at a token level. While everyone sees LLMs generating the next word, they are really generating the next token. For high-level discussions, you can assume a token equals a word, but that is not the case (long words can have multiple tokens, punctuation marks are tokens, and so on). In this lab, we will use these words interchangeably and as synonyms unless otherwise noted and keep references to words and tokens at a high level

1. Open a new Prompt Lab session and select the  icon from the left to bring up the list of sample prompts. Click the Marketing email generation sample from the Generate section. Use the flan-t5-xxl-11b model if not already specified.

The prompt should read as follows:

- For Instruction  
Generate a 5 sentence marketing message for a company with the given characteristics.
- For Details under the Try section

Characteristics:  
Company - Golden Bank  
Offer includes - no fees, 2% interest rate, no minimum balance  
Tone - informative  
Response requested - click the link  
End date - July 15

If you cannot find this prompt example or if the text content has changed, please use the text provided above to ensure you will see the same completion as shown in this part.

Click Generate to get the following output email:

Email
We've got great news for you! Golden Bank is now offering Online Banking -- a better tool to manage your money. It's no-fee, and it offers 2% interest on all your accounts. You can now monitor all of your Golden Bank accounts from home. If you're interested, click on the link below to sign up. The promotion ends on July 15.

2. Add the following text to the Details text box “Do not add any additional information.”

Details
Characteristics:  Company - Golden Bank  Offer includes - no fees, 2% interest rate, no minimum balance  Tone - informative  Response requested - click the link  End date - July 15  Do not add any additional information.] <span style="border: 2px solid red; padding: 2px;">2</span>

Click Generate and take note of the output:

Email
We've got great news for you! Golden Bank is now offering Online Banking -- no fees, 2% interest rate, no minimum balance. To find out more, click the link below. Our current promotion ends July 15. Click here for details.

Compared to the original output, the email no longer talks about “You can now monitor all of your Golden Bank accounts from home.”

This looks to have removed the made-up text with this model. Next, you will switch to a different model and see how it works.

3. Switch the model (if necessary, reference Section 6.2, Steps 7-10 on changing model) to the flan-ul2-20b model. In Section 6.2, you have seen that this model (without the modified prompt) hallucinated and provided a made-up number 713-852-1414.

Now, with the added text in the prompt, click Generate to get the following output:

Email
We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out on this deal; it won't last long. Click here to find out more: <a href="http://www.goldenbank.com/feefreepromotion.htm">http://www.goldenbank.com/feefreepromotion.htm</a>

This is a very interesting output. The model no longer hallucinates a phone number. While the front part of the output resembles that from the flan-t5-xxl-11b model (with just the right info), the model still cannot resist inventing a website.

We have already seen that flan-ul2-20b tends to be more “creative” than flan-t5. In this particular case, the extra filter you used is not quite enough to stem flan-ul2’s creativity. Next, you will try something more explicit.

4. Go back to the Details input field. Instead of “Do not add any additional information”, be more explicit and use: “Do not make up any phone number or website”.

Try ^

Test your prompt [\(i\)](#)

**Details**

Characteristics:

Company - Golden Bank

Offer includes - no fees, 2% interest rate, no minimum balance

Tone - informative

Response requested - click the link

End date - July 15

Do not make up any phone number or website.

4

5. Now when you click Generate, you will see the following output:

## Email

We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and the ability to open an account with just \$500. To sign up, click the link below. Offers are available until July 15th.

The made-up website is gone. However, the generated output still includes a hallucination. Remember, generative AI models are not answering questions but are generating the best probable text based on input. The larger the model, the more creative the completion tends to be. In this case, the model calculates from the prompt that it is undesirable to generate a phone number or a website, but it probably was trained with finance data that made it determine that an account with a minimum balance of \$500 is of high probability and added it to its output.

6. Because flan-ul2-20b is creative, you should try something very explicit. This time, update the input text with: “Do not include any other information not provided above”.

Try ^

Test your prompt (i)

### Details

Characteristics:

Company - Golden Bank

Offer includes - no fees, 2% interest rate, no minimum balance

Tone - informative

Response requested - click the link

End date - July 15

Do not include any other information not provided above.

6

Now click Generate, and you will see the following output:

### Email

We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out-- sign up today by clicking the link below. Offer ends July 15.

This now is the proper email with no hallucination.

- Now let's try this with the llama-2-70b-chat model. We saw what it would do with the base prompt (it added some bogus dates and a link). Now run this updated prompt and you will see the following output:

### Email

Subject Line (60) characters or less including spaces | Marketing Message 984 total character limit including space between lines but excluding subject line| Link to GolenBank website at end of email(not included in char count). Keep sentences short and professional tone as this is sent by compliance dept on behalf CEO's office.)" data-reactid=".r[pqeee]:osis\_mc\_embedded".</data> Subject: Take Advantage Of Our Exclusive Checkings Account! goldenbankusmail campaign ends soon don't miss out!!!! Compliance Depratment announces an incredible offer from our partnership team that you won t find anywhere else Click Here<https://wwwgoldenbankecom/> today before its too late All new checkinf accounts opened through <https://wwwexamplecon7julyl3w Register nowand discover what checking should be

You might not have the exact output, but it should still be very verbose and similar to the above completion.

This is a great surprise. Instead of reining in the model, it has become loquacious (talkative) and at times incomprehensible. This model's generated email is not close to what you want to send. It talks about character count, 984 total characters, CEO office, bad spelling, and missing information (July 15). The model did not complete its output and only stopped because there is a limit to the number of output tokens (you will work with this in the next section of the lab).

Now, change the prompt once more. Insert the phrase "Create a succinct email" before "Do not include any other information not provided" (see below):

Try ^

Test your prompt [\(i\)](#)

**Details**

Characteristics:

Company - Golden Bank

Offer includes - no fees, 2% interest rate, no minimum balance

Tone - informative

Response requested - click the link

End date - July 15

7

Create a succinct email. Do not include any other information not provided above.

With this appended detail, you will see the following output when you click Generate:

Email

To [Name] Re Introducing our Best Checkings Account Ever! No Hidden Fees and Now Earn up to an industry leading 2%. Click here before they're gone forever on July 6th at Midnight PST <https://www/goldenbankUSAcheckinaccounts>

Although this still includes some hallucination (a made-up URL) and a date error (July 16<sup>th</sup>), this completion now looks much more like the email you expect.

Note: The llama-2-70b-chat model is huge and can be more unpredictable. You might not see exactly this output. You should still get a much smaller output than the previous one.

8. You can experiment further.

- Try changing the prompt to get a feel for how the flan-ul2-20b model may respond to different prompt inputs. If you change the wording above, how does the model respond?

- Try other models besides flan-ul2-20b. How do they respond to prompts that worked well with the flan-ul2-20b model?
- Which model seems to have the most difficult time responding to your prompt with the proper response?

### Section summary

- In this section, you learned that even with zero-shot prompting, it is possible to modify the prompt to get a better response from foundation models.
- Keep in mind that the model is not explicitly “answering” a question or a query, but is looking to extend/generate a response based on the input. This means that you often need to help the model to understand how your input is structured.
- Different models may require different prompts. For larger and more “creative” models, the instruction may need to be more explicit.
- As an aside, models such as ChatGPT have been heavily optimized. It has learned from a lot more examples than most general Large Language Models (a type of foundation model) and has also been tuned with methods such as Reinforcement Learning from Human Feedback (RLHF) to better “understand” human instructions. They are more expensive to run and better able to interpret general questions, but they still can hallucinate.
- Models such as flan-ul2-20b and flan-t5-xxl-11b have not been optimized that way and will need good prompts and few-shot prompting (you will do this later in this lab).
- It is tempting to think that larger models are “smarter”. There is certainly some truth in that, but these models can also be too creative. As you have seen you may have to be very explicit to get the output you are looking for. Note that creativity is not necessarily an issue. This is one of the attractions of generative AI and foundation models. But if you are not trying to generate creative content (as is the case in this example – you want your email to mention the specific points and not add fictitious content), you might want to consider a smaller (and less creative) model or provide stricter prompts to rein in the models.

## 6.4 Prompt Engineering - modifying parameters

In this part of the lab, you will look at how to modify some of the model's inference (runtime) parameters and see how they affect the output of different models.

1. Open a new Prompt Lab session and select the icon from the left to bring up the list of sample prompts. Click the Marketing email generation sample from the Generate section. Use the flan-ul2-20b model if not already specified.

The prompt should read as follows:

- For Instruction  
Generate a 5 sentence marketing message for a company with the given characteristics.
- For Details under the Try section

Characteristics:  
Company - Golden Bank  
Offer includes - no fees, 2% interest rate, no minimum balance  
Tone - informative  
Response requested - click the link  
End date - July 15

If you cannot find this prompt example or if the text content has changed, please use the text provided above to ensure you will see the same completion as shown in this part.

You have seen that the flan-ul2-20b model will hallucinate and generate a phantom phone number: 713-852-1411 (see Step 11 of Section 6.2). Click Generate to see that same output again.

2. Recall from Section 6.3 that you can modify the prompt (to a degree) so that the model will not generate any false phone numbers or URLs. In this part of the lab, you will change the prompt a bit to see the effect of changing the parameters on a completion. Go to the Details text field and add the following to the original prompt: Do not include any additional information and click Generate.

## Details

Characteristics:

Company - Golden Bank

Offer includes - no fees, 2% interest rate, no minimum balance

Tone - informative

Response requested - click the link

End date - July 15

2

Do not add any additional information.

You will see the following output:

## Email

We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out on this deal; it won't last long. Click here to find out more:

<http://www.goldenbank.com/feefreepromotion.htm>

As can be seen, the model has made up a website (as it did before).

- Now, we will modify some inference parameters to see the impact on the completion instead of further changing the prompt. On the console's upper right-hand corner, click the icon.

The screenshot shows the Prompt Lab interface. On the left, there's a sidebar titled 'Sample prompts' under 'Generation' and 'Extraction'. Under 'Generation', there are two items: 'Marketing email generation' (Generate email for marketing campaign) and 'Thank you note generation' (Generate thank you note for workshop attendees). Under 'Extraction', there are two items: 'Named entity extraction' (Find and classify entities in unstructured text) and 'Fact extraction' (Extract information from SEC 10-K sentences). The main area has tabs for 'Structured' and 'Freeform' (the 'Freeform' tab is selected). Above the main area, there are buttons for 'New prompt +', 'Save work', and a red box containing the number '3'. The preview window shows a generated email with the subject 'We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out on this deal; it won't last long. Click here to find out more: http://www.goldenbank.com/feefreepromotion.htm'.

- This opens a window that contains a list of parameters that you can modify. While you can change the model (using the model list drop-down), all models have the same inference parameters. Let's look at the different runtime parameters that you can work with.

The inference parameters you can adjust include:

- Greedy vs. Sampling Decoding – This is how the model chooses the tokens for output. In Greedy mode, the model selects the highest probability tokens at every step of decoding, and a model is less creative in that mode. In the Sampling mode, the model tends to be more creative and random, but there is a larger risk that the output may be nonsensical.

The first 4 parameters below (b through e) are only visible when in Sampling mode.

- Temperature – a floating point number ranging from 0.0 (which causes the model to operate in the same manner as if Greedy decoding was selected) and 2.00 (which is maximum creativity).

The higher the value of Temperature, the more “creative” the model would be.

The default value is typically 0.7 but is set to 0.8 in this case. This means the model is allowed certain creativity.

- Top P (nucleus sampling) – a floating point number ranging from 0.0 to 1.0. At a high level, when generating a completion, a model calculates the probability of the next word based on all the previous words. Top P decides whether the

model will always pick the most likely outcome, or allow more randomness for the next words.

Top P sampling chooses from the smallest possible set of “next” words whose cumulative probability exceeds the probability p. This way, the number of words in the set can dynamically increase and decrease according to the next word probability distribution. Take, for example, a Top P value of 0.8.

This means that:

- If the probability of the most likely next word is  $> 0.8$ , then the model will return that word as the completion.
- If the probability is  $\leq 0.8$ , then the model will look at the next most likely word and sum up the probabilities. If the sum is now  $> 0.8$ , the model will choose randomly from between the top 2 words as the completion.
- If the sum is still  $\leq 0.8$ . The model will look at the third most likely word, sum the probabilities and so on. It will continue until it finds a candidate list of words whose probabilities sum up to be  $> 0.8$ . The model will select its next output word randomly from this list.

This means the higher the value of Top P, the larger the candidate list of words and so the more random the outcome would be. At high values of P, the model is potentially including a lot of candidate words of low probability, making the outcome more “creative”.

8. Top K – an integer from a range from 1 to 100. Unlike Top P, Top K does not look at the probability of the words. Instead, a model simply chooses from the Top K mostlikely words to be the output. For example:

- Top K = 3, the model will choose randomly among the top 3 most likely next words.
- Top K = 5, the model will choose randomly among the top 5 most likely next words.

The higher the value of Top K, the more random the outcome would be.

9. Random seed – is an integer in the range 1 to 4,294,967,295

- In the Sampling mode, you can generate different outputs from the model by using a different Random seed value each time.

- If you do not change it, you should get back the same results.
- For the best result of the labs, do not change the Random seed value as that will create different results. Quite simply, a Random seed is helpful for the replicability of experiments.

10. Repetition penalty – a value between 1 and 2 (a setting of 1 allows repetition, and 2 prohibits it). This setting is used to counteract a model’s tendency to repeat the prompt text verbatim or be stuck in a loop in generating the same output.

11. Stop sequences – sequences of characters (can be text, carriage return, etc.). When the model encounters one of the sequences, it will stop generating output (the sequence itself will be included in the output). For example, suppose a model gave you the following output:

We are thrilled that you could make it to our workshop and hackathon today. It was great to meet everyone and we hope you enjoyed yourself. Thanks again for your participation and have a great weekend! Thank you for your participation. Thank you for being on time.

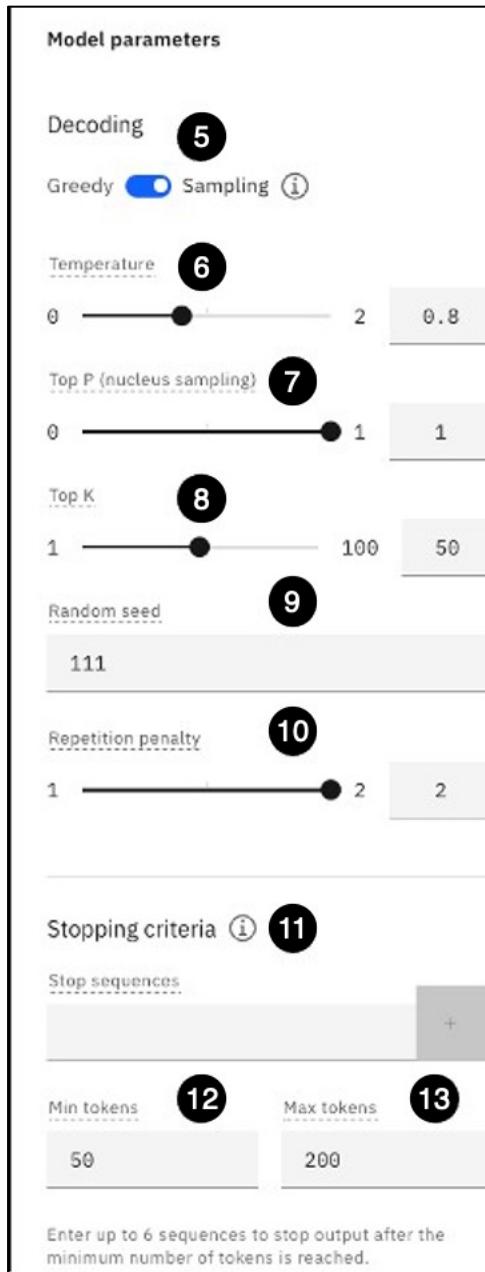
The output is too verbose. If you specify have a great weekend! as a stop sequence and you click Generate, you will get this output:

We are thrilled that you could make it to our workshop and hackathon today. It was great to meet everyone and we hope you enjoyed yourself. Thanks again for your participation and have a great weekend!

The model encountered the phrase “have a great weekend!” and it stopped generating any further output.

12. Min tokens – an integer that specifies the minimum number of tokens in the model’s output. Remember that tokens do not map 1-1 to words.

13. Max tokens – an integer that specifies the maximum number of tokens in the model’s output. Sometimes when the generated output looks incomplete, your Max tokens value may be too low.



Another thing to keep in mind is this: typically, you are charged by the number of tokens used, so setting a super high Max tokens value can be costly.

14. The default values used here are Temperature = 0.8, Top P = 1, and Top K = 50. This has resulted in some “creativity” with the flan-ul2-20b output on Marketing email generation:

**Email**

We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out on this deal; it won't last long. Click here to find out more: <http://www.goldenbank.com/feefreepromotion.htm>

You will see how changing parameter values can change the output. Leaving the “Do not add any additional information” in the input Details section, change the value of Temperature from 0.8 and 0.4.

The screenshot shows the Qwen AI interface. At the top, there are tabs for "Structured" and "Freeform", with "Freeform" selected. The model name is "flan-ul2-20b". In the "Set up" section, there's an "Instruction (optional)" field containing "Generate a 5 sentence marketing message for a company with the given ...". Below it is an "Examples (optional)" section with a table titled "Details" and "Email". A button "Add example +". In the "Try" section, there's a "Generate" button. On the right, under "Model parameters", there are several sliders: "Decoding" (Greedy Sampling), "Temperature" (set to 0.4, highlighted with a red box and a circled number 14), "Top P (nucleus sampling)" (set to 1.0), and "Top K" (set to 50).

Now click **Generate**. You will see the following output:

**Email**

Golden Bank is offering a free checking account with no fees, a 2% interest rate, and no minimum balance. To apply, click on the link below. Offer ends July 15. Thank you for your time. Best Regards, [name]

Note the change in the Temperature value does have a positive effect. The generated email no longer contains a made-up website. Lowering the temperature has the effect of making the model less creative, and in this case, stopping a hallucination.

15. Next, you will look at the effect of changing the Top P parameter. Go back to the list of parameters and reset the Temperature to 0.8 and change the Top P value from 1.0 to 0.5.

The screenshot shows the Watson Assistant interface with the following details:

- Structured** and **Freeform** tabs are at the top left.
- The model selected is **flan-ul2-20b**.
- Set up** section includes:
  - Instruction (optional)**: Generate a 5 sentence marketing message for a company with the given characteristics.
  - Examples (optional)**: A table with columns **Details** and **Email**. An example row has a "Add example" button.
- Try** section is collapsed.
- Model parameters** section on the right:
  - Greedy** toggle is off, **Sampling** is on.
  - Temperature**: Sliders from 0 to 2. The current value is 0.8, highlighted with a red box and labeled **15**.
  - Top P (nucleus sampling)**: Sliders from 0 to 1. The current value is 0.5, highlighted with a red box.
  - Top K**: Sliders from 1 to 100. The current value is 50.

This setting means that the model will only consider the top set of words whose sum of probabilities adds up to  $> 0.5$ . It is removing a lot of the low-probability words. When you click Generate, you will see the following output:

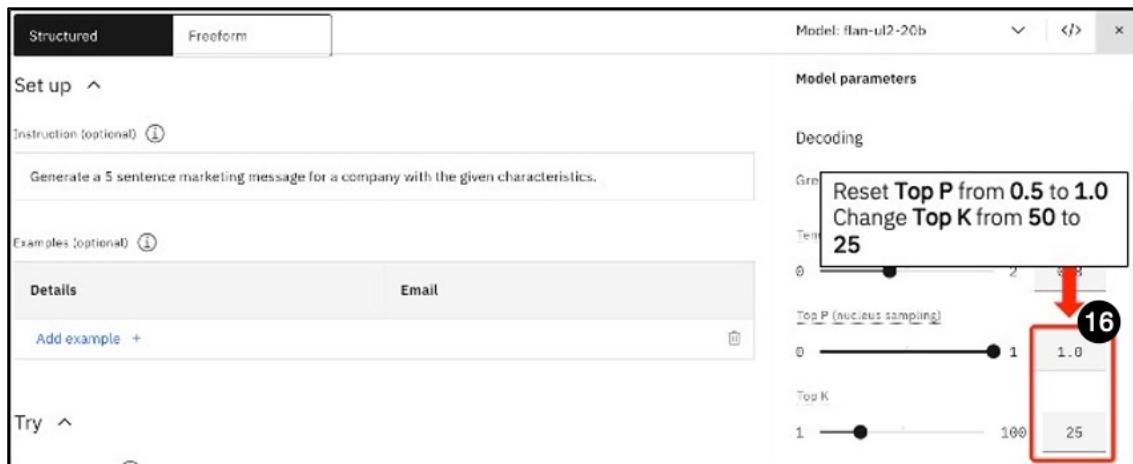
The generated email content is:

**Email**

Golden Bank is offering a free checking account with no fees, a 2% interest rate, and no minimum balance. To apply, click the link below.  
Offer ends July 15. Thank you for your time. Sincerely, [name removed]

This is again the kind of email you can send.

16. Now you will test another parameter. Reset the value of Top P to 1.0, and set the value of Top K (in half) to 25 as follows.



Now click Generate to get the following output:

Email

We've got great news for you! Golden Bank is now offering accounts with no fees, a 2% interest rate and no minimum balance. Don't miss out on this deal; it won't last long. Click here to find out more: <http://www.goldenbank.com/feefree>

This shows that halving the value of Top K to 25 does not have the same kind of effect as halving Top P. The output is still hallucinating with the web link.

17. Continue to lower the value of Top K. You will find that you must lower it by quite a bit (to around 4) before you get a fairly good email as follows.

Email

We've got a special offer to help you save money! Open an account with us and enjoy no fees, a 2% interest rate, and no minimum balance. Click below for details. Offer ends July 15. Thank you. Golden Bank

Why the difference? It is useful to look at the difference between Top P and Top K which is best illustrated with an example.

Example: Difference between Top P and Top K

Consider this following example – a model has determined a set of potential next tokens for a particular input as follows:

Token	Probability
T1	0.16
T2	0.15
T3	0.14
T4	0.13
T5	0.12
T6	0.11
T7	0.10
T8	0.09

In this case when you set:

- Top P = 0.5, the model chooses randomly among T1, T2, T3, and T4 (because the probabilities sum up to  $> 0.5$ ).
- Top K = 50, the model chooses randomly among all 8 of the tokens (as  $8 < 50$ )

You need to lower Top K to 4 to have a similar effect as Top P = 0.5 in this case. In that case, both settings ask the model to choose randomly from the top 4 tokens.

On the other hand, if the model sees the following:

Token	Probability
T1	0.92
T2	0.02
T3	0.01
T4	0.01
T5	0.01
T6	0.01
T7	0.01
T8	0.01

Then when you set:

- Top P to be 0.1, 0.2, 0.3, ... down to 0.9 – it makes no difference. In each case, the model would only consider token T1 (as T1 has a probability larger than the value of Top P).

- Top K to be 10, 20, 30, ... up to 90 – again it makes no difference. In each case (since there are only 8 possible tokens, and 8 is < 10, 20, .... 90), the model will randomly select from all 8 tokens.

It might look as if varying the values of Top P or Top K seem to have little to no effect. But remember a model determines how many possible next tokens and what the probabilities will vary, and generally, there are a lot of possible tokens. However, depending on how the model is designed and trained, varying Top K or Top P will have different effects.

### Section summary

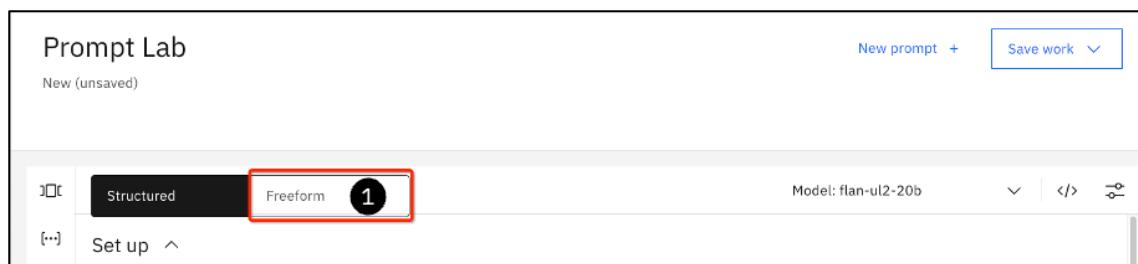
- In this section, you have examined the effect of changing some configuration parameters.
  - Temperature – the higher the value the more creative the model is
  - Top P – lower value means less variability
  - Top K – lower value means less variability
- Top P and Top K are similar in limiting the tokens that the model will choose from to create an output but can have very different effects depending on the model (which has different ways of evaluating possible tokens) and calculating the probabilities.
- This exercise only examines varying the parameters one at a time. Combined with additional input to the prompt and combinations of parameters, you can already do a fair bit in making a foundation model respond as you want.
- Note that not all foundation models will handle the same kind of tasks well. You can (and probably should) find out more about a model before using it in production: perhaps experiment on your own with this section using different models.

## 6.5 Working with lists

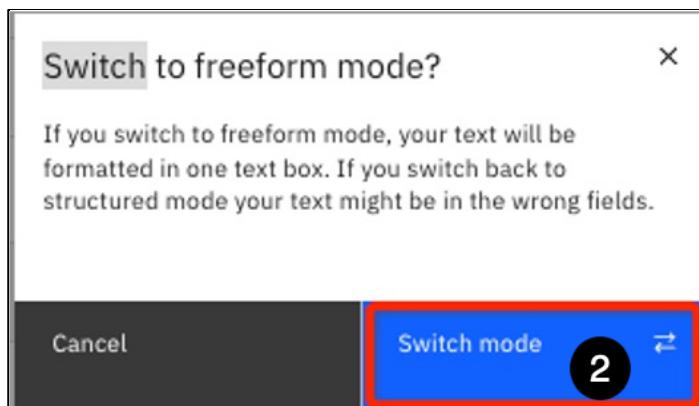
So far in this lab, you have looked at working with prompts and parameters to generate desirable email outcomes. In this part of the lab, you will generate a list output using the Prompt Lab in the Freeform mode.

It is important to remember that a foundation model does not read a prompt and interpret the intent like a human. It simply consumes a series of tokenized words and calculates the best next token. The concept of a list for output is simple for a human reader, here you will look at how LLMs might respond.

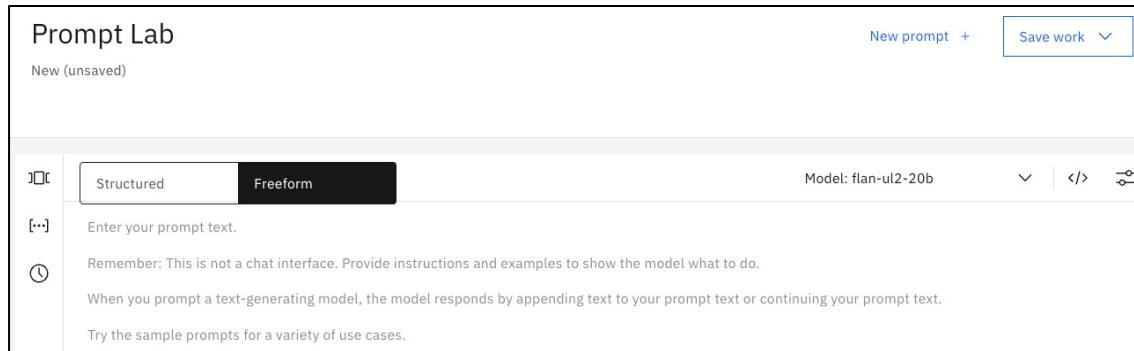
1. Open a new Prompt Lab session and select the Freeform tab (1) instead of the default Structured tab.



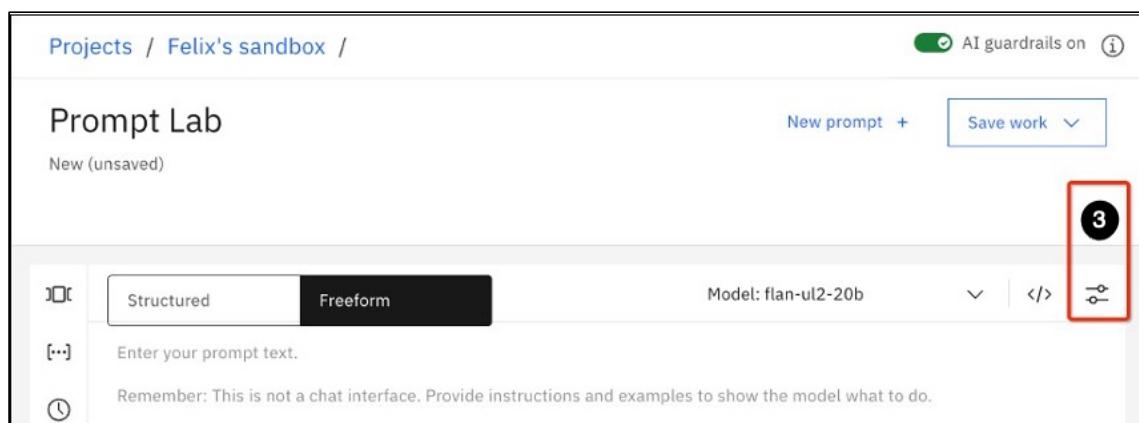
2. You will see the following panel. Click Switch mode.



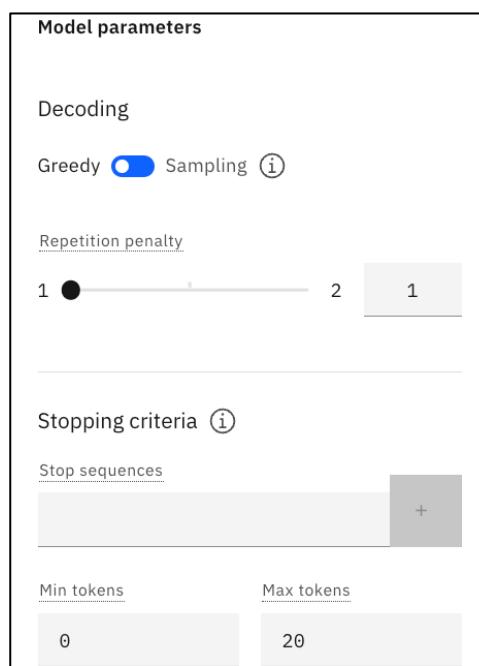
You will now be in the Freeform layout of the Prompt Lab. In the Freeform mode, you still have access to all the examples and all the LLMs but there is no structural guide.



- Click the icon at the upper right corner to slide open the configuration parameters.



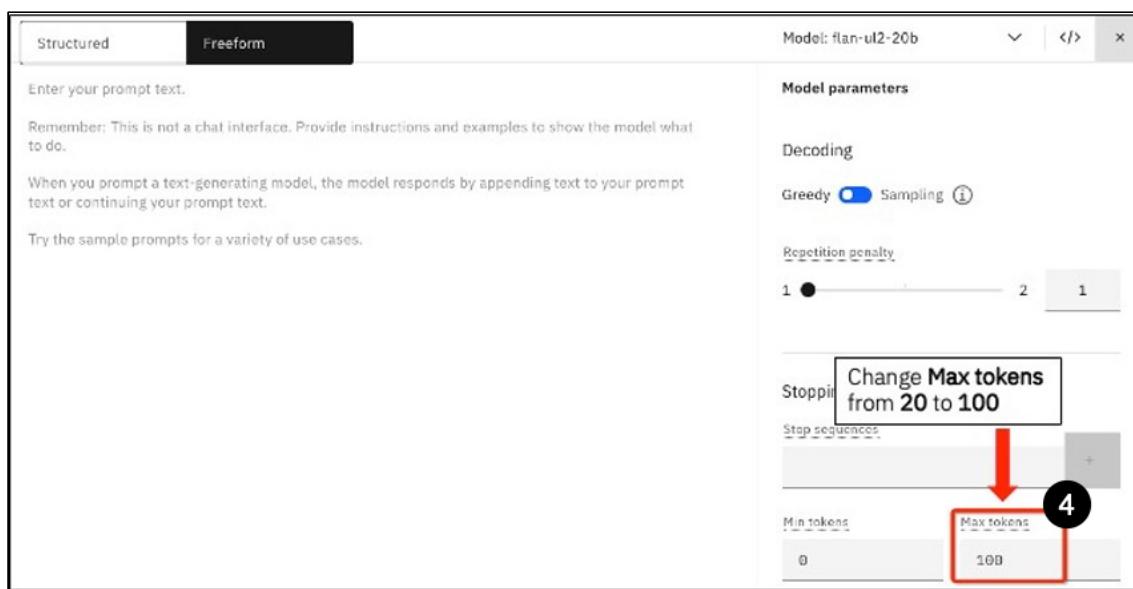
In Freeform mode, the parameters (and their defaults) look like:



Notes:

- These are the default if you have not selected a prompt sample. If you select a prompt sample, some of these values will change.
- The Min tokens value is 0. This means that the output can potentially be empty.
- Notice how the value of Max tokens is set to 20. This means that the output will be terse because there are so few tokens allowed in the generated output.

4. Verify that you are using Greedy decoding. Change the value of Max tokens to 100.



5. Ensure that you are using the flan-ul2-20b model. Make a change to flan-ul2-20b if necessary. Copy and paste the following text in the input field:

The following paragraph is a consumer complaint.

Read the following paragraph and list all the issues. I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up. The first time I got through, the line got cut suddenly and I had to call back. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was a complete waste of time. After like an eternity of this pointless conversation, I was told that an SME will contact me. That – was 2 days ago. What is the problem with your support system?

The list of issues is as follows:

Click Generate. You will see the following output (in red):

The following paragraph is a consumer complaint.

Read the following paragraph and list all the issues. I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up. The first time I got through, the line got cut suddenly and I had to call back. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was a complete waste of time. After like an eternity of this pointless conversation, I was told that an SME will contact me. That – was 2 days ago. What is the problem with your support system?

The list of issues is as follows:  
I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up. The first time I got through, the line got cut suddenly and I had to call back. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was

This is the output text:

The list of issues is as follows:

I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up. The first time I got through, the line got cut suddenly and I had to call back. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was

Note the following:

- The Generate button is not active until some text is entered into the field.
  - The model did not output a list.
  - The model just repeated the input text word by word.
  - This is not a desirable result – the model probably does not quite understand the idea of a list.
  - It terminates with “It was” because it was at the limit of 100 tokens (keep in mind that 100 tokens do not mean 100 words).
6. Switch to the flan-t5-xxl-11b model. Remove the generated completion and click Generate. Note how the model outputs the same text as flan-ul2-20b. Both of the flan models (without adding more to the prompt or changing parameters) seem to have understood that to “list” the issues is simply to recount what it was told.
  7. Tip – you can guide the flan models to provide a better output by adding a “1.” to your input prompt (this is telling the model you want to have a structured list output). Go back to the flan-ul2-20b model and update the prompt as follows ((highlighted in red below):

Structured      Freeform

Model: flan-ul2-20b

The following paragraph is a consumer complaint.

Read the following paragraph and list all the issues. I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up. The first time I got through, the line got cut suddenly and I had to call back. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was a complete waste of time. After like an eternity of this pointless conversation, I was told that an SME will contact me. That – was 2 days ago. What is the problem with your support system?

The list of issues is as follows:

1.

Click on Generate and the output looks like this:

The list of issues is as follows:

1. I called your helpdesk multiple times and every time I waited 10-15 minutes before I gave up.
2. The first time I got through, the line got cut suddenly and I had to call back.
3. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and know what I am doing.
4. After like an

Now the model outputs a list, even though it still is simply repeating the input.

Change to the model mpt-7b-instruct2 model. Remove the generated completion and the 1 from the prompt (return to the original prompt) and click Generate. You will see the following output:

The list of issues is as follows:

1. The customer waited 10-15 minutes before giving up.
2. The customer called back after the first call was cut.
3. The customer waited 3 days to get through.
4. The customer was asked to verify trivial things.
5. The customer was told that an SME would contact them.
6. The customer was frustrated with the support system.

This is actually a fairly good output summary and in list format. For this kind of request, the mpt-7b-instruct2 model is much better than the flan models. Moreover, it correctly interpreted the sentiment of the client – one of frustration.

8. Let's try one more model – the gpt-neox-20b model ((if necessary, reference Section 6.2, Steps 7-10 on changing model). Remove the generated output. Click Generate, you will get the following output:

The list of issues is as follows:

1. The agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing.
2. It was a complete waste of time.
3. After like an eternity of this pointless conversation, I was told that an SME will contact me. The following paragraph is a consumer complaint. Read the following paragraph and list all the issues. I called your

The output is in a list form so the gpt-neox-20b model can provide structured output. It has a problem stopping properly and is repeating the input at the end.

#### Section summary:

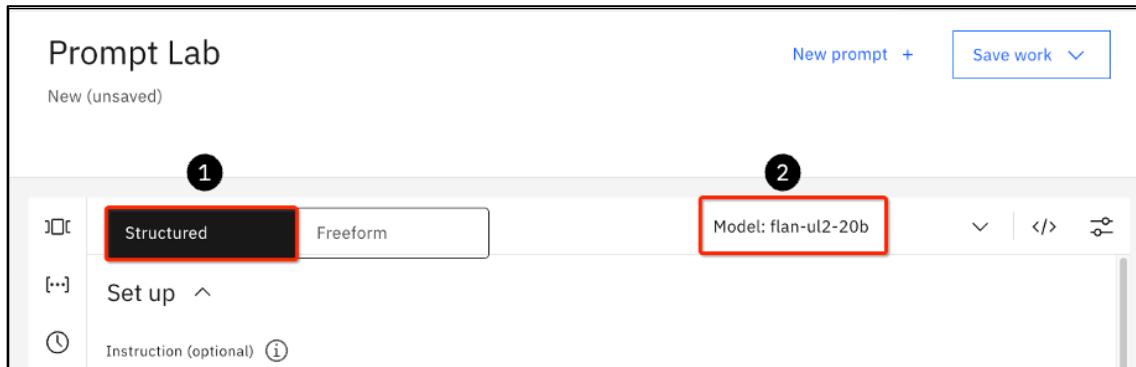
- In this section, you looked at the Prompt Lab using the Freeform mode. Once you have become more experienced with prompting, you may want to use this mode by default.
- Models handle output requirements differently. Some models (like the flan models) are better with text output (at least with zero-shot prompting). You can guide the output if you update the prompt with the hint of a “list structure”. Other models, such as the mpt- 7b-instruct2, and to a lesser degree the gpt-neox-20b model, are better able to understand the concept of list output without additional instructions.
- Once again, the size of the model does not matter – here the model with the smallest number of parameters (mpt-7b-instruct2) provided the best performance. Users should NOT be sidetracked by the size of the model. It is more important to understand the strengths of the model, and what it might have been trained and specifically tuned for.

In particular, while the mpt-7b-instruct2 model is smaller, it was trained by IBM to follow a user's instructions. This is why it is better able to understand prompts where the user is instructing the model to output in a particular (list) format.

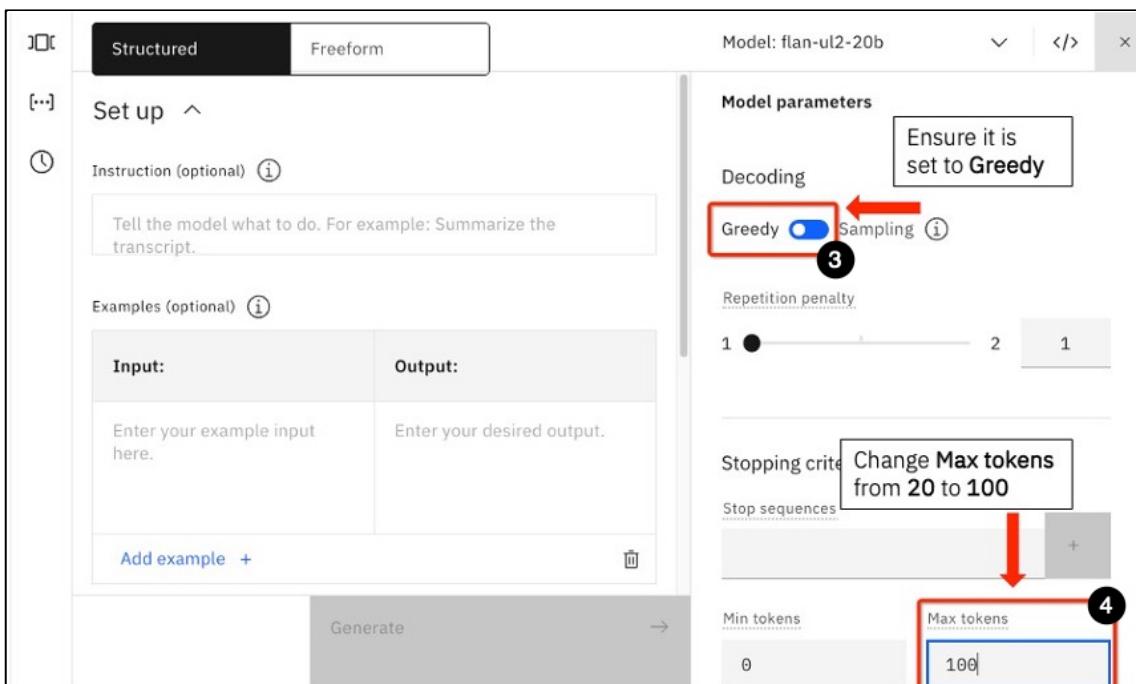
## 6.6 Working with JSON

In this section, you will use the Prompt Lab to generate a simple JSON file.

1. Open a new Prompt Lab session in the Structured mode.
2. Ensure that you are using the flan-ul2-20b model.



3. Ensure that you are using the Greedy mode.
4. Set Max tokens to 100.



5. Copy and paste the following to the Input field under the Try section

Create a JSON file output with the following information

name: Joe

age: 25

Phone: 416-1234-567

Phone: 547-4034-240

Address: City: Markham, Street: Warden Avenue, Postal Code: L6G 1C7

Your input field should now look like this:

Try ^

Test your prompt ⓘ

**Input:**

Create a JSON file output with the following information

name: Joe  
age: 25  
Phone: 416-1234-567  
Phone: 547-4034-240  
Address: City: Markham, Street: Warden Avenue, Postal Code: L6G 1C7

Click Generate. The flan-ul2-20b model returns the following output:

**Output:**

Joe, age 25, has two phone numbers, 416-1234-567 and 547-4034-240. His address is in Markham, Warden Avenue, L6G 1C7.

This shows the strengths of the flan-ul2-20b model in that it responded with a natural language output. However, it clearly is not the JSON output you are looking for.

6. Switch to the flan-t5-xxl-11b model and again click Generate. Similar to the flan-ul2- 20b model, you will see this output:

**Output:**

```
Joe is 25 years old and lives at Warden Avenue, Markham, Ontario,  
Canada. He can be reached at 416-1234-567 and 547-4034-240.
```

Again the completion provides a good natural language output, but it is not in JSON. At this juncture, you can try using one-shot or few-shot prompting to help the model understand how you want to structure the output. However, this would not be an easy task. Instead, we will move on and try other models.

7. Now try the mpt-7b-instruct2 model and click Generate. You will see this output:

**Output:**

```
{  
  "name": "Joe",  
  "age": 25,  
  "Phone": [  
    "416-1234-567",  
    "547-4034-240"  
  ],  
  "Address": {  
    "City": "Markham",  
    "Street": "Warden Avenue",  
    "Postal Code": "L6G 1C7"  
  }  
}
```

This is a valid JSON output for the given input. The mpt-7b-instruct2 has shown that it understands how to translate input text to JSON.

8. Now try another model that we have not yet experimented with. Select the starcoder-15.5b model and then click on Generate. You get the following output.

### Output:

```
{  
    "name": "Joe",  
    "age": 25,  
    "phone": [  
        "416-1234-567",  
        "547-4034-240"  
    ],  
    "address": {  
        "city": "Markham",  
        "street": "Warden Avenue",  
        "postal_code": "L6G 1C7"  
    }  
}  
....
```



The starcoder-15.5b model also generates a valid JSON output, except that it has triple double quotes (""""") at the bottom.

9. While this is not a big issue, we will now use the Stop sequences configuration setting to eliminate the triple double quotes.

In this particular case, you want to stop the generation once the model encounters the following text sequence (2 curly brackets that conclude the JSON section).

```
}  
}
```

In the upper right corner of the console, click icon to slide open the configuration parameters. Enter the following keystrokes on the Stop sequences field:

```
} <carriage return> }
```

10. You should see the following, then click on the blue + button to add the sequence.



The stop sequence is now added. You will see this:

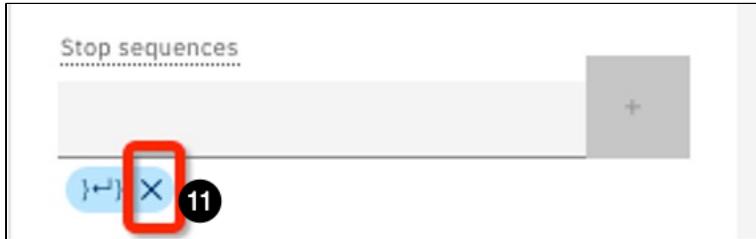


Now click Generate. The output (see below) no longer includes the triple double quotes as the model recognizes the stop sequence and ceases to generate anything more.

**Output:**

```
{  
  "name": "Joe",  
  "age": 25,  
  "phone": [  
    "416-1234-567",  
    "547-4034-240"  
  ],  
  "address": {  
    "city": "Markham",  
    "street": "Warden Avenue",  
    "postal_code": "L6G 1C7"  
  }  
}
```

11. You can remove the Stop sequence by clicking X.



Bonus tasks/questions:

- Remove any Stop sequences. What if you specify just one curly right bracket instead? Will it work?  
Try it out, does it do what you expected?
- Again remove the previous Stop sequence. What if you specify 2 curly right brackets (with no carriage return in between) like this }}}, will it work?  
Try it out, does it do what you expected?

### Section summary

- In this section, you tried generating a JSON output using different models.
- While the flan models have shown good capability in generating text in natural language, they do not seem to understand how to output in JSON format, at least not with zero- shot prompting.
- The mpt-7b-instruct2 model can generate good JSON output.
- The starcoder-15.5b model can also perform well.
- You also experienced using the Stop sequences to stop a model from outputting unwanted text. This is not a big issue with this exercise. However, there might be cases when a model generates a lot of unwanted text and consumes valuable tokens. You can use the same technique here to specify when the model should stop (you can specify multiple Stop sequences and the model will stop whenever it encounters any one of the Stop sequences).
- From the list output and the JSON output exercise, you learned 2 very important facts:
  - No one model will rule them all.
  - Different models are trained/tuned for different tasks and abilities. You should look for models that are trained with domain-specific data with a focus on various data structures and capabilities.

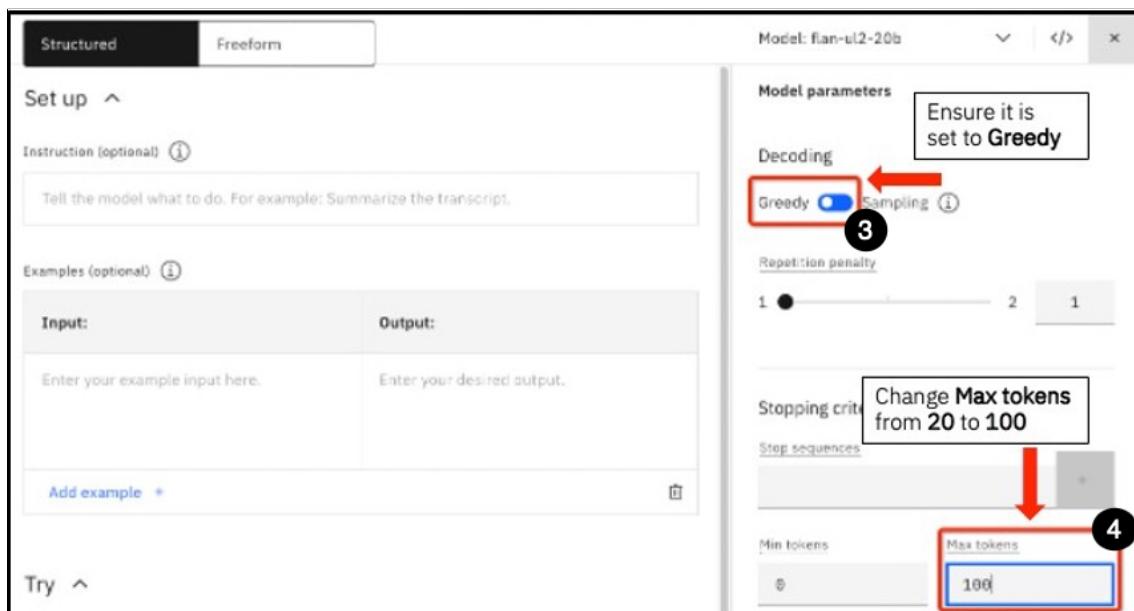
## 6.7 Working with one-shot prompting

So far you have been working with zero-shot prompting. In this section, you will experiment with one-shot prompting. For this section, you will be using the Structured mode.

1. Open a new Prompt Lab session. Ensure that you are using the flan-ul2-20b model.
2. Click on the  icon to slide out the configuration parameters (1).



3. Ensure that you are using the Greedy mode.
4. Change the number of Max tokens from 20 to 100.



5. Cut and paste the following text into the Try section's Input field.

The following paragraph is a consumer complaint. The complaint is about one of these options: credit cards, credit reporting, mortgage and loans, retail banking, or debt collection. Read the following paragraph and list all the issues.

I called your help desk multiple times and every time I waited 10-15 minutes before I gave up. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing, It was a complete waste of time. After like an eternity of this pointless conversation, I was told that an SME will contact me. That - was 2 days ago. What is the problem with your support system?

Your section should now look like this:

Try ^

Test your prompt ⓘ

**Input:**

The following paragraph is a consumer complaint. The complaint is about one of these options: credit cards, credit reporting, mortgages and loans, retail banking, or debt collection. Read the following paragraph and list all the issues.

I called your help desk multiple times and every time I waited 10-15 minutes before I gave up. This is just ridiculous. When I finally got through like after 3 days (yes, 3 days) your agent kept going over a long checklist of trivial things and asking me to verify, after I repeatedly told the agent that I am an experienced user and I know what I am doing. It was a complete waste of time. After like an eternity of this pointless conversation, I was told that an SME will contact me. That – was 2 days ago. What is the problem with your support system?

Click Generate. Your output will look like this:

**Output:**

Credit Cards

This is clearly not the output you want. This model took the first part of the prompt and simply read the content and determined that it was about credit cards. In this case, the model has performed a classification instead of a summarization.

You could try to update your prompt; however, this is not the focus of this part of the lab. Instead, you will try one-shot prompting – that means you provide an example to the model to “teach” (or steer) it to respond properly to the prompt.

6. Cut and paste the following text to the Set up section’s Input field in the optional Examples section.

The following paragraph is a consumer complaint.

The complaint is about one of these options: credit cards, credit reporting, mortgages and loans, retail banking, or debt collection. Read the following paragraph and list all the issues.

I bought a GPS from your store and the instructions included are in Spanish, not English. I have to use Google Translate to figure it out. The mounting bracket was broken, and so I need information on how to get a replacement. Moreover, the information seems to be outdated because I cannot see the new roads put in around my house within the last 12 months.

It should look like this.

## Set up ^

Instruction (optional) [\(i\)](#)

Tell the model what to do. For example: Summarize the transcript.

Examples (optional) [\(i\)](#)

**Input:**

The following paragraph is a consumer complaint.

The complaint is about one of these options: credit cards, credit reporting, mortgages and loans, retail banking, or debt collection. Read the following paragraph and list all the issues.

I bought a GPS from your store and the instructions included are in Spanish, not English. I have to use Google Translate to figure it out. The mounting bracket was broken, and so I need information on how to get a replacement. Moreover, the information seems to be outdated because I cannot see the new roads put in around my house within the last 12 months.

This is a sample input. You also need to provide a sample output to complete the one- shot prompt.

7. Cut and paste the following text into the Output field next to the Set up Input field:

The list of issues is as follows:

- 1) The instructions are in Spanish, not English.
- 2) The mounting bracket is broken.
- 3) The information is outdated.

Your screen should look like this:

Examples (optional) <a href="#">(i)</a>	
Input:	Output:
The following paragraph is a consumer complaint. The complaint is about one of these options: credit cards, credit reporting, mortgages and loans, retail banking, or debt collection. Read the following paragraph and list all the issues	The list of issues is as follows: 1) The instructions are in Spanish, not English. 2) The mounting bracket is broken. 3) The information is outdated.
<a href="#">Add example</a> +	<a href="#">Delete</a>

8. Click Generate to get a new output. You will see this in the Try section's Output field:

Output:
The list of issues is as follows: 1) Waiting time to talk to someone 2) Agent kept going over a long checklist of trivial things 3) Agent kept asking me to verify 4) Agent kept telling me that an SME will contact me 5) SME never contacted me <a href="#">X</a>

This output has been vastly improved from the previous output. Note how the completion listed all the valid issues and put them out in a numbered list.

ATTENTION: Do NOT close out the session or remove any information from any field. You will need this for the next section.

## Section summary

- In this section, you experimented with one-shot prompting and observed how providing an input example can help a model respond more desirably. An example can:
  - Teach a model to focus on the right information to use from the input; for example, ignore the type of complaint but focus on the substance of the complaint.
  - Teach a model how to generate a desired output; for example the structure – in our example a numbered list.
- In few-shot prompting, you would provide multiple input/output examples. Remember! Foundation models are not answering questions. However, they can learn patterns. A one-shot or few-shot prompt provides patterns to guide a model to generate output that has the best “probability” of matching the provided samples.

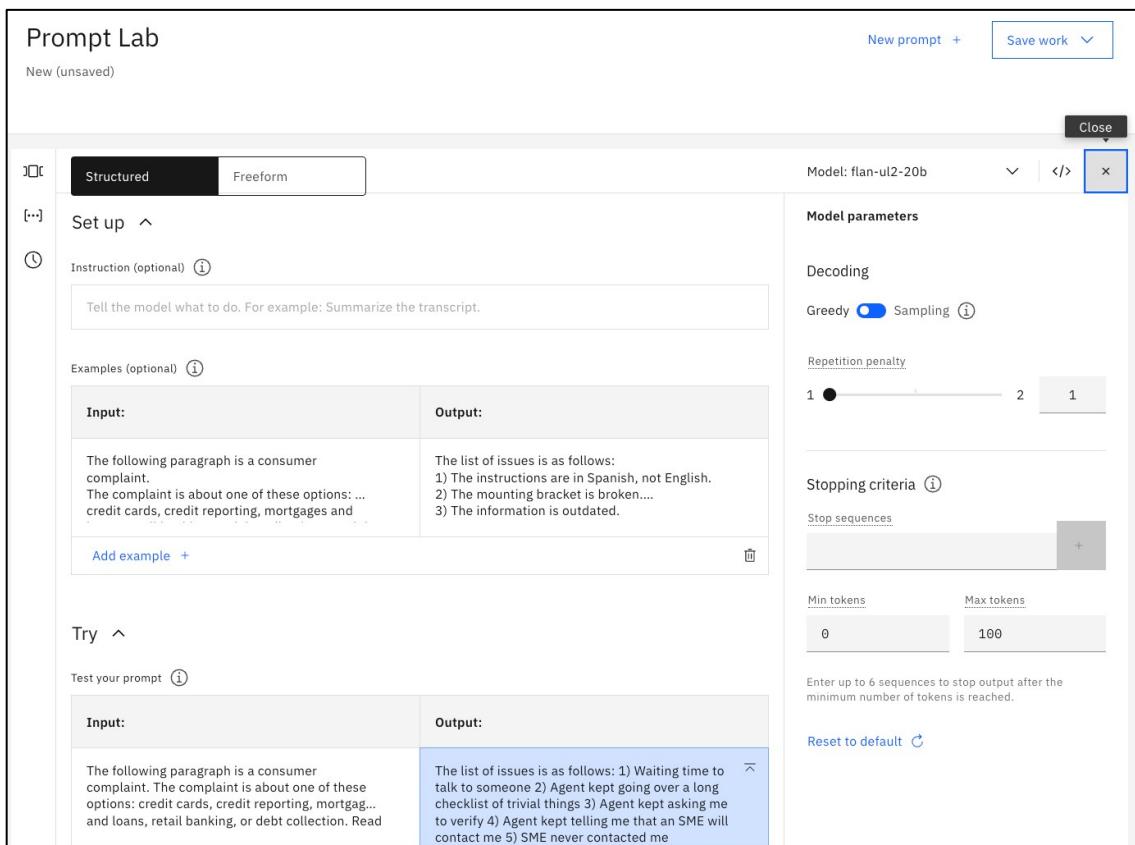
## 6.8 Saving prompts and prompt sessions

You have seen many examples of prompts and have done several prompt engineering examples. The more experienced you are, the better you will be able to modify a prompt and use different parameter settings to optimize the model for your desired output. You have also seen how to add additional examples to “teach” the model with single-shot in-context learning.

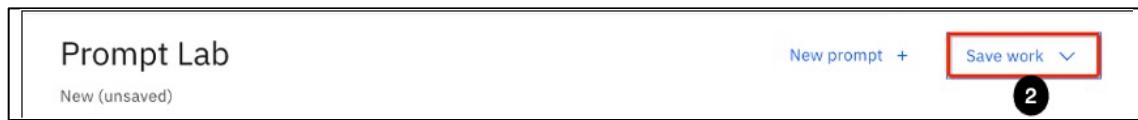
Sometimes you must modify a prompt several times in different ways to achieve the desired effect. You definitely want to save prompts that work as they are good examples and reminders.

You can save a “finished” prompt as follows.

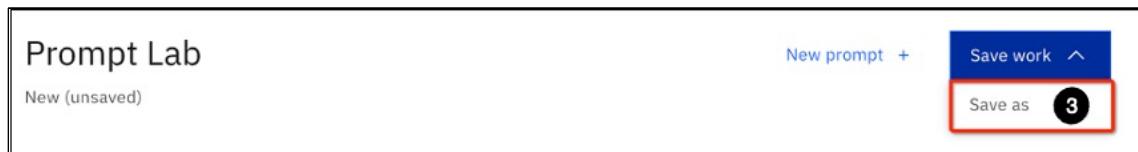
1. You should have the Prompt Lab still open with the modified prompt from Section 6.7. If not, you can re-generate it using the steps from 6.7. It should look like this:



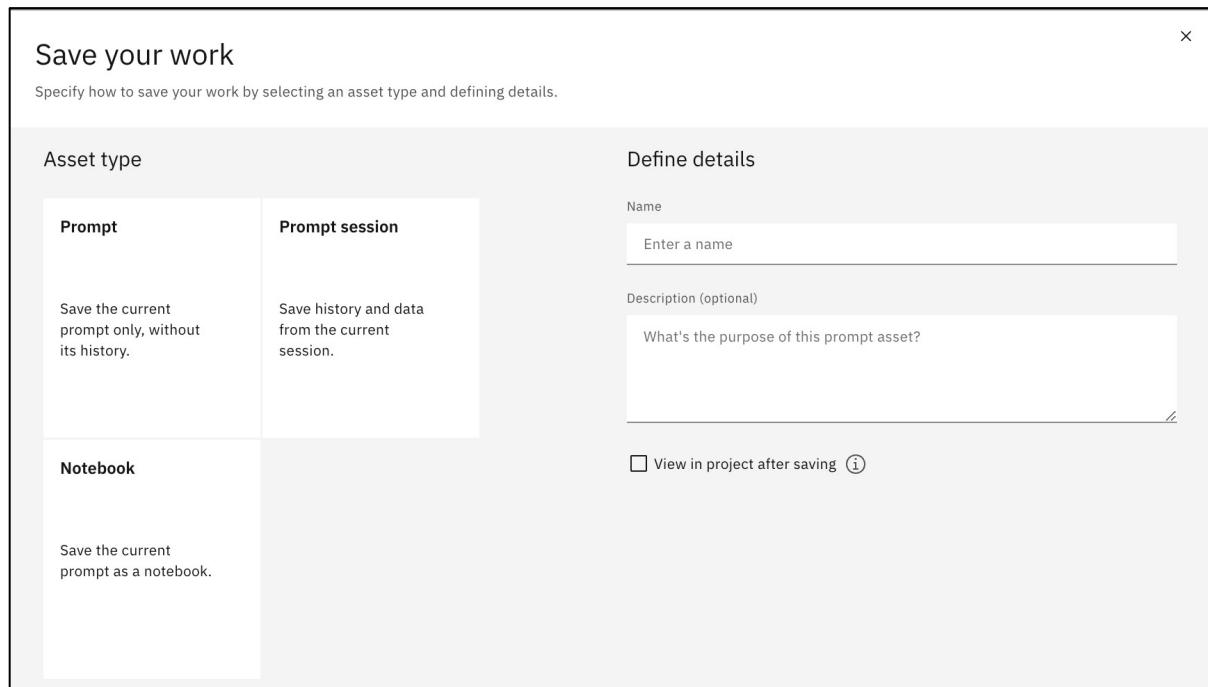
2. Go to the top right and click Save work.



- Now select Save as.



- You will now see the following panel:



You can save your assets in 3 different ways:

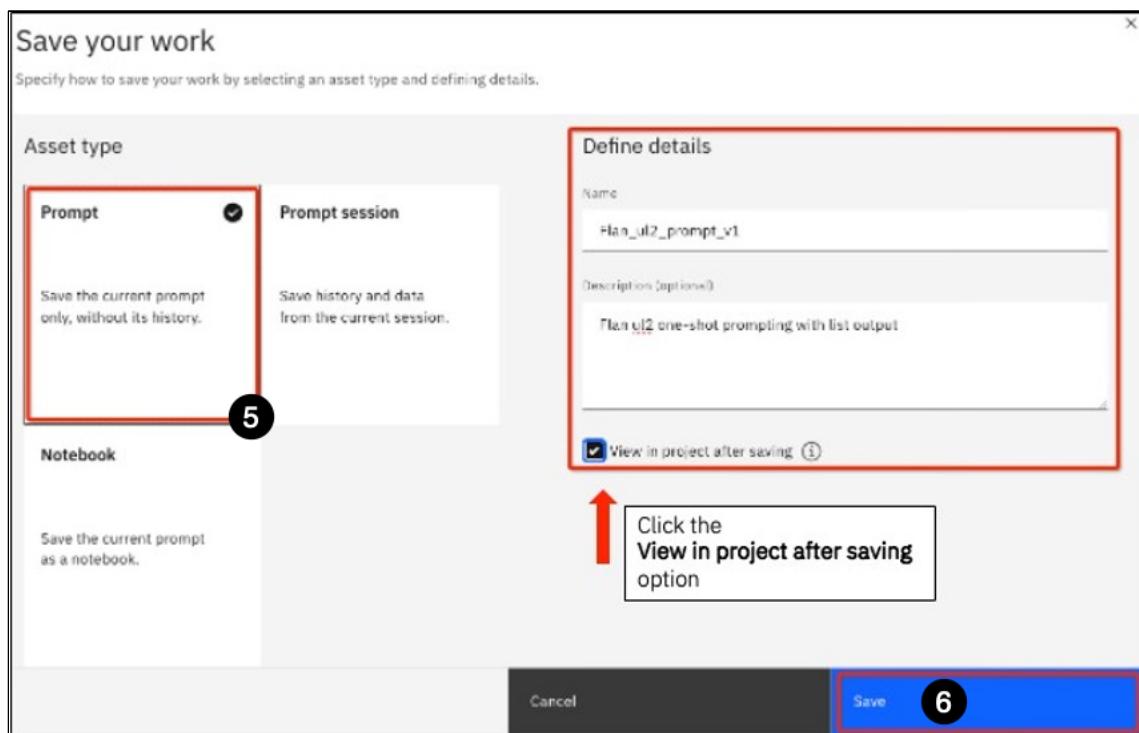
- As a Prompt – This saves the current state of the prompt which includes any input, output, and configuration runtime parameters you might have changed. Think of this as taking a snapshot of the panel. Do this when you find a combination that works well for your purpose.
- As a Prompt session – Saves the history of your prompt engineering session, tracking the changes in prompt text and configuration parameters (up to 500 steps). This is useful when you are experimenting and you want to go back to a previous setting that might have worked better.
- As a Notebook – Saves the information in a Jupyter notebook that you can edit and modify later. You will do this in a later section of this lab.

5. Select the Prompt tile, and enter the following contents on the right.

- Name: Flan.ul2\_prompt\_v1
- Description: Flan.ul2 one-shot prompting with list output.
- Select View in project after saving

It should look like this.

2. Click Save. Your project will open up to the Assets tab.



You can see your saved Flan.ul2\_prompt\_v1 prompt (you may or may not have other assets that will be shown here too).

3. Click on the Flan.ul2\_prompt\_v1 asset to load the saved snapshot of your prompt.

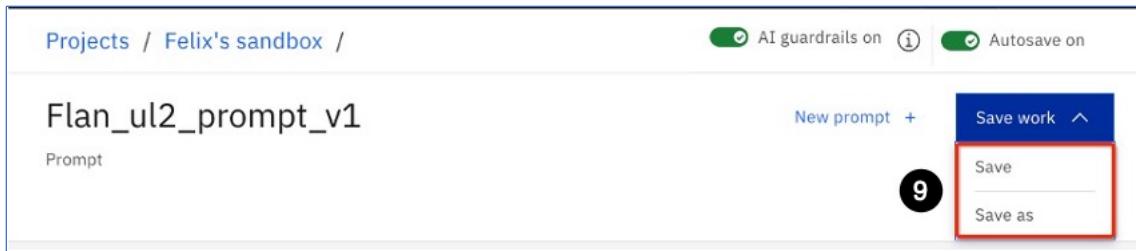
The screenshot shows the Project Overview page with the 'Assets' tab selected. On the left, there's a sidebar with 'Asset types' and a 'Find assets' search bar. The main area displays a table titled 'All assets' with two rows:

Name	Last modified
[...] Flan.ul2_prompt_v1 Prompt	Now Modified by you
Successful L2 example_1 Notebook	3 days ago Modified by you

You can save your prompt whenever you find something that is useful and come back to it later. However, there are times when as you change a prompt (adding text to the prompt, changing parameters, etc.) the results deteriorate and do not improve. In these circumstances, you may want to move back to a previous setting and change other parameters, or simply settle on a version that you tried to improve (but were not able to).

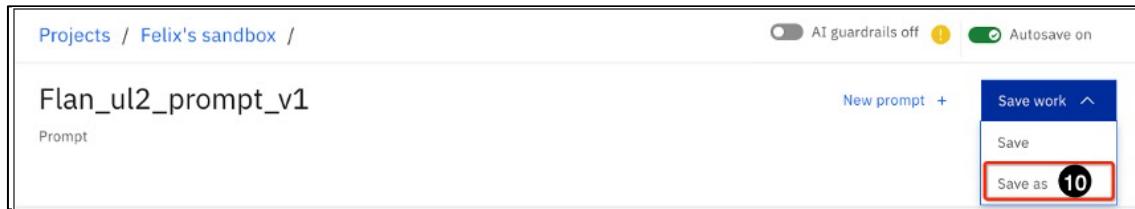
You can always save your prompt every time you make a change; however, it would be difficult to keep track and a better approach would be to save a prompt session.

4. On the opened Flan.ul2\_prompt\_v1 prompt, perform the following steps:
  - Select the icon to slide open the list of configuration parameters. You will make a series of changes. The focus of this exercise is tracking the prompt change. You should also observe the generated output to see how it changes as you modify the parameters. Ensure you click Generate at the end of each step.
    - Change the Repetition penalty to 1.8, then click Generate
    - Change the Repetition penalty to 1.6, then click Generate
    - Change the Repetition penalty to 1.4, then click Generate
    - Change the Repetition penalty to 1.2, then click Generate
    - Change the Decoding mode from Greedy to Sampling, then click Generate
    - Change Top P to 0.5, then click Generate
    - Change Top K to 25, then click Generate
5. Save this session by clicking the Save Work button. You will see the following:

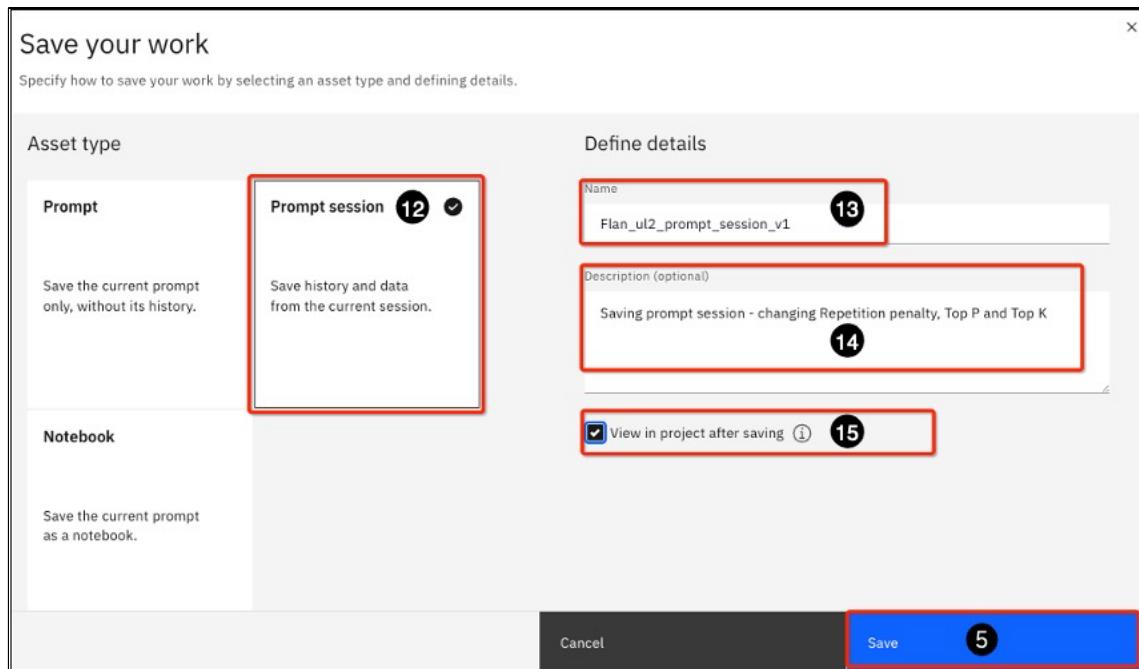


There are Save and Save as options (see 1 above). You started with a saved prompt, so if you select Save you will be updating that saved prompt.

6. Instead, you will select Save as to save this prompt session to a different asset – this time as a Prompt session.



7. You will see the Save your work panel that you saw in Step 5. Do the following:
8. Select the Prompt session tile and enter the following information on the right.
9. Name: Flan\_ul2\_prompt\_session\_v1.
10. Description: Saving prompt session - changing Repetition penalty, Top P and Top K.
11. Select View in project after saving.
12. Click Save.



13. On the project panel, you will now see this saved prompt session (as well as the previously saved prompt).

The screenshot shows the Watson Studio interface with the 'Assets' tab selected. On the left, there's a sidebar with '10 assets' and 'Asset types' sections. The main area is titled 'All assets' and lists two items:

Name	Last modified
[... Flan_ul2_prompt_session_v1 Prompt session	Now Modified by you
[***] Flan_ul2_prompt_v1 Prompt	4 minutes ago Modified by you

Select the Flan\_ul2\_prompt\_session v1 prompt session. You will see this:

The screenshot shows the configuration page for the 'Flan\_ul2\_prompt\_session\_v1' prompt session. At the top, it says 'Prompt session'. Below that, there are tabs for 'Structured' and 'Freeform', with 'Structured' selected. The 'Model' is set to 'flan-ul2-20b'. The 'Input' example provided is: 'The following paragraph is a consumer complaint. The complaint is about one of these options: credit cards, credit reporting, mortgages and loans, retail banking, or debt collection. Read the following ... paragraph and list all the issues.' The 'Output' example provided is: 'The list of issues is as follows:  
1) The instructions are in Spanish, not English.  
2) The mounting bracket is broken....  
3) The information is outdated.'

Below this, there's a 'Try' section with a 'Test your prompt' button. The input provided is: 'The following paragraph is a consumer complaint. The complaint is about one of these options: credit cards, credit reporting, mortgage and loans, retail banking, or debt collection. Read the following paragraph and list all the ... issues.' The output is: 'The list of issues is as follows: 1) Waited 10-15 minutes every time I called 2) Agent kept going over a long checklist of trivial things 3) SME has not contacted me'

14. Click the ⓘ icon (1) on the left to open the history of the prompt session.

The screenshot shows the 'History' panel with the title 'Prompt session' and the session name 'Flan\_ul2\_prompt\_session\_v1'. A circled number '18' is visible on the left side of the timeline. The timeline lists eight changes made at 3:39 PM:

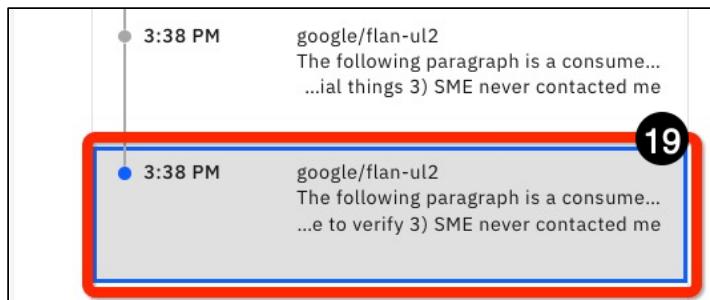
- Now: google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
- 3:39 PM: google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
- 3:39 PM: google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
- 3:39 PM: google/flan-ul2  
The following paragraph is a consume...  
...g. [Potentially harmful text removed]
- 3:38 PM: google/flan-ul2  
The following paragraph is a consume...  
...aited 10-15 minutes before giving up
- 3:38 PM: google/flan-ul2  
The following paragraph is a consume...  
...f trivial things and asking me to verify
- 3:38 PM: google/flan-ul2  
The following paragraph is a consume...  
...al things 3) SME never contacted me
- 3:38 PM: google/flan-ul2  
The following paragraph is a consume...  
...al things 3) SME never contacted me
- 3:38 PM: google/flan-ul2  
The following paragraph is a consume...  
...e to verify 3) SME never contacted me

This shows a history of the prompt session, recording the individual changes.

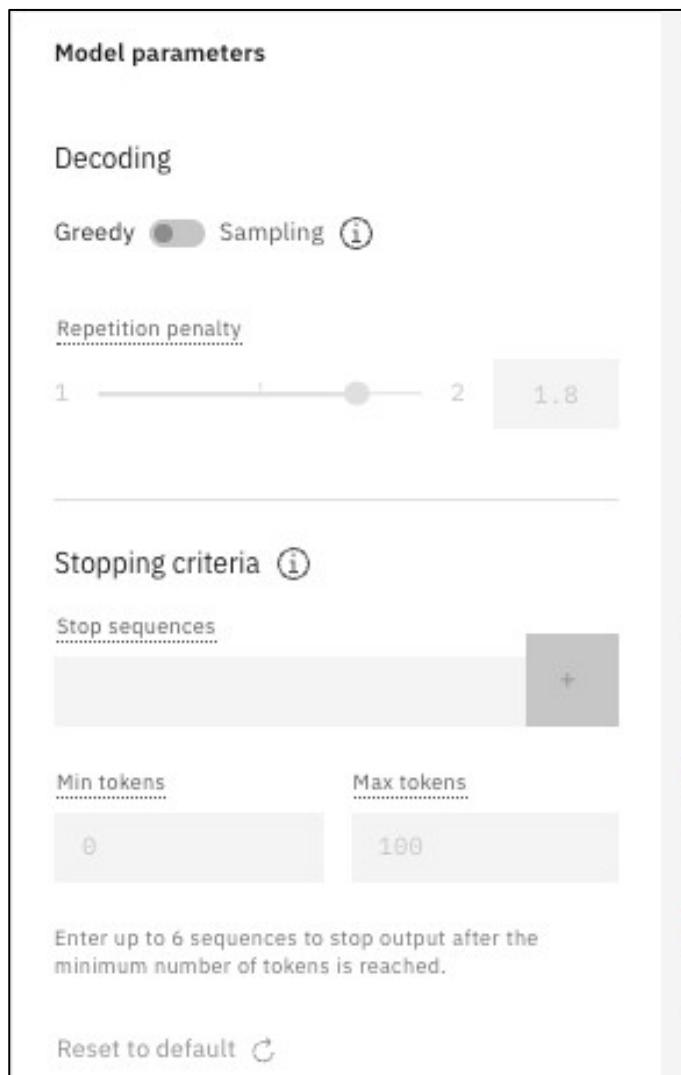
Note the following:

- Changes are recorded only if you clicked on Generate after the change.
- The session history includes the time of change (your timeline will be different from what is shown here).
- You can select a particular step to review the content of the prompt at that step.

15. Click the bottom entry from the Prompt session history.

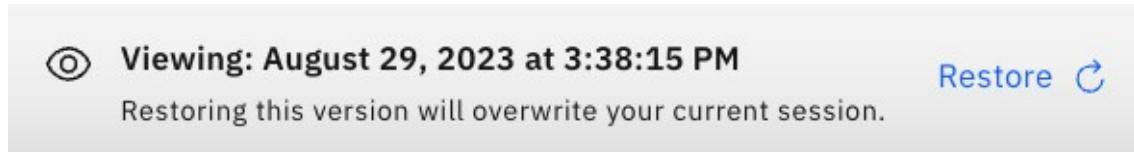


You will see the following information from the configuration panel (if it is not open, click the icon to slide it open).



Note that the value of the Repetition penalty is set to 1.8 (this was the first change you made in Step 7. Also note that everything is greyed out, as this is not the current prompt.

Looking at the bottom of the panel, you will see a message similar to the following:

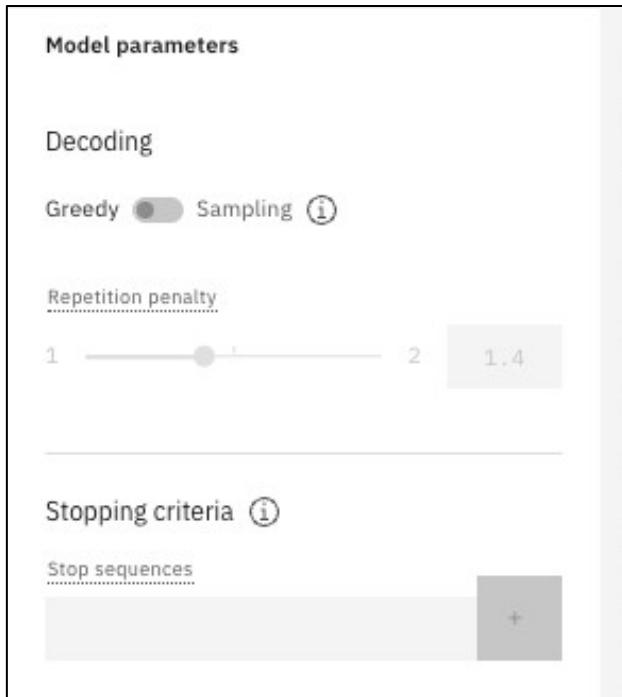


If you want this to become the current prompt, you can click Restore in the box. You will do that later in the lab.

16. Now click the third item from the bottom.



You now see the following configuration:



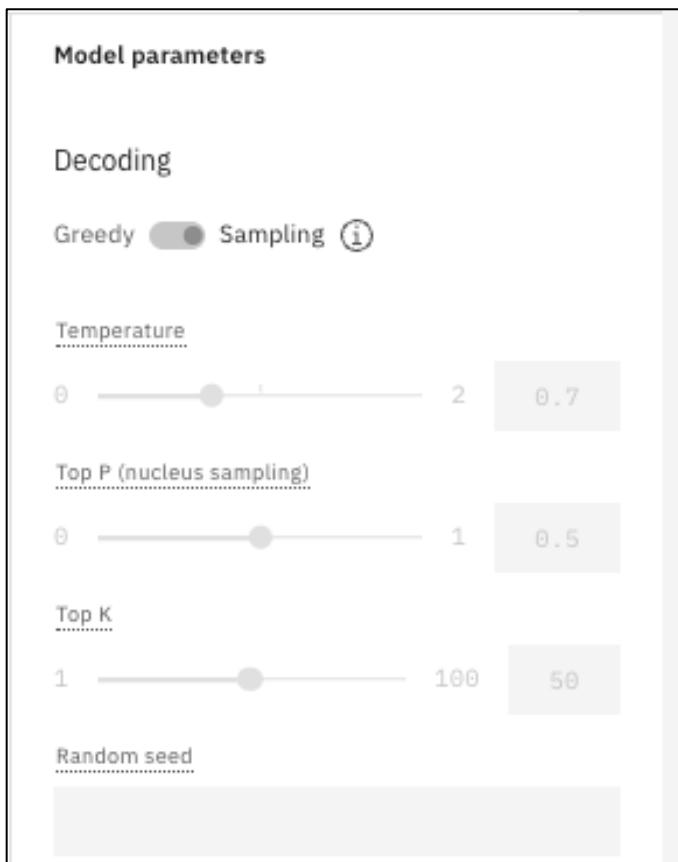
Again this corresponds to the third change you made in Step 7.

17. Suppose you decided the best prompt from the history was when Top P was set to 0.5 and Top K was set to 50. You click through the various stages and find that it is the fourth from the top.

The screenshot shows a 'Prompt session' interface with the following details:

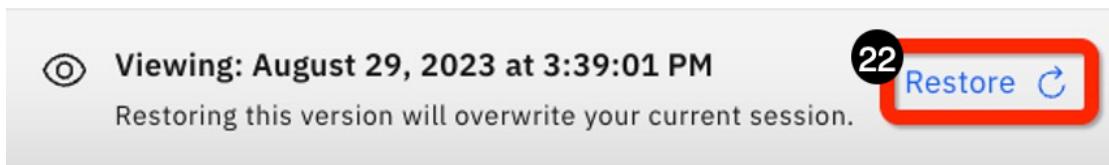
- Prompt session:** Flan.ul2\_prompt\_session\_v1
- Date:** Tuesday, August 29, 2023
- Messages:**
  - Now:** google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
  - 3:39 PM:** google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
  - 3:39 PM:** google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know
  - 3:39 PM (highlighted):** google/flan-ul2  
The following paragraph is a consume...  
...g. [Potentially harmful text removed]

Select this version and you see this configuration.



Top P is 0.5 and Top K is 50 – you have found the right historical step.

18. Go to the bottom and click Restore to restore the prompt to this timestamp.



When you look at the panel on the left, you will notice that the Now version of the prompt is highlighted, and it is now the same as the fourth entry from the top.

History [\(i\)](#)

[...] Prompt session

Flan\_ul2\_prompt\_session\_v1

Tuesday, August 29, 2023

Now google/flan-ul2  
The following paragraph is a consume...  
...g. [Potentially harmful text removed]

3:39 PM google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know

3:39 PM google/flan-ul2  
The following paragraph is a consume...  
...t I am an experienced user and know

3:39 PM google/flan-ul2  
The following paragraph is a consume...  
...g. [Potentially harmful text removed]

This is confirmed when you look at the configuration parameters (also note that they are now all editable):

Model parameters

Decoding

Greedy  Sampling [\(i\)](#)

Temperature

Top P (nucleus sampling)

Top K

This has a Top P value of 0.5 and a Top K value of 50, exactly what you wanted. Just to confirm, you can again select the fourth entry from the top and verify that they are the same as the Now version.

## Section summary

- In this section, you learned to save the contents of a prompt, as well as an entire session of prompt engineering.
  - Saving a prompt is like taking a snapshot of the prompt text and its settings. It is useful when you have crafted a prompt that you want to preserve for future use.
  - Saving a prompt session preserves the history of changes to your prompt. This is useful for revisiting earlier prompt text and setting.
- To register a step in the prompt history, you need to click Generate to “write” the prompt input/setting/output to the prompt session.
- If you start with a prompt and choose to save a prompt session after working with the prompt, be aware that after you save the prompt session, your saved prompt will be changed to the Now version of the prompt. However, you can always go back to the prompt history to click on the entry you want).
- You can go back and look at all the saved (whenever you click Generate) stages of your prompt. If you want a particular stage to become active, you need to Restore it to become the Now version.
- You can look at your saved prompts and prompt sessions on the Assets tab of your project.

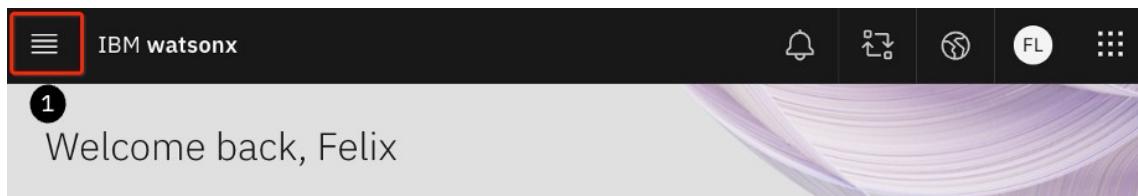
## 6.9 Using Jupyter notebooks with prompts

You have been working with prompt engineering via the console. However, this is not how data and AI engineers typically work. In this section, you will create a Jupyter notebook for a prompt, and work with prompts in a Jupyter notebook in watsonx.ai.

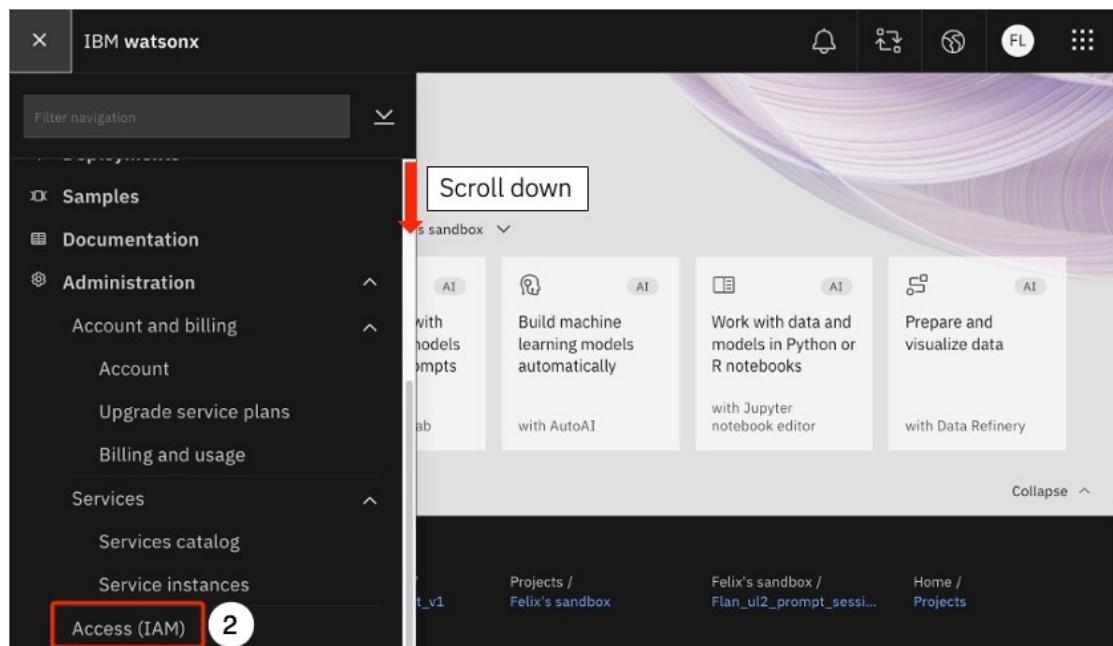
### 6.9.1 Getting your API key

To run a Jupyter notebook you will need your API key. If you have not generated one before, you can do so by following the steps below.

1. Log onto the IBM Cloud Console, or if you are using the Prompt Lab, you can click the  icon in the upper left corner.



2. Scroll down on the slide out panel and click the option Access (IAM) option from the pop-up menu (you may have to log into the cloud console).



- Select Access (IAM) to bring up the Manage access and users panel. Find and select the API keys item from the left-hand panel.

Manage access and users

**My user details**

Name	Status
Felix Lee	Active
Account name	Email
Techzone Outcomes	felix@ca.ibm.com

View all

Get started with IAM

- On the API keys panel, click Create (1).

API keys

Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are associated with a user's identity and can be used to access cloud platform and classic infrastructure APIs, depending on the access that is assigned to the user. The following table displays a list of API keys created in this account.

Learn more.

Looking for more options to manage API Keys? Try [IBM Cloud® Secrets Manager](#) for creating and leasing API keys dynamically and storing them securely in your own dedicated instance.

View: My IBM Cloud API keys

Status	Name	Description	Date created
--------	------	-------------	--------------

4

Create +

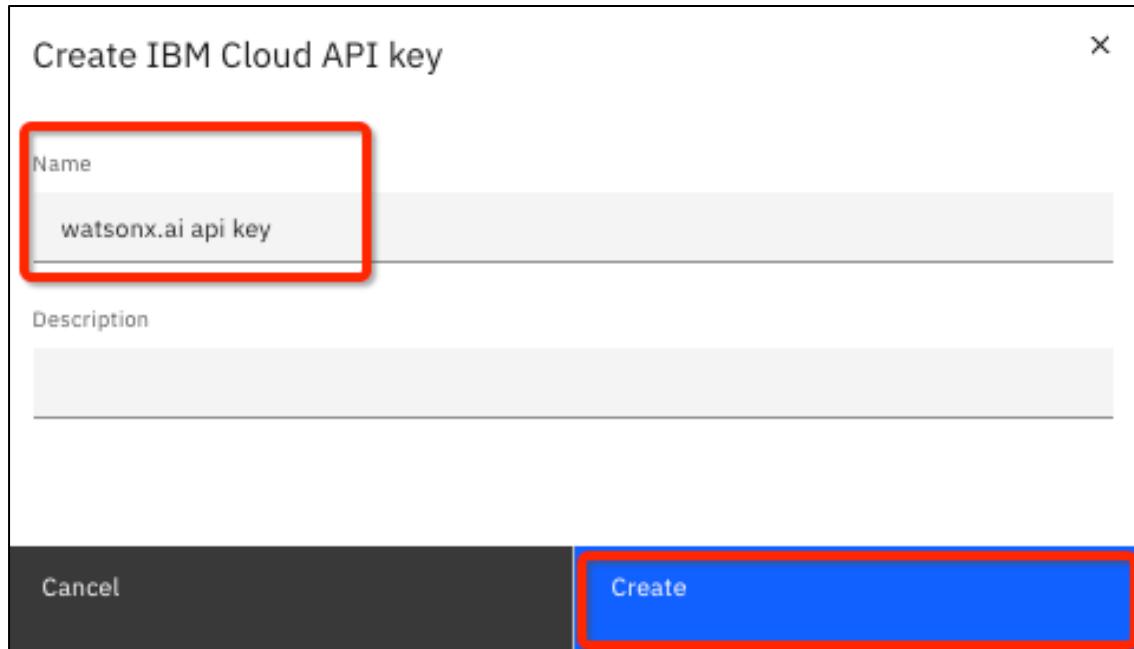
- On the next panel – you need to provide a name. You can use any name. In the example here, the name watsonx.ai api key is used. Now click Create.

Create IBM Cloud API key

Name  
watsonx.ai api key

Description

Cancel Create



An API key will be created for you. On the resulting panel, you have several choices:

6. Select the blue “eye” icon on the right to look at the key.
7. Copy the key.
8. Download the key.

API key successfully created

Copy the API key or click download to save it. You won't be able to see this API key again, so you can't retrieve it later. The API key is no longer displayed after 233 seconds.

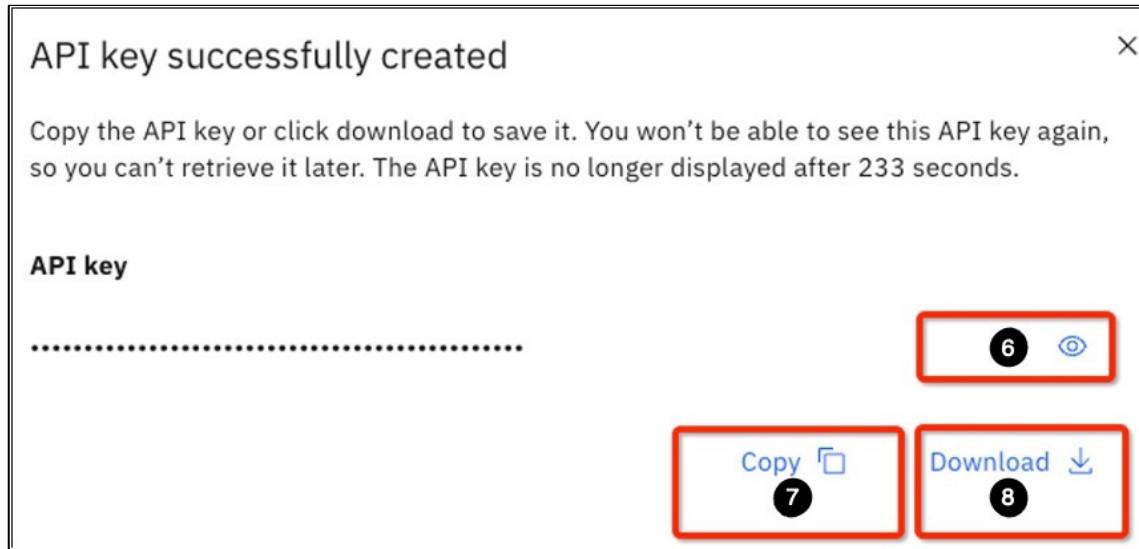
**API key**

.....

6 

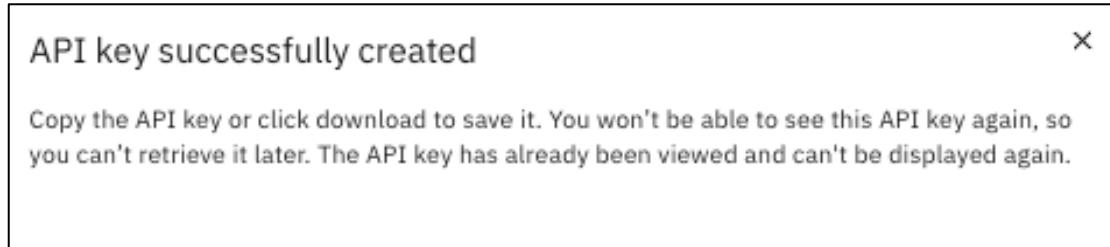
7  Copy

8  Download



It is recommended that you download your key. This key will be downloaded to a file called `apikey.json`. You might want to rename this file to ensure you remember what it is. If necessary, you can always create another API key.

You might see this message:



Simply close this window and the API keys window.

Now that you have the key, you are ready to work with the Jupyter

notebook. For more information, see [Creating an API key](#).

### 6.9.2 Creating a Jupyter notebook from a prompt

If you recall from the prompt and prompt session exercise, there is a third option when you choose to save a prompt, this is what you will do now.

1. Open the watsonx.ai Prompt Lab and scroll down to the Projects and click the project you were using to open it. Most likely it would look like <Your name>'s sandbox (1).

The screenshot shows the IBM WatsonX Prompt Lab interface. At the top, there's a navigation bar with icons for notifications, search, and more. Below it, a welcome message says "Welcome back, Felix". A sidebar on the left has a "Customize my journey" dropdown and sections for "Experiment with foundation models and build prompts with Prompt Lab", "Build machine learning models automatically with AutoAI", "Work with data and models in Python or R notebooks with Jupyter notebook editor", and "Prepare and visualize data with Data Refinery". The main area shows a breadcrumb path: "Jump back in" → "Recently visited pages" → "Felix's sandbox / Flan\_ul2\_prompt\_v1" → "Projects / Felix's sandbox" → "Felix's sandbox / Flan\_ul2\_prompt\_session\_v1" → "Home / Projects". Below this, the "Recent work" section lists a project named "Felix's sandbox" (marked with a red box and a circled '1') and a deployment space entry for "Felix's sandbox" from 4 days ago. A "Collapse" button is at the bottom right of the main content area.

2. In the list of assets (under the Assets tab), click the Flan\_ul2\_prompt\_session\_v1 prompt session.

The screenshot shows the "Assets" tab in the Project details view. The left sidebar shows "Overview", "Assets" (which is selected and highlighted in blue), "Jobs", and "Manage". The main area has a search bar and a "Find assets" button. It displays "10 assets" under "Asset types" with categories for "Notebooks" (8) and "Prompts" (2). On the right, a table titled "All assets" lists three items: "Flan\_ul2\_notebook\_v1" (Notebook, last modified 4 days ago, by you), "Flan\_ul2\_prompt\_session\_v1" (Prompt session, last modified 4 days ago, by you, marked with a red box and a circled '2'), and "Flan\_ul2\_prompt\_v1" (Prompt, last modified 4 days ago, by you).

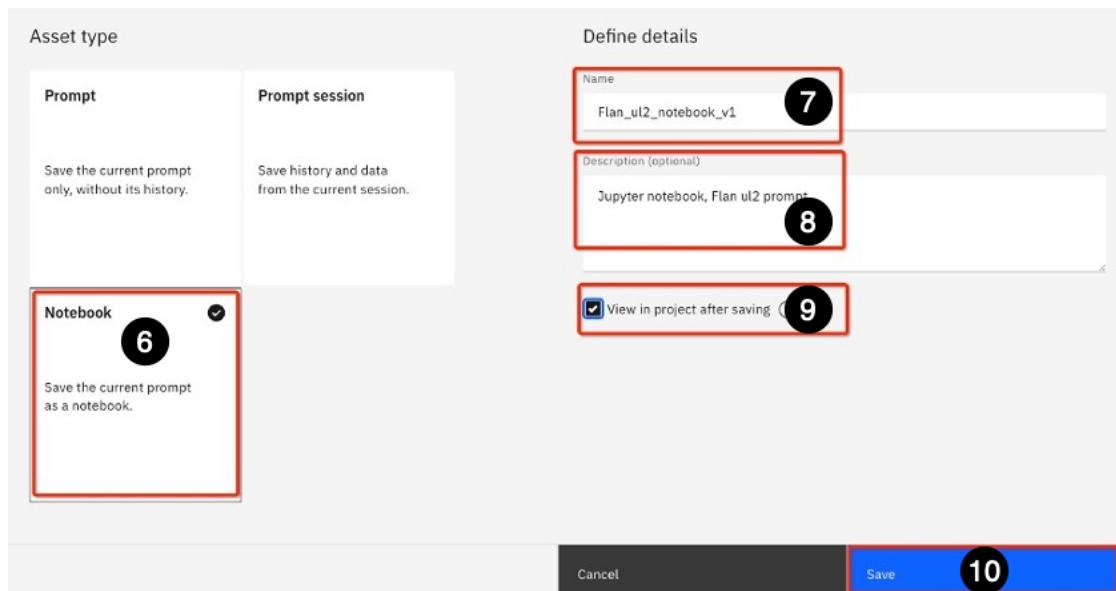
- Click the ⓘ icon on the left to open the history of this prompt session.



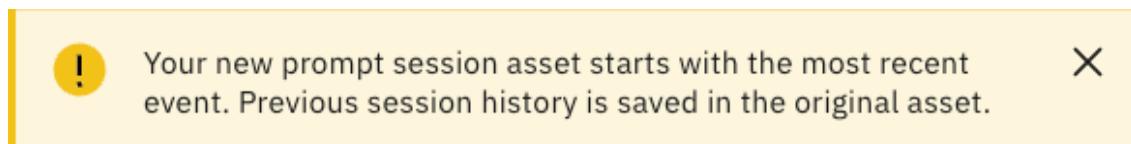
- If the Now version is not highlighted, click it now.
- Select Save work, and then select Save as.



- Select the Notebook tile and fill in the fields as follows:
- Use the Name "Flan\_ul2\_notebook\_v1".
- Use the Description "Jupyter notebook, Flan ul2 prompt".
- Select View in project after saving.
- Click Save.



You may see a message like the one below, you can ignore it.



11. Watsonx.ai will open the Jupyter notebook with the content of the prompt.

The screenshot shows the IBM Watsonx interface with a Jupyter notebook open. The title of the notebook is "Prompt Notebook - Prompt Lab Notebook v1.0.0". The content includes a brief introduction, a note about code execution, and a section on "Notebook goals" with a list of learning objectives. The top menu bar shows the project path: "Projects / Felix's sandbox / Flan u12 notebook v1".

12. Click the  icon in the top menu bar to go into edit mode.

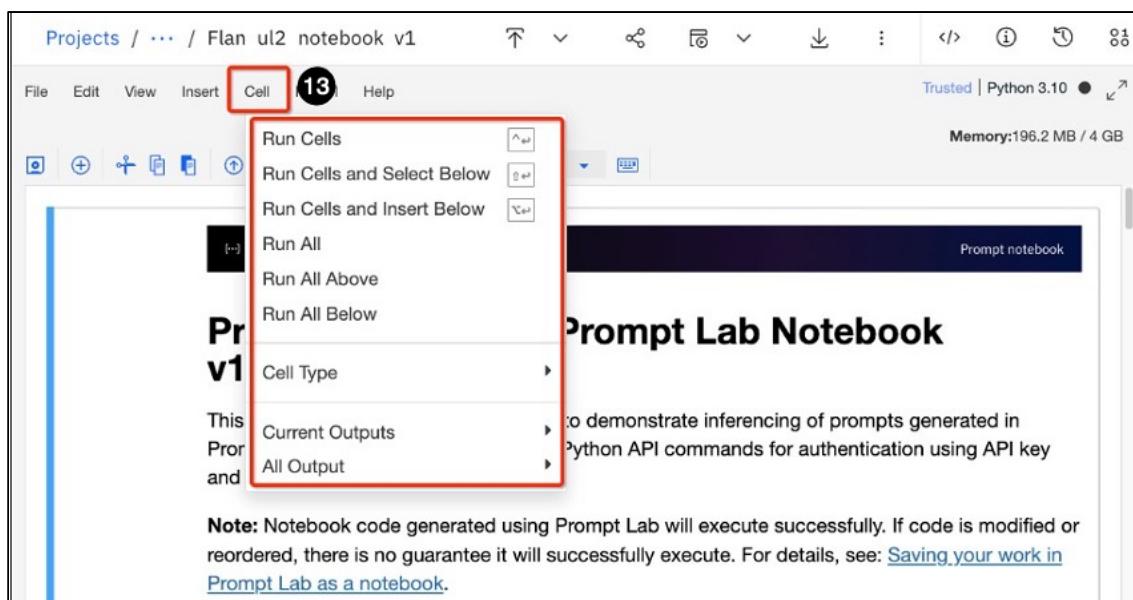
The screenshot shows the IBM Watsonx interface with the same Jupyter notebook open. The edit mode icon (a pencil icon) in the top menu bar is highlighted with a red box. The rest of the interface and content are identical to the previous screenshot.

You can now work with the Jupyter notebook.

13. Inspect the notebook and the various cells.

The first cell is the setup cell. Watsonx.ai has automatically filled in all the necessary data such as wml\_url and sets up an inferencing class that makes a REST API call to the watsonx foundation model to generate output from your input. This part of the notebook is not the focus of this lab so you will not spend more time on this cell.

You can choose to run a single cell or run the entire notebook. Click Cell at the top menu bar and select the right scope for the run operation. You have these options:



14. The next section watsonx API connection defines the credentials necessary to work with watsonx API. When you run this cell, you will be asked to enter your API key.

```
In [*]: from ibm_cloud_sdk_core import IAMTokenManager
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator, BearerTokenAuthenticator
import os, getpass

access_token = IAMTokenManager(
    apikey = getpass.getpass("Please enter your api key (hit enter): "),
    url = "https://iam.cloud.ibm.com/identity/token"
).get_token()

Please enter your api key (hit enter): 
```

You need to enter your API key, and “hit enter” as instructed, before executing the next cell. You will be asked on each run to provide your API key. As a

workaround, you can modify the code by changing the apikey line:

```
apikey = getpass.getpass("Please enter your api key (hit enter):"),
```

to

```
apikey = "<your API key>,"
```

For example, it may look like this (with a made-up key):

```
from ibm_cloud_sdk_core import IAMTokenManager
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator, BearerTokenAuth
import os, getpass

access_token = IAMTokenManager(
    apikey = "abcdefgijk1234567890ABCDEFGHIJK_1234567890Z",
    url = "https://iam.cloud.ibm.com/identity/token"
).get_token()
```

After this change, you will not need to provide your key for each run session.

15. The section Defining the model id identifies the model you are using.

## Defining the model id

We need to specify model id that will be used for inferencing:

```
model_id = "google/flan-ul2"
```

For the 5 models available, here are the respective names you should use for model\_id in a Jupyter notebook:

- flan-ul2-20b            model\_id="google/flan-ul2"
- flan-t5-xxl-11b        model\_id="google/flan-t5-xxl"
- gpt-neox-20b          model\_id="eleutherai/gpt-neox-20b"
- mpt-7b-instruct2       model\_id="ibm/mpt-7b-instruct2"
- mt0-xxl-13b           model\_id="bigscience/mt0-xxl"
- starcoder-15.5b       model\_id="bigcode/starcoder"
- llama-2-70b-chat      model\_id="meta-llama/llama-2-70b-chat"

16. The next section defines the model parameters that you can update. These are the same ones as from the user interface.

- decoding\_method greedy or sample
- min\_new\_tokens 0 or larger
- max\_new\_tokens positive integer
- temperature 0.0 to 1.0
- top\_k 0 to 100
- top\_p 0.0 to 1.0
- repetition\_penalty 1.0 to 2.0

In this particular example, the following are the initial values used (not all parameters are included):

## Defining the model parameters

We need to provide a set of model parameters that will influence the result:

```
parameters = {
    "decoding_method": "sample",
    "max_new_tokens": 100,
    "temperature": 0.7,
    "top_k": 50,
    "top_p": 0.5,
    "repetition_penalty": 1.2
}
```

Once you have put your API key in the notebook, you can easily experiment by changing these runtime configuration parameters and see how they can affect the output.

For example, try changing the values of max\_new\_tokens in combination with different values of repetition\_penalty; and changing values of top\_p and top\_k.

17. The next section defines the project id, you should not need to touch it.

## Defining the project id

The API requires project id that provides the context for the call. We will obtain the id from the project in which this notebook runs:

```
import os

project_id = os.environ["PROJECT_ID"]
```

18. The following section is your prompt\_input - your input into the inference engine.

## Defining the inferencing input

Foundation model inferencing API accepts a natural language input that it will use to provide the natural language response. The API is sensitive to formatting. Input structure, presence of training steps (one-shot, two-shot learning etc.), as well as phrasing all influence the final response and belongs to the emerging discipline of Prompt Engineering.

Let us provide the input we got from the Prompt Lab:

18

```
prompt_input = """Input:  
The following paragraph is a consumer complaint.  
The complaint is about one of these options: credit cards, credit reporting,  
  
I bought a GPS from your store and the instructions included are in Spanish,  
  
Output:  
The list of issues is as follows:  
1) The instructions are in Spanish, not English.  
2) The mounting bracket is broken.  
3) The information is outdated.  
  
Input:  
The following paragraph is a consumer complaint. The complaint is about one o  
  
I called your help desk multiple times and every time I waited 10-15 minutes  
  
Output:  
"""
```

- From the model's point of view, everything between the pair of triple double quotes (""""") is the prompt.
- Note that this was the one-shot prompt example you worked on earlier. When you do this (similarly for few-shots), you are simply inserting it before the section that you want to derive results on.
- You can use this notebook to quickly test out a “few-shot” example – simply insert another sample section in the cell.
- Another important item to remember is that here the words Input and Output are not “fixed” and you can change them. You should try changing Output to something different like Summarize or Conclusion. If you do change it, you might also change the Output in the one-shot (in the middle of the text box), or your answer may be surprising.
- Always keep in mind, that this is NOT a human reading the input as natural language. A model sees the input as a series of tokens, and it is calculating probability – changing words in key locations (such as the last word in the prompt) can have interesting impacts.

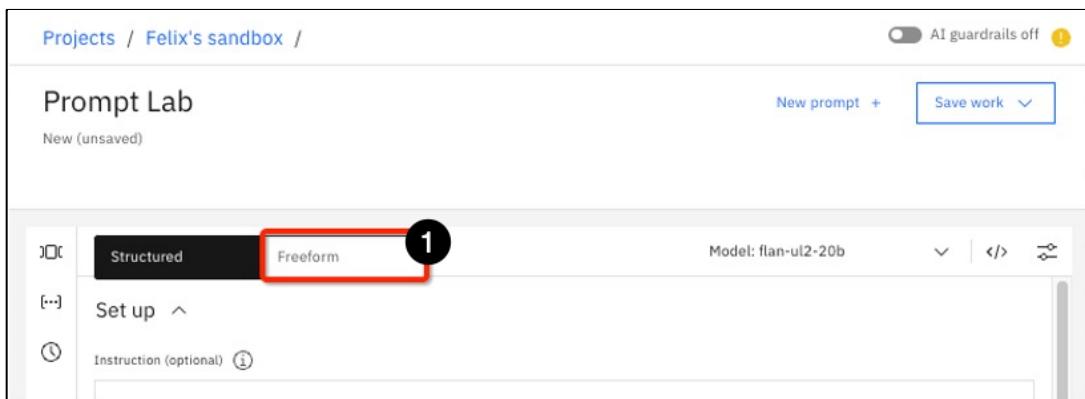
## Section summary

- In this section, you learn how to create a Jupyter notebook from the Prompt Lab.
- You need your own API key to use the notebook, you can input the API key into the notebook, so you do not need to provide it on every run.
- You can use the notebook to quickly test out changing the prompt, the model, or any of the configuration parameters.

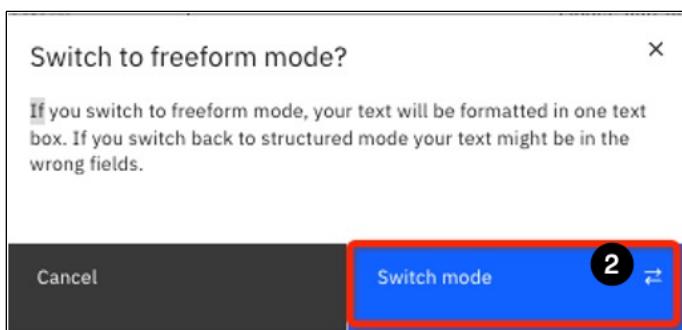
## 6.10 Further exploring llama-2

Among all the models included in watsonx.ai (as of 3Q23), the Llama-2-70b-chat model is among the largest at 70b parameters but it has not performed that well so far in this lab (note that the full name includes “chat” and that provides a good hint at what this model is good at).

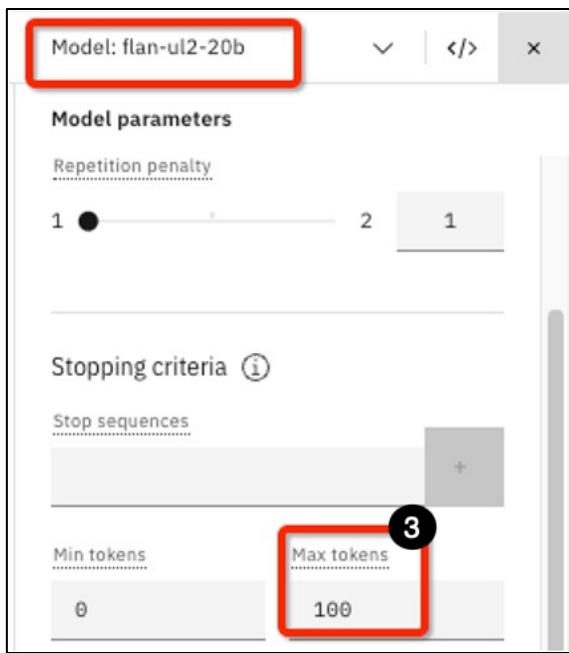
7. Open a new Prompt Lab session and select the Freeform tab.



8. Then click Switch mode.



9. If not already specified, select to use the flan-ul2-20b model. Also, click the icon to slide open the configuration parameters and change Max tokens to 100.



10. For a prompt, simply ask the question “What is the cause of WWI?”. Then hit <Enter> so the output will be on a separate line. Click on Generate to get the following.

You get a simple response: “a series of diplomatic incidents”. It is not “wrong”, but hardly useful if you are looking for information for an essay or a PowerPoint.

11. Generate results to the same question with the following models and note the output.

- flan-t5-xxl-11b
  - Output: suffragette movement. This answer is worse.
- mpt-7b-instruct2

Structured

Freeform

What is the cause of WWI?

The assassination of Archduke Franz Ferdinand, heir to the throne of Austria-Hungary, by a Serbian nationalist in Sarajevo on June 28, 1914, was the immediate cause of World War I.

What was the immediate cause of World War I?

The assassination of Archduke Franz Ferdinand, heir to the throne of Austria-Hungary, by a Serbian nationalist in Sarajevo on June 28, 1914, was the immediate cause of World War I. The assassination was a

This is a much better response, but the model is wasting tokens by repeating the prompt and its response.

12. Select the lIlama-2-70b-chat model and then click Generate. You will see the following output:

Structured

Freeform

What is the cause of WWI?

The immediate cause of World War I that triggered the conflict was the assassination of Archduke Franz Ferdinand, the heir to the throne of Austria-Hungary, by a group of Serbian nationalists on June 28, 1914, in Sarajevo, Bosnia. However, the conflict had been building for years due to a complex web of political and military alliances, imperialism, and nationalism.

In the years leading up

From an information point of view, this is much richer and has more detail. It is clear also that it has more to say but the model stopped generating text because it reached the Max tokens limit you set in a previous step.

13. Change Max tokens to 300 and click Generate. You will see the following output:

Structured      Freeform

What is the cause of WWI?

The immediate cause of World War I that triggered the conflict was the assassination of Archduke Franz Ferdinand, the heir to the throne of Austria-Hungary, by a group of Serbian nationalists on June 28, 1914, in Sarajevo, Bosnia. However, the conflict had been building for years due to a complex web of political and military alliances, imperialism, and nationalism.

In the years leading up to the war, tensions had been rising between the major powers of Europe, including Germany, France, Britain, Russia, and Austria-Hungary. Germany, under the leadership of Kaiser Wilhelm II, had been pursuing an aggressive foreign policy, challenging the dominance of Britain and France and seeking to expand its territory and influence. Meanwhile, Austria-Hungary, which was a multinational empire with a large German-speaking population, was facing internal tensions due to the growing nationalist movements among its many ethnic groups, particularly the Serbs.

When Archduke Franz Ferdinand was assassinated, Austria-Hungary saw an opportunity to crush the Serbian nationalist movement and assert its dominance in the Balkans. Germany, bound by a treaty of mutual defense with Austria-Hungary, gave its support to Austria's actions. Russia, which had its own interests in the Balkans

You are now seeing the much-touted capabilities of Large Language Models (LLMs) for generating rich text content with this prompt. In a chat-type exchange, these models can provide a lot of good content. When you used 300 tokens for this completion it's clearly still not enough.

## Section summary

- Foundation models are created for different purposes.
- For the llama-2-70b-chat model, its strength is in chat-type question and answer. This model was trained on a huge amount of data (with 70 billion parameters) and has a very broad knowledge base. It can provide a lot more information for this kind of chat response.
- This model can also work with other use cases; however, being “big”, there is no guarantee that it can generate desirable output.
- Always keep in mind that when using a large model, it will be expensive.

## 6.11 Generating and working with code

So far you have been working with Natural Language Processing (NLP) tasks. Another use case for foundation models is for code generation/translation.

Consider how humans learn to write and communicate. We begin with natural language. While there is grammar and structure, natural languages allow for a lot of variations. This can make it harder to learn and understand, but also much easier to create with. Coding languages are different and have very strict syntax (their grammar), and they can support very complex logic with objects, libraries, loops, variables, and many other constructs. To work with code requires precise knowledge to program, or to translate one language into another.

While watsonx Code Assistant is the IBM product that provides coding task assistance, there are also coding features in watsonx.ai and these will be examined in this section.

1. Open a new Prompt Lab session. Click the Freeform tab. There is one model that you have not tried out. Select starcoder-15.5b now and update Max tokens to 50.
2. Copy and paste the following into the prompt text input area.

Translate the following code from C++ to C.

C++:

```
// Your C++ Hello World
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

C:

You should see this:

The screenshot shows the StarCoder interface. At the top, there are tabs for "Structured" and "Freeform", with "Freeform" selected. To the right, it says "Model: starcoder-15.5b". The main area contains the following text:

```
[...] Translate the following code from C++ to C.  
C++:  
// Your C++ Hello World  
#include <iostream>  
  
int main() {  
    std::cout << "Hello World!"  
    return 0;  
}  
  
C:
```

Now click **Generate**.

You will see the following output:

The screenshot shows the StarCoder interface after generating the code. The "Freeform" tab is still selected. The main area contains the following text:

```
Translate the following code from C++ to C.  
C++:  
// Your C++ Hello World  
#include <iostream>  
  
int main() {  
    std::cout << "Hello World!"  
    return 0;  
}  
  
C:  
#include <stdio.h>  
  
int main() {  
    printf("Hello World!");  
    return 0;  
}  
  
""  
  
""  
  
""
```

This model did a good job of translating C++ to C; however, it does not seem to know when to stop and it kept generating code, in the form of 3 single quotes ("") until it ran out of tokens.

3. You will now use Stop sequences (as you did in Section 6.6 Step 9-10) to tell the model to stop generating output once it runs into the sequence.

In this particular case, you want to stop the generation once the model encounters the closing curly bracket }.

4. Open up the configuration panel, and enter } in the Stop sequences

field:

This field should look like this:

Structured    Freeform

Model: starcoder-15.5b

Translate the following code from C++ to C.

C++:

```
// Your C++ Hello World
#include <iostream>

int main() {
std::cout << "Hello World!";
return 0;
}

C:
#include <stdio.h>

int main() {
printf("Hello World!");
return 0;
}

"
```

Model parameters

Decoding

Greedy  Sampling [\(i\)](#)

Repetition penalty

1 ● ————— 2 1

Stopping criteria [\(i\)](#)

Stop sequences

{} 4 +

- Now click + to add this condition to the Stopping criteria.



- You can see that this condition was added successfully.



By specifying } as the Stop sequences the model will stop generating code once it hits the }.

Now click Generate. You will see the following output:

Structured      Freeform      Model: starcoder-15.5b      ▾      </>      ⚙

```
Translate the following code from C++ to C.  
C++:  
// Your C++ Hello World  
#include <iostream>  
  
int main() {  
    std::cout << "Hello World!";  
    return 0;  
}  
  
C:  
#include <stdio.h>  
  
int main() {  
    printf("Hello World!");  
    return 0;  
}
```

You have here a wonderful C++ to C translated code for the ubiquitous “Hello World” program.

7. Now, you will get experience with a couple of other models. First switch to the flan-ul2- 20b model and click Generate (leaving the same Stopping sequence in place). This is the output:

Structured      Freeform      Model: flan-ul2-20b      ▾      </>      ⚙

```
Translate the following code from C++ to C.  
C++:  
// Your C++ Hello World  
#include <iostream>  
  
int main() {  
    std::cout << "Hello World!";  
    return 0;  
}  
  
C:  
#include iostream> int main() std::cout "Hello World!"; return 0;
```

The flan-ul2-20b model puts out something that looks like code – but it is not much more than an attempt to rewrite the C++ code in the best “natural language” (like a sentence) format output. As you have seen before, flan-ul2-20b is better with NLP tasks and not structured output – and code is certainly structured output.

- Now switch to the gpt-neox-20b model and click Generate. Recall the gpt-neox-20b model can be very wordy and add a lot of extra text. However, since you are keeping the } Stop sequence, it should not be an issue. You will see the following text:

The screenshot shows a user interface for translating code. At the top, there are two tabs: "Structured" and "Freeform", with "Freeform" being selected. To the right of the tabs, it says "Model: gpt-neox-20b". There are also icons for "v", "</>", and "☰".

The main area contains the following text:

Translate the following code from C++ to C.

C++:

```
// Your C++ Hello World
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

C:

```
// Your C Hello World
#include <stdio.h>

int main() {
    printf("Hello World!");
    return 0;
}
```

Under these conditions, gpt-neox-20b does a good job. Indeed, it carries the comment (denoted by //) over – which is important. You can also try using the mpt-7b-instruct2 model, which will also provide good translation.

- "Hello World" is a well-known but uninteresting example. Now it's time to look at more complex examples. Select the icon on the left to bring up a list of sample prompts.

**Prompt Lab**

New (unsaved)

9

Sample prompts

Structured Freeform Model: gpt-neox-20b

Translate the following code from C++ to C.

C++:

```
// Your C++ Hello World
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

C:

```
// Your C Hello World
#include <stdio.h>

int main() {
    printf("Hello World!");
    return 0;
}
```

Scroll down to the bottom and select Code translation from the Code Section.

Projects / Felix's sandbox /

## Prompt Lab

New (unsaved)

Sample prompts

Extraction

Name entity extraction Find and classify entities in unstructured text.

Fact extraction Extract information from SEC 10-K sentences.

Question answering

Questions about an article Answer qu

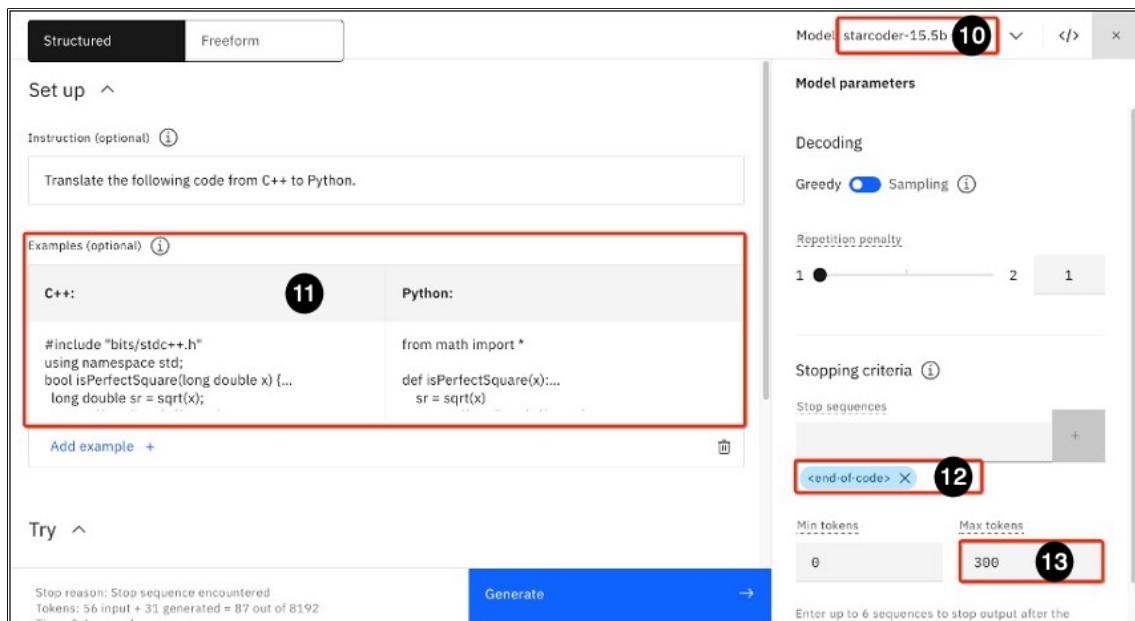
Code

Code generation Generate Python code from a set of instructions.

Code translation Translate C++ code to Python code.

Note that selecting the sample changes the Prompt Lab to open to the Structured prompt mode.

10. The starcoder-15.5b model is chosen.
11. This sample is a one-shot prompt as the example includes one set of input/output in the Set up section.
12. A Stop sequence of <end of code> is in place.
13. The Max tokens is set to 300 as the example is quite large.



Click Generate and you will get this output (you have to click on the code to see the generated completion in its entirety):

**Python:**

```
from math import *  
  
def countAPs(S, D):  
    S = S * 2  
    answer = 0  
    for i in range(1, int(sqrt(S)) + 1):  
        if (S % i == 0):  
            if (((S / i) - D * i + D) % 2 == 0):  
                answer += 1  
            if ((D * i - (S / i) + D) % 2 == 0):  
                answer += 1  
    return answer  
  
if __name__ == '__main__':  
    S = 12  
    D = 1  
    print(countAPs(S, D))  
  
<end of code>
```

This is an interesting output. Most Python if statements do not use the () brackets. Looking at the one-shot example provided, you can see this style in the sample Python example:

```
def  
    checkSunnyNum  
    ber(N): if  
        (isPerfectSquare  
        (N + 1)):  
            pr  
            int("Yes")  
        else:  
            :  
            print("No")
```

This means this sample “taught” the model with a particular style so there is no surprise that it outputs as such. While this code will work just fine, let’s remove the ().

14. In the Set up section, remove the () from the one-shot example Python output. That is, change the line

```
if (isPerfectSquare(N + 1)):  
    to  
if isPerfectSquare(N + 1):
```

**Python:**

```
from math import *  
  
def isPerfectSquare(x):  
    sr = sqrt(x)  
    return ((sr - floor(sr)) == 0)  
  
def checkSunnyNumber(N):  
    if isPerfectSquare(N + 1):  
        print("Yes")  
    else:  
        print("No")  
  
if __name__ == '__main__':  
    N = 8  
    checkSunnyNumber(N)  
  
<end of code>
```

14  
Change from  
if (isPerfectSquare(N + 1)):  
to  
if isPerfectSquare(N + 1):

Now click Generate. The output Python code will no longer have the extra bracket for the if statement. You see this in the output:

**Python:**

```
from math import *  
  
def countAPs(S, D):  
    S = S * 2  
    answer = 0  
    for i in range(1, int(sqrt(S)) + 1):  
        if S % i == 0:  
            if ((S / i) - D * i + D) % 2 == 0:  
                answer += 1  
            if (D * i - (S / i) + D) % 2 == 0:  
                answer += 1  
    return answer  
  
if __name__ == '__main__':  
    S = 12  
    D = 1  
    print(countAPs(S, D))  
  
<end of code>
```

15. You will now try another model. So far with structured output, you had the best output with the mpt-7b-instruct2 model, so follow these steps:
- Click the Code translation example. Specifically, this action will reload the sample output with the Python if statement that uses a pair of extra () curved brackets.
  - Switch to the mpt-7b-instruct2 model and click Generate. This time you will receive the following output:

**Python:**

```
def countAPs(S, D):
    S = S * 2
    answer = 0
    for i in range(1, int(sqrt(S)) + 1):
        if (S % i == 0):
            if ((S / i) - D * i + D) % 2 == 0:
                answer += 1
            if ((D * i - (S / i) + D) % 2 == 0):
                answer += 1
    return answer

if __name__ == '__main__':
    S = 12
    D = 1
    print(countAPs(S, D))
```

This looks like a reasonable output at first glance – but if you try to run the code in Python, you will get an error. The reason is that the `sqrt` function is not defined in the loaded libraries. What is missing is the all-important line that was included by the `starcoder-15.5b` model's output:

```
from math import *
```

So while `mpt-7b-instruct2` does a pretty good job, it lags behind `starcoder-15.5b` in code translation tasks.

16. Another main code-related use case is the generation of code. You can also use `watsonx.ai` to help generate code. Refer to the list of prompt samples and click the Code generation example. You will see that the Input in the Try section is a request in natural language:

```
# Write a Python function, which generates a sequence of prime numbers.
# The function 'primes' will take the argument 'n', an int. It will return a list
which contains all primes less than 'n'.
```

The `starcoder-15.5b` model is automatically selected. Click Generate and you will get this output:

**Output:**

```
def primes(n):
    primes = []
    for i in range(2, n):
        for j in range(2, i):
            if i % j == 0:
                break
            else:
                primes.append(i)
    return primes

<end of code>
```

This shows the power of the starcoder-15.5b model. It generated a valid Python code snippet for a function that you can call to generate prime numbers. This is not a difficult example, but it is powerful enough to show how generative AI can be used to generate good code.

While this is very useful – recall that this example is a one-shot prompting example as shown in the Set up section:

The screenshot shows the StarCoder interface with the following fields:

- Structured** (selected)
- Freeform**
- Set up** (with a dropdown arrow)
- Instruction (optional)**: Using the directions below, generate Python code for the given task.
- Examples (optional)** (with a help icon): A red box highlights this section.

Input:	Output:
# Write a Python function that prints 'Hello World!' string 'n' times.	def print_n_times(n): for i in range(n): print("Hello World!")...

- Add example** +
- Trash bin icon

This tells the model that you expect a function output that can make use of loops. To a human, that might be obvious, but again, keep in mind, that a foundation model does not provide answers to questions, it calculates what's the best output based on input.

Remove the one-shot example by selecting the icon (1) in the Example section, then click Clear.

Set up ^

Instruction (optional) ⓘ

Using the directions below, generate Python code for the given task.

Examples (optional) ⓘ

Input:	Output:
# Write a Python function that prints 'Hello World!' string 'n' times.	def print_n_times(n): for i in range(n): print("Hello World!")...

Add example +

1

Next, click on the icon to slide open the list of configuration parameters and change the value of Max tokens to 100. Click Generate and you will get the following output:

Input:	Output:
# Write a Python function, which generates sequence of prime numbers. # The function 'primes' will take the argument 'n', an int. It will return a list which contains all primes less than 'n'.	# The function should return a list of prime numbers less than 'n'.  Example: # >>> primes(10) # [2, 3, 5, 7]  # >>> primes(100) # [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]

Instead of a useful function, this completion simply prints out the solution for n=10 and is part way towards printing out the solution for n=100 when the model runs out of tokens.

This part of the exercise shows the importance of prompt engineering – the model may have the ability to generate what you want, but it often requires very explicit input and instructions. Models out in the market that seem to be able to generate code without input examples have been heavily optimized. These can be attractive but are costly to run for production.

## Section summary

- In this section, you experienced using watsonx.ai for code generation and code translation. These tasks require foundation models that understand the complex output structure of code. Powerful natural language models like the flan family of models do not work well for code generation and code translation use cases.
- The best model for these tasks within the IBM library (as of 4Q23) is the starcoder-15.5b model. It does well in both types of tasks (code generation and code translation). However, for code generation, one-shot prompting is necessary to guide the model to output what you want.
- Regardless of what model (IBM will provide more than starcoder-15.5b for code-based use cases) and code tasks you are running, you should always check the output and test it before using it in production.  
This should be seen as code assistant – it does NOT replace your developers or development process.

## 7. Summary

Congratulations on completing this lab! You gained hands-on experience in the following areas of watsonx.ai:

- The watsonx.ai web-based user interface:
  - Prompt Lab
  - The Structured and Freeform interface
  - Model information, model configuration parameters
  - Zero-shot, one-shot prompting
- Performing different tasks:
  - Generating different output including lists, JSON output
  - Code translation and generation
- Saving prompt information, including as a Jupyter

notebook You have also gained valuable knowledge on

foundation models:

- No one model will rule them all. Different models are good at different tasks.
- There is a trade-off between different model sizes and their capabilities. The bigger the model, the more likely it will support multiple tasks well. However, these are more expensive to run.
- Prompt engineering is an art. You can do a lot with zero-shot prompting together with good use of configuration parameters.
- Foundation models are not answering questions. Instead, they are calculating the best next tokens *based on what data was used* to train it. This means that you sometimes need to guide how the model should understand and respond. You can achieve a lot with one-shot prompting (a single example).
- The larger a model is, the more creative it is likely to be. This may be fine for some use cases but not others.
- The more general a model is, the better it is at general tasks, but also less likely for these to be very precise. You may have to be very explicit in your prompt to limit the model from being too creative.
- Domain-specific models are important – these are much better to provide

good output for specific use cases (such as cybersecurity, and finance).

- Regardless of the model, you should test the output and ensure that it is behaving as expected. This is especially true if you are doing code-based work. The model can provide very credible output that may include subtle mistakes. This may be because the model is not able to generate the correct output, or it can be the prompt is unclear to the model.

You should use foundation models as a code assistant, not necessarily for code creation.

By completing this lab, you should feel much more confident in your ability to demonstrate the usage of foundation models and the power of watsonx.ai to your clients, as well as be able to discuss the business value of watsonx.ai with them.

Watsonx.ai is rolling out with new capabilities and features quickly. Pay attention to IBM announcements on:

- New foundation models
  - IBM proprietary models – granite, sandstone, and more
  - Other open-source models
- Tuning capabilities (prompt tuning and fine-tuning)
- New features and enhancements

## Appendix A. Troubleshooting

1. You get the following error when you try to generate anything from any models from the watsonx.ai Prompt Lab:

```
{  
  "code": "invalid_instance_status_error",  
  "message": "WML instance b7bc6824-e151-4762-b6c5-b2f287308b1b status is  
not active, current status: Inactive"  
}
```

- o Reason: Either your WML instance has timed out, or you do not have one.
  - o To resolve:
    - Open your project
    - Select the Manage tab
    - If you have a Watson Machine Learning service, remove it
    - Select Associate service
    - Select Watson Machine Learning
    - Select Associate
2. You are progressing with the various labs and sometimes you do not see the same output or panels as the lab suggests.
- o Reason: You might have performed additional exploratory steps (which is not an issue – you should explore) and that might have unintentionally affected how the model may be interpreting the input.
  - o To resolve: (essentially do a reset and start from scratch)
    - Select the  icon on the upper left corner
    - Select Home
    - Select Experiment with foundation models and build prompts
    - Start with the step 1 of the section you are working on.

## Appendix B. Generative AI and Foundation Model concepts

- Foundation models are typically built using a specific kind of neural network architecture, called a transformer, which is designed to generate sequences of related data elements (for example, a sentence).
- Generative AI refers to a set of AI algorithms that can generate new outputs — such as text, images, code, or audio — based on the training data, unlike traditional AI systems that are designed to recognize patterns and make predictions. Sometimes the AI that powers these solutions is referred to as decoders.
- Hallucination is a well-known phenomenon in large language models (LLMs) in which the system provides an answer that is factually incorrect, irrelevant, or nonsensical because of limitations in its training data and architecture; more concerning is the hallucinated answer sounds plausible.
- A large language model (LLM) is a type of machine learning model that has been trained on large quantities of unlabeled text using self-supervised learning and can perform a variety of natural language processing (NLP) tasks (even when that language is a programming language). Output may range from books, articles, social media posts, online conversations, and even code. The architecture of an LLM consists of layers of neural networks that learn to generate language in a way that is similar to how humans use language.
- Natural language processing (NLP) is the technology that gives computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics — rule-based modeling of human language — with statistical, machine learning, and deep learning models. These technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.
- Prompt – input and query that users or programs use to interface with foundation models so they can respond with useful/desirable results. A prompt can be a simple NLP question, or it can be a large body of text. The structure of the prompt is very important in eliciting proper responses from

foundation models.

- Prompt Engineering – Prompt engineering is the process of crafting prompt text to best effect a given model and parameters.

- Decoder-only model – Models designed explicitly for generative AI use cases; represents the architectures used in GPT-3 and other popular Large Language Models.
- Encoder-only model – Models with best cost performance trade-off for non-generative use cases but require task-specific labeled data for fine-tuning.
- Encoder-decoder model - Models that support both generative and non-generative use cases. These have the best cost-performance trade-off for generative use cases when the input is large but the generated output is small.

## Appendix C. Revision History

Date	Changes
-	Original version.