Enabling Business User Interaction with IBM OpenPages through AI Agent

Automating compliance management in OpenPages with an AI agent

Krishna Damarla, Aug 9, 2025

Summary

This presentation demonstrates how to seamlessly integrate AI agents into existing enterprise software platforms to drive automation and enhance user support.

It details the architecture and step-by-step methodology for building, deploying and embedding AI agents within IBM OpenPages-a leading Governance, Risk, and Compliance (GRC) solution.

AI agents built within IBM Watsonx Orchestrate are designed to help business users in regulated industries to efficiently manage compliance requests (BPM), streamline business process workflows and improve operational efficiency.

Use Case: Automating Compliance Tasks

The following is an example use case for enabling business users to interact with OpenPages via an AI agent to automate the resolution of compliance issues

Compliance Issue	Business User Action	Agent Action in OpenPages UI
New compliance issues reported	Assign all new compliance issues to Agent	Create new cases in OpenPages
Compliance issue investigation complete	Approving all compliance issues that have been investigated and have a low-risk rating	Update case status in OpenPages
Compliance issue remediation required	Taking action to mitigate all high-risk compliance issues	Create new action task in OpenPages

Steps Involved

1: Define the Requirements

- Identify the specific compliance issues that the business user needs to take action on

- Determine the actions that the business user needs to perform (e.g. assign, approve, reject)

- Define the rules and workflows that govern the compliance issue lifecycle

2: Configure the IBM OpenPages Environment

- Set up the OpenPages environment to allow for agent-based interaction

- Configure the necessary workflows, rules, and permissions to enable business user interaction

- Ensure that the OpenPages environment is properly integrated with other relevant systems

3: Develop the Agent Chatbot Interface

- Develop Watsonx AI agent & Chatbot interface that allows business users to interact with OpenPages

- Ensure that the interface provides clear guidance on the actions that can be taken

4: Implement Workflow Automation

- Implement workflow automation to streamline the compliance issue lifecycle

- Use BPM tools to define and execute workflows

- Ensure that workflows are properly integrated with the OpenPages environment

Architecture Overview: AI Agent Integration with OpenPages

Architecture
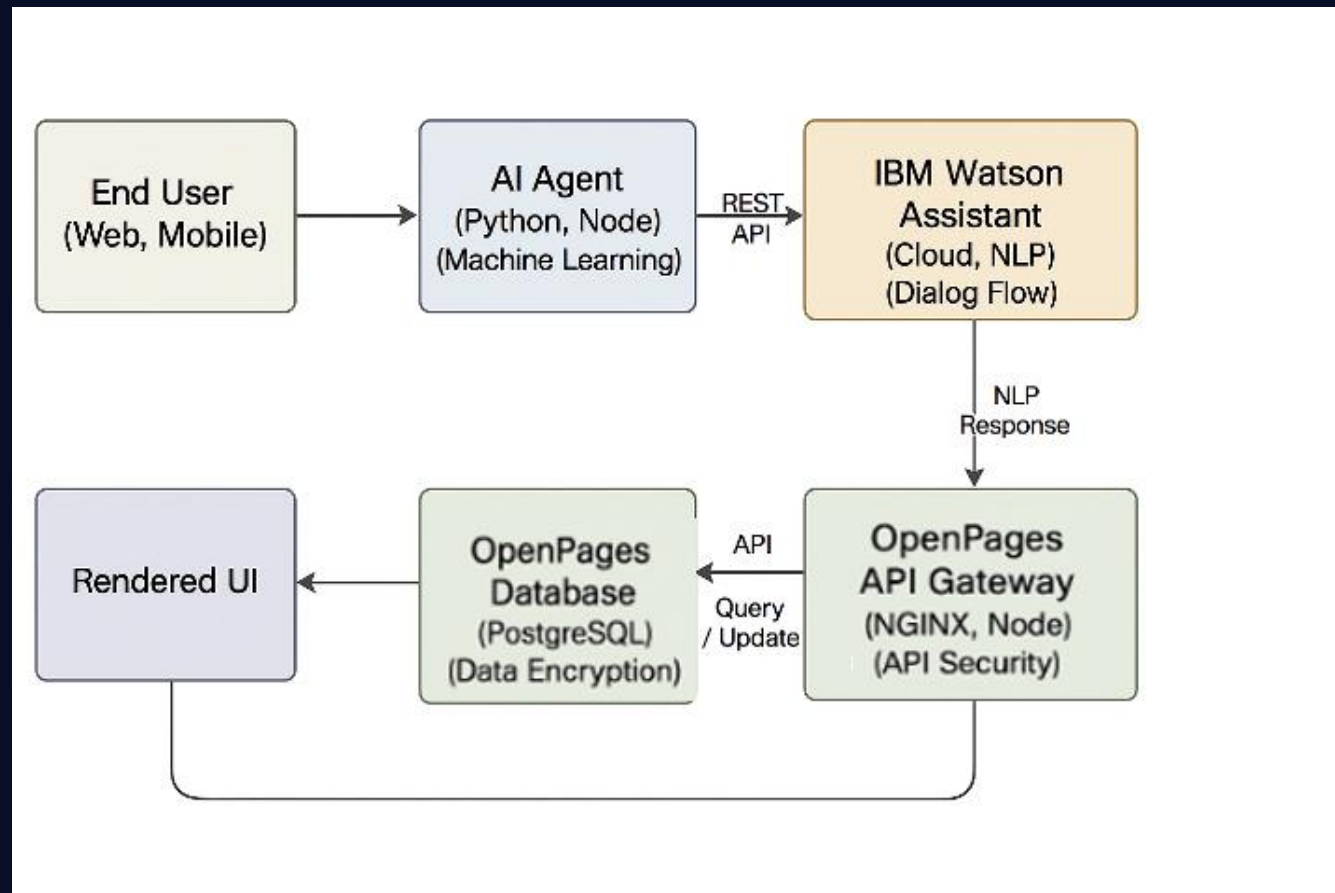
In this diagram, the flow is as follows:

- The **End User** interacts with the **AI Agent** to get insights and automate tasks in OpenPages.

- The **AI Agent** sends REST API calls to the **IBM Watson Assistant** to leverage its natural language processing (NLP) capabilities.

- The **IBM Watson Assistant** processes the user's input using NLP and sends the output to the **OpenPages API Gateway**.

- The **OpenPages API Gateway** receives the API calls and authenticates the request.

- The **OpenPages API Gateway** then calls the **OpenPages Database** to retrieve or update data.

- The **OpenPages Database** stores and retrieves data related to OpenPages.

- The **OpenPages UI** provides a user interface for users to interact with the system.

Building & Deploying AI Agent in Watsonx Orchestrate

Create watsonx orchestrate service in IBM Cloud
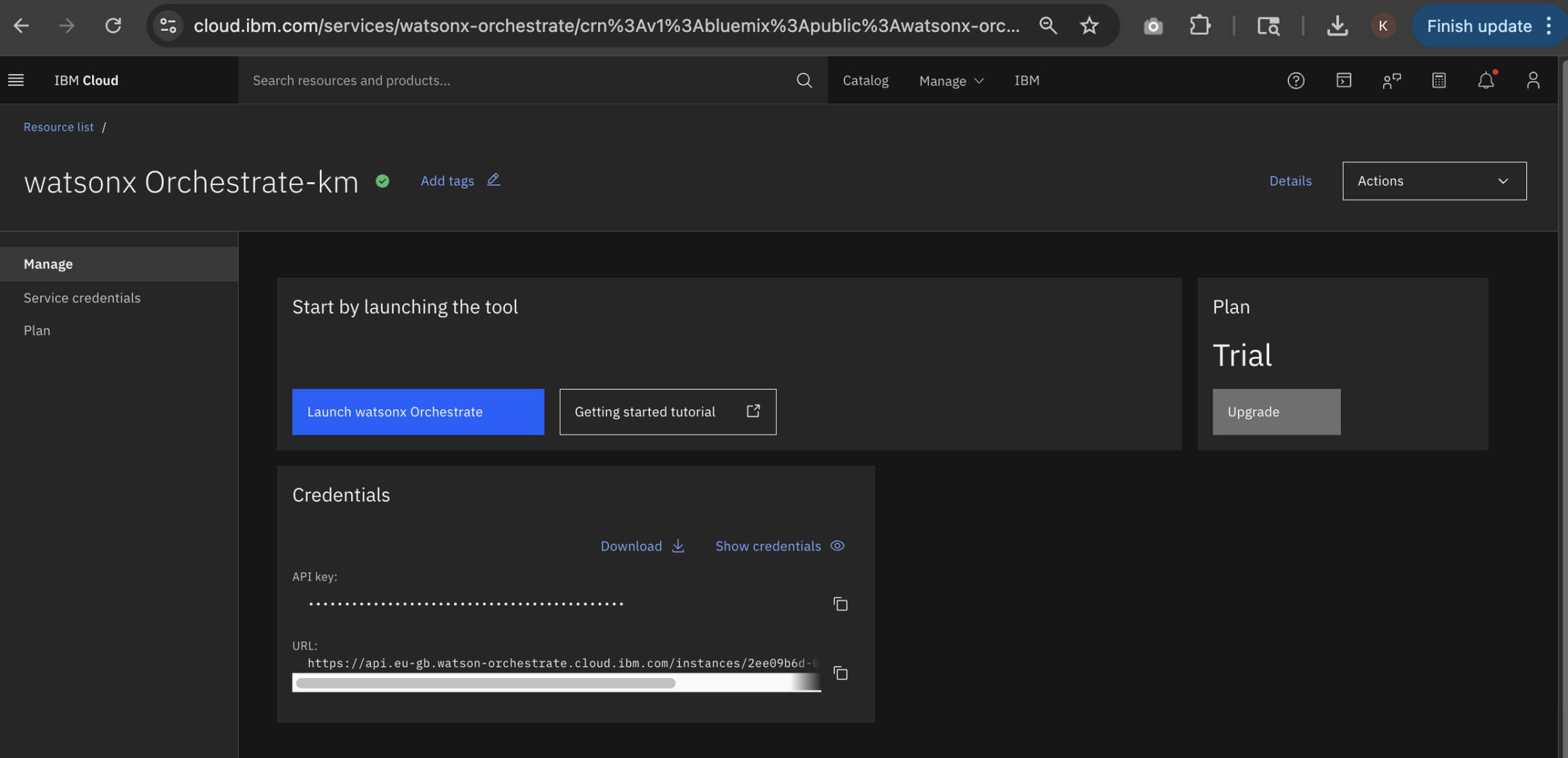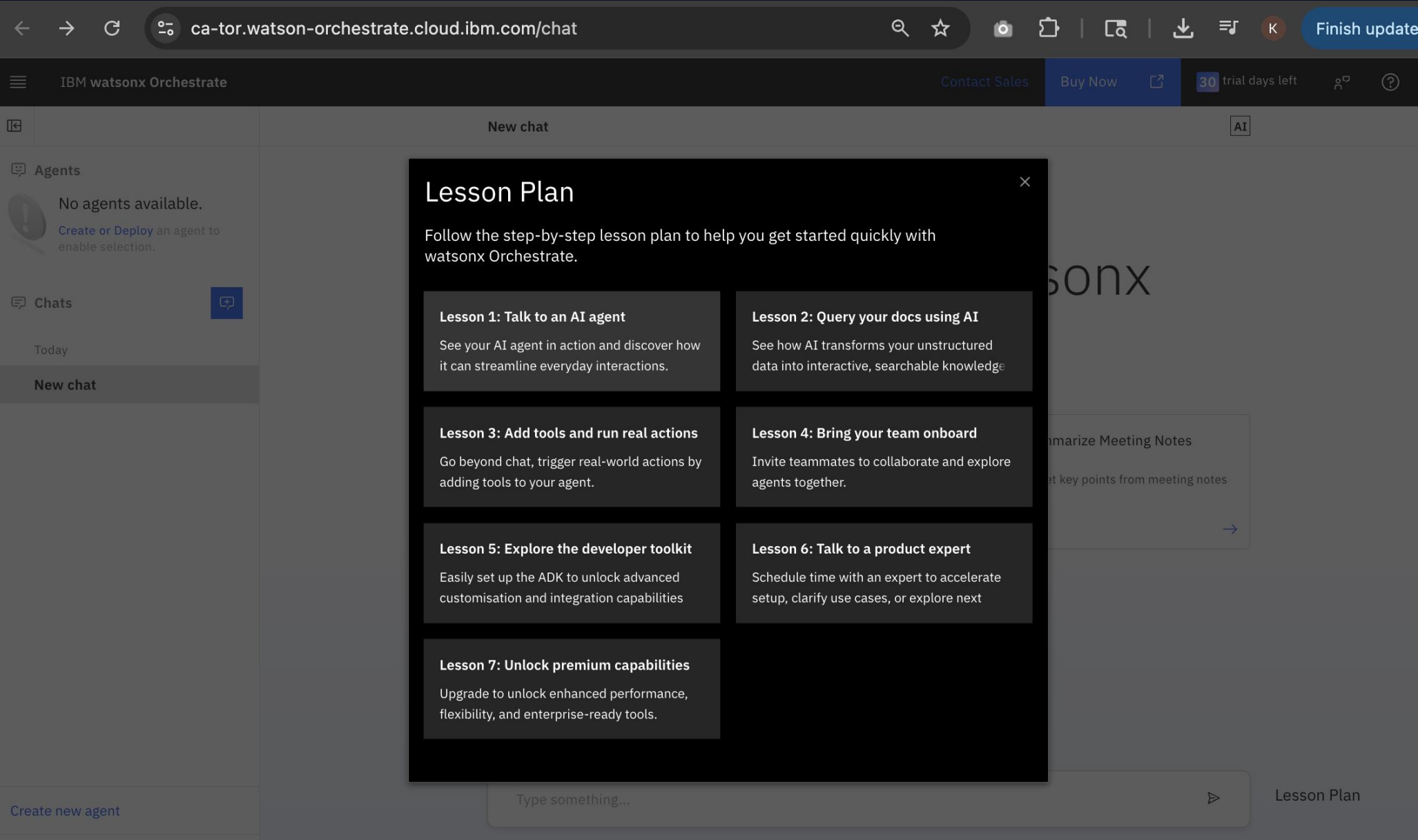

Steps Involved

- Log in to the Watson Orchestrate dashboard and navigate to the "Agents" tab.

- Click on "Create a new agent" and follow the prompts to create a new AI agent.

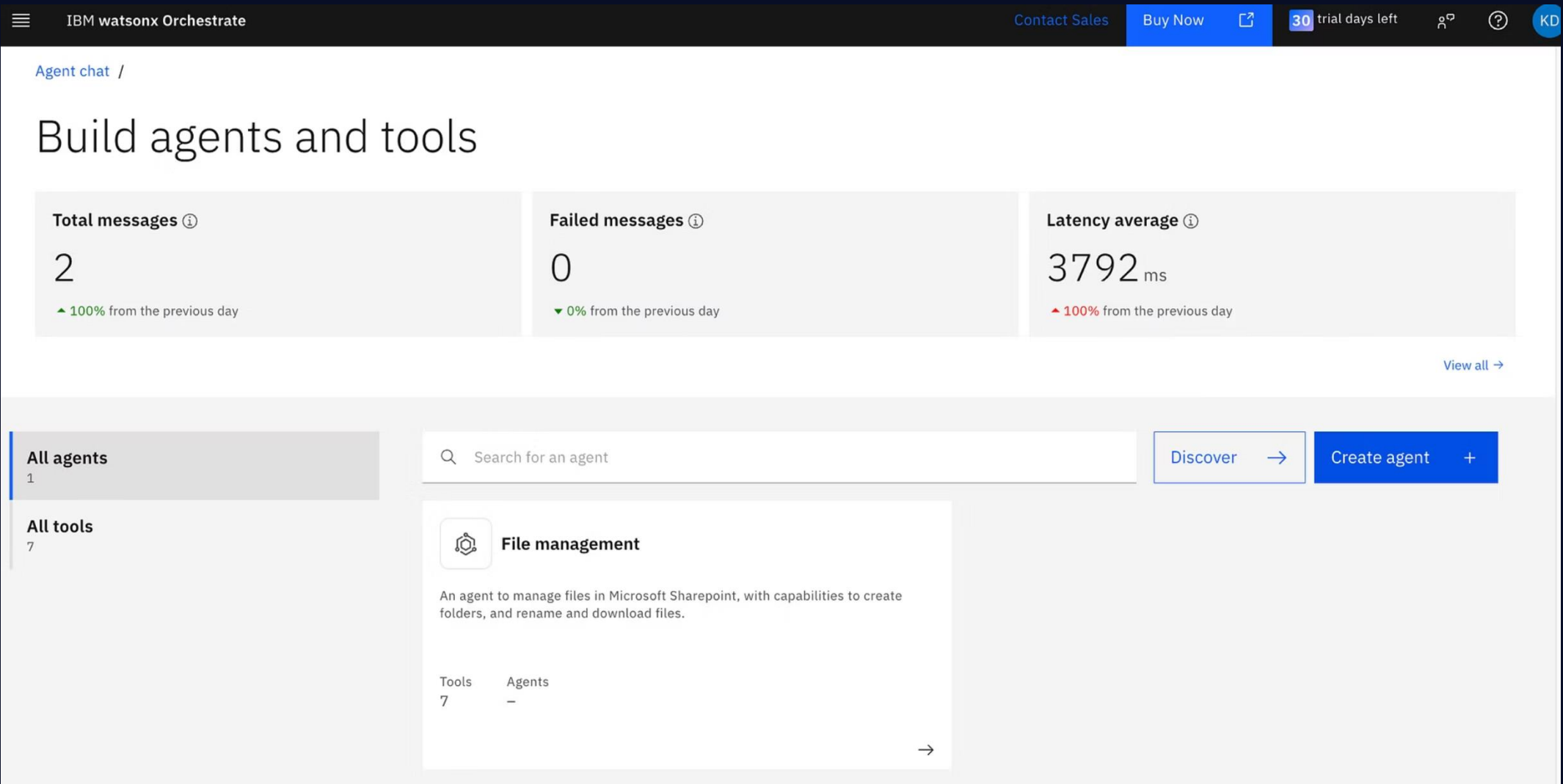- Configure the agent to use the Watson Orchestrate API.

Ref: https://www.ibm.com/docs/en/watsonx/watson-orchestrate/base

Launch watsonx orchestrate service in IBM Cloud

Watsonx Orchestrate Capabilities

Build AI Agent in Watsonx Orchestrate
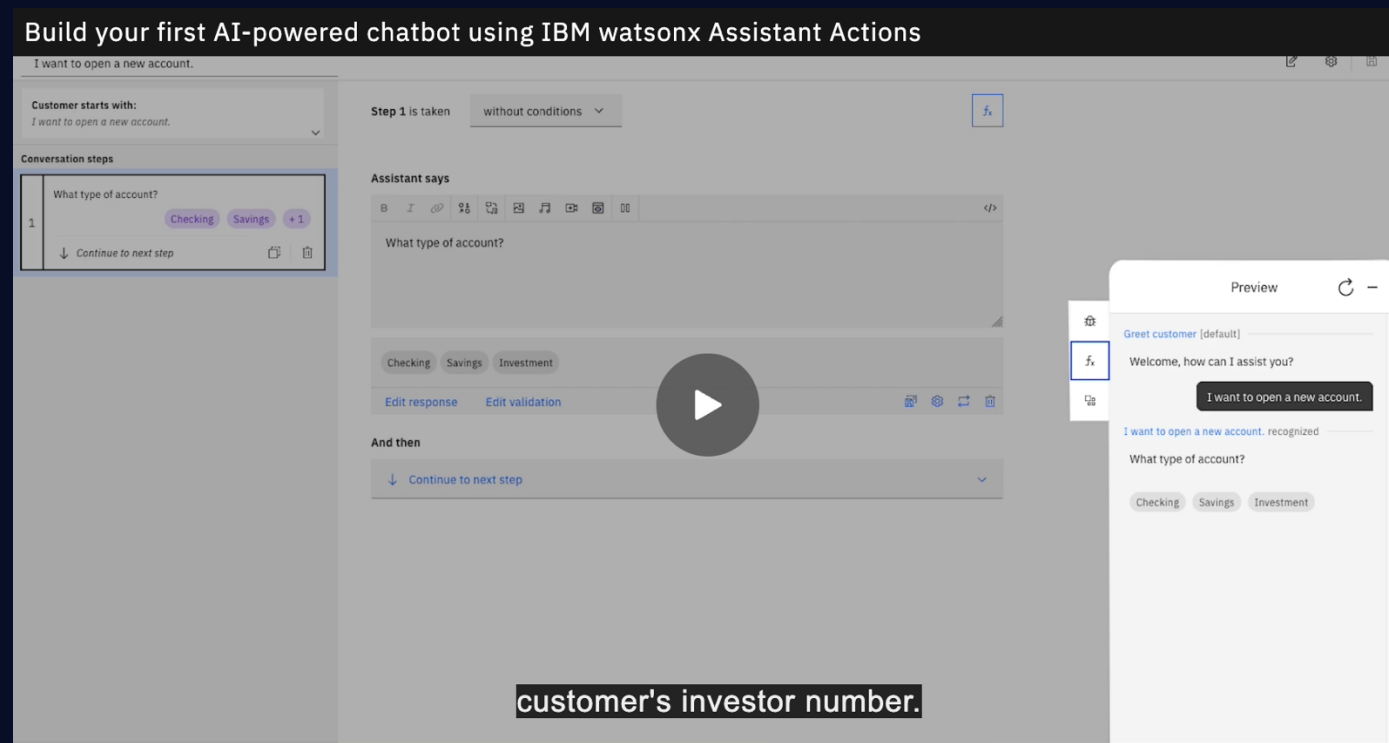

Ref: https://github.com/erifsx/ibm-watsonx-orchestrate

Deploy Agent

Integrate Watsonx Assistant with AI agent

Create Watsonx Assistant Chatbot Service

Steps Involved
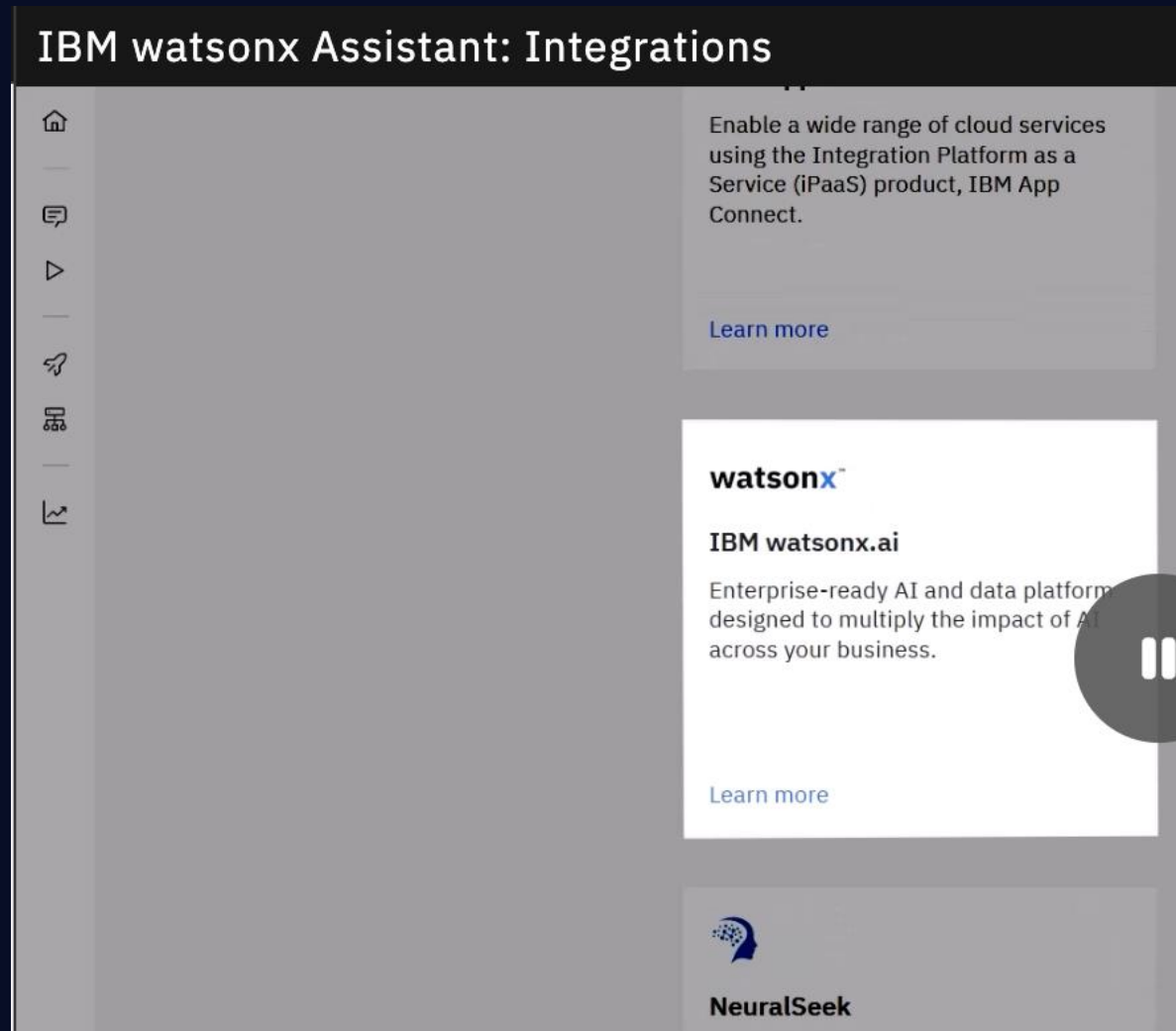
- Log in to the IBM Cloud dashboard and navigate to the Watson Assistant service.

- Click on "Create a new assistant" and follow the prompts to create a new chatbot.

- Configure the chatbot to use the Watson Assistant API.

Ref: https://www.ibm.com/products/watson-assistant/demos/lendyr/demo.html?page=build-experience-tour&ion=build-experience&panel=build-experience-building-conversation

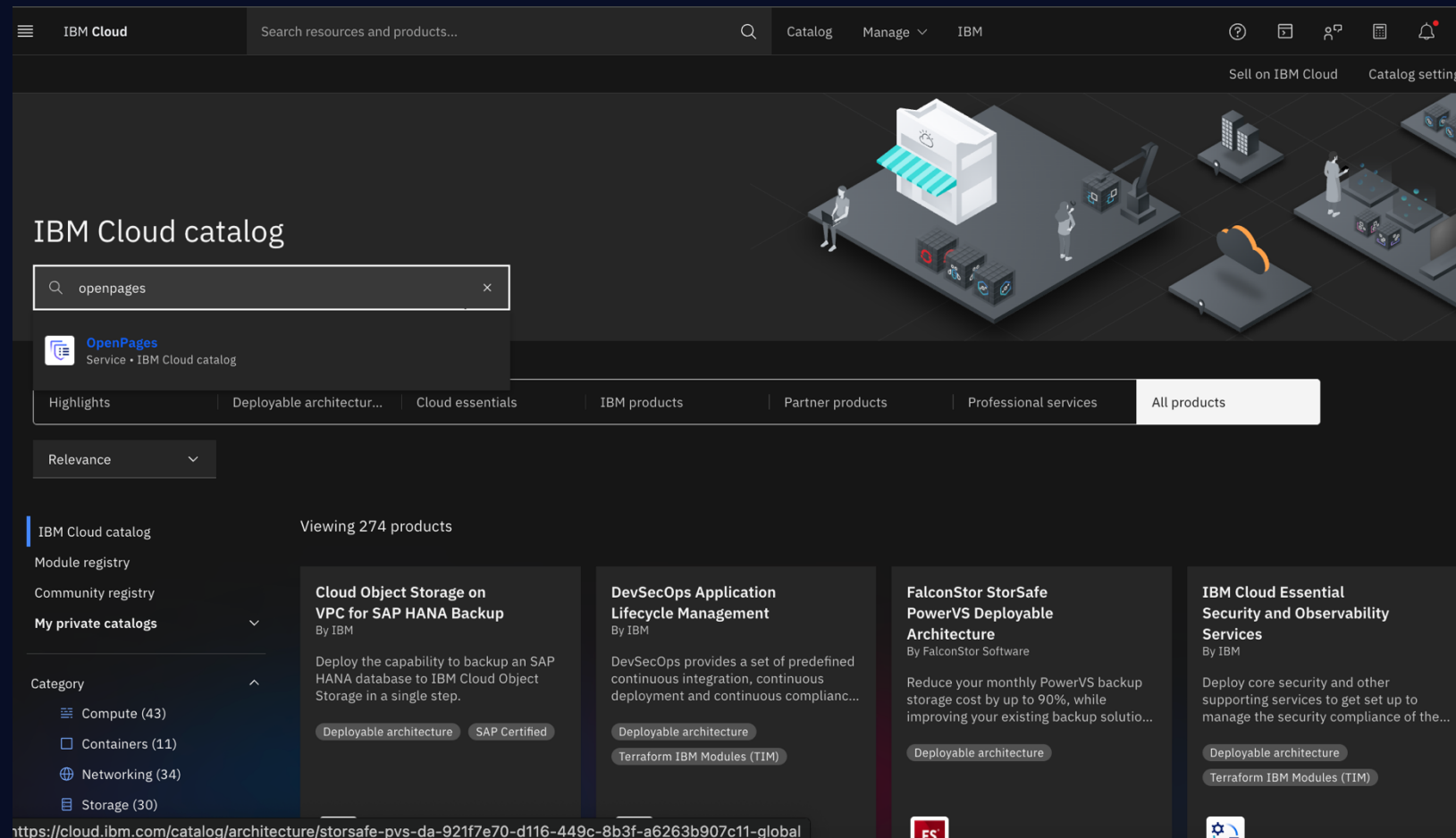# Integrate Watsonx Assistant Chatbot with AI Agent

Steps Involved

Configure the Watson Assistant chatbot to use the AI agent

- In the Watson Assistant dashboard, navigate to the "Settings" tab.

- Click on "Integrations" and select "Watson Orchestrate" as the integration type.

- Enter the Watson Orchestrate API credentials and configure the integration settings.

- Configure the chatbot to use the AI agent's intents and entities.

- In the Watson Orchestrate dashboard, navigate to the "Integrations" tab.

- Click on "Add a new integration" and select "Watson Assistant" as the integration type.

- Enter the Watson Assistant API credentials and configure the integration settings.

- Map the AI agent's intents and entities to the Watson Assistant chatbot's intents and entities.

Integrating Watsonx Assistant with IBM OpenPages

Search for OpenPages in IBM Cloud Catalog

https://cloud.ibm.com/docs/openpages?topic=openpages-gettingstartedtutorial

https://www.ibm.com/docs/en/openpages/9.1.x?topic=guide-openpages

https://cloud.ibm.com/apidocs/openpages

Create OpenPages Service in IBM Cloud

Steps Involved

- In the IBM OpenPages dashboard, navigate to the "Settings" tab.

- Click on "Integrations" and select "Watson Assistant" as the integration type.

- Enter the Watson Assistant API credentials and configure the integration settings.

- Configure the OpenPages application to use the Watson Assistant chatbot.

https://www.ibm.com/docs/en/openpages/9.1.x?topic=guide-openpages

https://cloud.ibm.com/apidocs/openpages

Compliance & Risk Assessment Tasks in OpenPages

Watsonx Assistant Chatbot in OpenPages

Implementation code

Code

Example code snippet on how to integrate the AI agent deployed within Watsonx Orchestrate to Watsonx Assistant chatbot within IBM OpenPages using Python

```python
import requests

# Watson Orchestrate API credentials
orchestrate_url = "https://api.us-south.orchestrate.watson.cloud.ibm.com"
orchestrate_username = "your_username"
orchestrate_password = "your_password"

# Watson Assistant API credentials
assistant_url = "https://api.us-south.assistant.watson.cloud.ibm.com"
assistant_username = "your_username"
assistant_password = "your_password"

# IBM OpenPages API credentials
openpages_url = "https://your_openpages_instance.com"
openpages_username = "your_username"
openpages_password = "your_password"

# Define the AI agent's intents and entities
intents = ["assign_compliance_issue", "approve_compliance_issue"]
entities = ["compliance_issue_id", "assignee"]

# Define the Watson Assistant chatbot's intents and entities
assistant_intents = ["assign_compliance_issue", "approve_compliance_issue"]
assistant_entities = ["compliance_issue_id", "assignee"]

# Integrate the AI agent with Watson Assistant chatbot
def integrate_ai_agent_with_assistant():
    # Authenticate with Watson Orchestrate
    orchestrate_auth = requests.auth.HTTPBasicAuth(orchestrate_username, orchestrate_password)

    # Authenticate with Watson Assistant
    assistant_auth = requests.auth.HTTPBasicAuth(assistant_username, assistant_password)

    # Map the AI agent's intents and entities to the Watson Assistant chatbot's intents and entities
    for intent in intents:
        for entity in entities:
            # Create a new intent in Watson Assistant
            intent_url = f"{assistant_url}/v1/workspaces/{assistant_workspace_id}/intents"
            intent_data = {
                "intent": intent,
                "entities": [entity]
            }
            response = requests.post(intent_url, auth=assistant_auth, json=intent_data)

            # Create a new entity in Watson Assistant
            entity_url = f"{assistant_url}/v1/workspaces/{assistant_workspace_id}/entities"
            entity_data = {
                "entity": entity,
                "values": [entity]
            }
            response = requests.post(entity_url, auth=assistant_auth, json=entity_data)

# Integrate the Watson Assistant chatbot with IBM OpenPages
def integrate_assistant_with_openpages():
    # Authenticate with IBM OpenPages
    openpages_auth = requests.auth.HTTPBasicAuth(openpages_username, openpages_password)

    # Configure the OpenPages application to use the Watson Assistant chatbot
    openpages_url = f"{openpages_url}/api/v1/chatbot"
    openpages_data = {
        "chatbot_id": assistant_workspace_id,
        "chatbot_url": assistant_url
    }
    response = requests.post(openpages_url, auth=openpages_auth, json=openpages_data)

integrate_assistant_with_openpages()

integrate_ai_agent_with_assistant()
```

Code

Example code snippet on how you can implement the implementation capabilities for the AI agent to create a new case in OpenPages, assign it to an investigator, and provide any necessary documentation or evidence using Python

```python
import watson_orchestrate

# Create a new agent
agent = watson_orchestrate.Agent(
    name="Compliance Issue Agent",
    description="This agent creates a new case, assigns it to an investigator, and provides documentation or evidence"
)

# Define the agent's steps
agent.steps = [
    watson_orchestrate.Step(
        name="Create Case",
        description="Create a new case in OpenPages",
        action=create_case
    ),
    watson_orchestrate.Step(
        name="Assign Case",
        description="Assign the case to an investigator",
        action=assign_case
    ),
    watson_orchestrate.Step(
        name="Provide Documentation",
        description="Provide documentation or evidence for the case",
        action=provide_documentation
    )
]

# Save the agent
agent.save()
```

```python
import watson_orchestrate

# Trigger the agent to run
agent = watson_orchestrate.Agent.get("Compliance Issue Agent")
agent.trigger()
```

```python
import requests
import json

# Define the OpenPages API endpoint and credentials
openpages_url = "https://your-openpages-instance.com/api/v1"
openpages_username = "your-username"
openpages_password = "your-password"

# Define the case details
case_title = "New Compliance Issue"
case_description = "This is a new compliance issue"
case_investigator = "investigator@example.com"

# Create a new case in OpenPages
def create_case(case_title, case_description, case_investigator):
    # Set the API endpoint and headers
    endpoint = f"{openpages_url}/cases"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Basic {openpages_username}:{openpages_password}"
    }

    # Set the case data
    case_data = {
        "title": case_title,
        "description": case_description,
        "investigator": case_investigator
    }

    # Make the API call to create the case
    response = requests.post(endpoint, headers=headers, json=case_data)

    # Check if the case was created successfully
    if response.status_code == 201:
        print("Case created successfully")
        return response.json()["id"]
    else:
        print("Error creating case:", response.text)
        return None

# Assign the case to an investigator
def assign_case(case_id, investigator):
    # Set the API endpoint and headers
    endpoint = f"{openpages_url}/cases/{case_id}/assign"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Basic {openpages_username}:{openpages_password}"
    }

    # Set the assignment data
    assignment_data = {
        "investigator": investigator
    }

    # Make the API call to assign the case
    response = requests.put(endpoint, headers=headers, json=assignment_data)

    # Check if the case was assigned successfully
    if response.status_code == 200:
        print("Case assigned successfully")
    else:
        print("Error assigning case:", response.text)

# Provide documentation or evidence for the case
def provide_documentation(case_id, documentation):
    # Set the API endpoint and headers
    endpoint = f"{openpages_url}/cases/{case_id}/documentation"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Basic {openpages_username}:{openpages_password}"
    }

    # Set the documentation data
    documentation_data = {
        "documentation": documentation
    }

    # Make the API call to provide the documentation
    response = requests.post(endpoint, headers=headers, json=documentation_data)

    # Check if the documentation was provided successfully
    if response.status_code == 201:
        print("Documentation provided successfully")
    else:
        print("Error providing documentation:", response.text)

# Create a new case, assign it to an investigator, and provide documentation
case_id = create_case(case_title, case_description, case_investigator)
if case_id:
    assign_case(case_id, case_investigator)
    provide_documentation(case_id, "This is some documentation for the case")
```

End User Training & Support

IBM OpenPages with Watson Training Material

- https://www.credly.com/badges/ad433cae-d940-4e82-9812-cd3b7cd619d6

- https://learn.ibm.com/mod/video/view.php?id=165039

- IBM AI ML Ops for IT Operations https://www.ibm.com/solutions/ai-assistants#ai-assistants