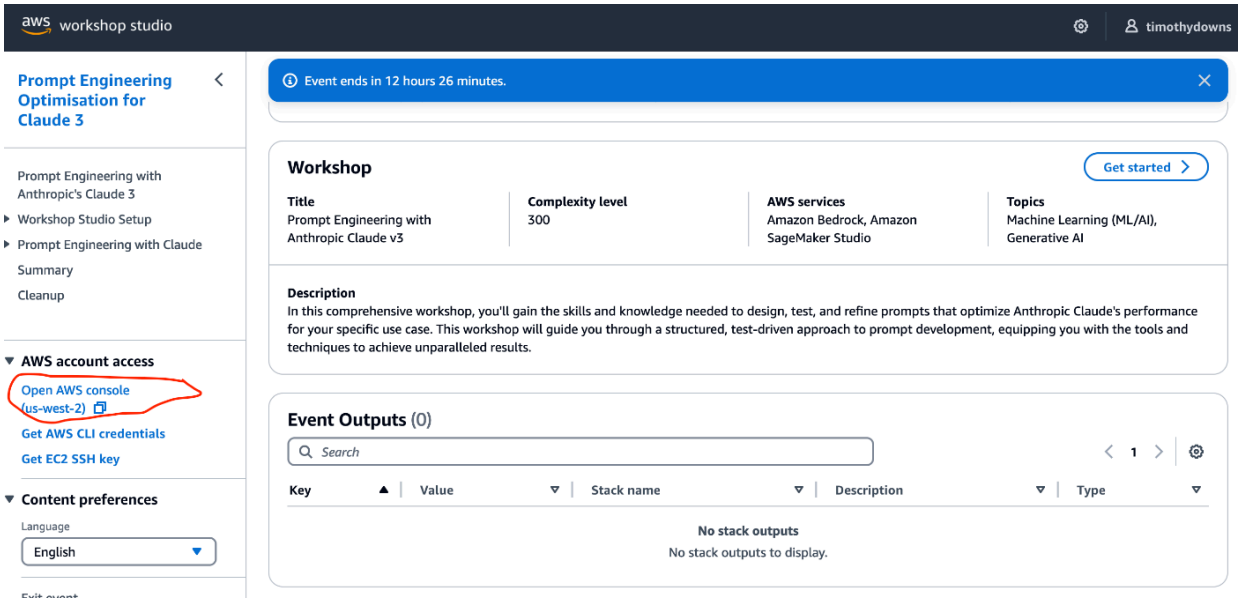
Instructions for prompt engineering workshop

Join workshop at https://catalog.us-east-1.prod.workshops.aws/join?access-code=49d5-0eb96a-08

Once logged in, click “Open AWS Console:” on the left side of the screen.

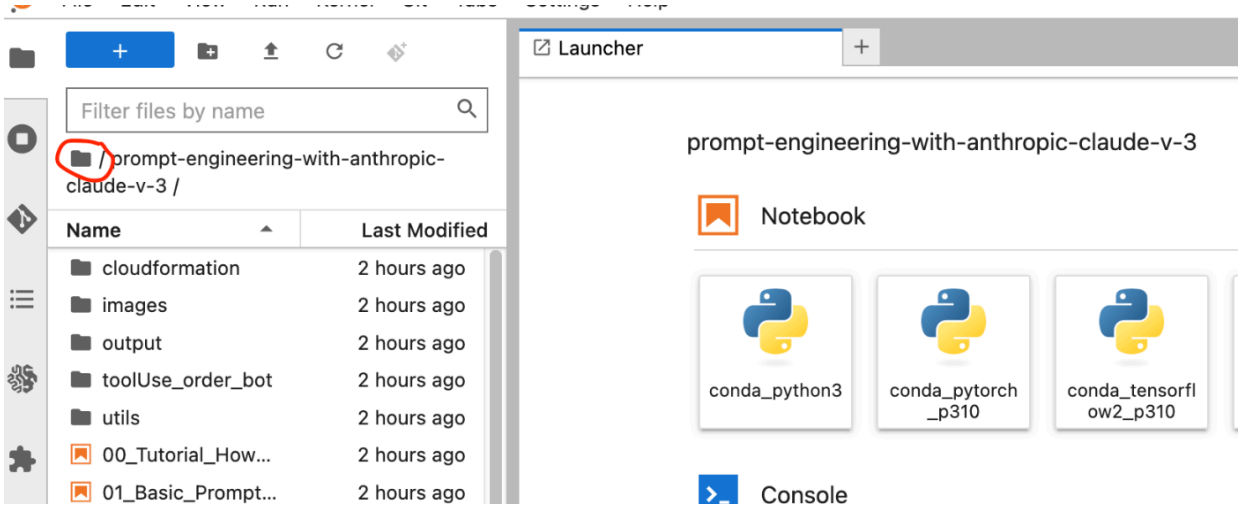


Go to “https://us-west-2.console.aws.amazon.com/bedrock/home?region=us-west-2#/modelaccess”, Edit Model Access - Enable specific model “Claude 3 Haiku”

Go to https://us-west-2.console.aws.amazon.com/sagemaker/home?region=us-west-2#/notebooks-and-git-repos and click “Open JupyterLab“, then open a new terminal

In the terminal, type MYALIAS="CHANGE_THIS_TO_YOUR_ALIAS" ; cp /home/ec2-user/SageMaker/prompt-engineering-with-anthropic-claude-v-3 /home/ec2-user/SageMaker/$MYALIAS -r

Then, go to the top level directory by clicking on the icon on the left side of the screen



Then open the directory named the alias that you chose in the step above, and then open 00_Tutorial_How ...

Choose the default Kernel - “conda_python3”

Then, run thru the notebooks, running each cell one by one - reading the context to understand what is happening

NB: We left a syntax error in a cell in the second notebook file 01_Basic_Prompt...


```python
def get_completion(prompt, system_prompt=None):
    # Define the inference configuration
    inference_config = {
        "temperature": 0.0,  # Set the temperature for generating diverse responses
        "maxTokens": 200  # Set the maximum number of tokens to generate
        "topP": 1,  # Set the top_p value for nucleus sampling
    }
    # Create the converse method parameters
    converse_api_params = {
        "modelId": modelId,  # Specify the model ID to use
        "messages": [{"role": "user", "content": [{"text": prompt}]}],  # Provide the user's prompt
        "inferenceConfig": inference_config,  # Pass the inference configuration
    }
    # Check if system_text is provided
    if system_prompt:
        # If system_text is provided, add the system parameter to the converse_params dictionary
        converse_api_params["system"] = [{"text": system_prompt}]

    # Send a request to the Bedrock client to generate a response
    try:
        response = bedrock_client.converse(**converse_api_params)

        # Extract the generated text content from the response
        text_content = response['output']['message']['content'][0]['text']

        # Return the generated text content
        return text_content

    except ClientError as err:
        message = err.response['Error']['Message']
        print(f"A client error occured: {message}")
```