

# Deploying Loan Risk AI Agent on IBM Cloud Code Engine

GitHub repository: <https://github.com/IBM/ai-agent-for-loan-risk>

## Table of Contents

<b>Prerequisites .....</b>	<b>1</b>
<b>Deployment Steps .....</b>	<b>2</b>
1. Confirm prerequisites and gather information .....	2
2. Create a Project in watsonx.ai .....	2
3. Create a namespace in Container Registry .....	4
4. Deploy application to Code Engine .....	5
5. Validate the application. ....	9
6. Enhance application with additional optional features. ....	9
<b>Optional Features .....</b>	<b>10</b>

## Prerequisites

Requires IBM Cloud account with:

- watsonx.ai services ([refer](#)). This includes:
  - watsonx.ai Runtime
  - watsonx.ai Studio
  - Cloud Object Storage
- Account user with administrative privileges for:
  - watsonx.ai – to access services, create Projects for storing assets, get Project ID etc. ([refer](#))
  - Code Engine – to create Project and deploy the application. ([refer1](#) [refer2](#))
  - Container Registry – to create namespace, store images for deploying the application (
- Account user's API key ([refer](#))

# Deployment Steps

*Note on costs:* The deployment will incur costs for Code Engine runtime, watsonx.ai runtime and watsonx.ai LLM inferencing.

## 1. Confirm prerequisites and gather information

- 1.1. Note the user's API key value. It will be needed for creating container registry secret. It will also be the environment variable WATSONX\_AI\_APIKEY.
- 1.2. Note the region where watsonx.ai is deployed. Get the service URL API endpoint for your region from [this list](#) - e.g., for us-south/Dallas region it is <https://us-south.ml.cloud.ibm.com>. This will be the environment variable WATSONX\_SERVICE\_URL.
- 1.3. Note/decide the region to deploy the application resources, e.g. us-south, use-east, eu-de etc.
- 1.4. Note/decide/create the resource group to use in the account, e.g. agentic-ai-rg

## 2. Create a Project in watsonx.ai

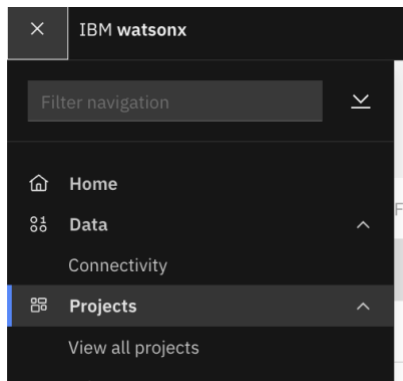
(Or use an existing one and capture its Project Id)

To create a new Project, you will need to configure the Object Storage. Associate the Project with watsonx.ai Runtime service.

Detailed steps for reference:

- 2.1. Create a Project. Also define the Cloud Object Storage for storage.

<https://dataplatform.cloud.ibm.com/docs/content/wsj/getting-started/projects.html?context=wx&audience=wdp>  
<https://dataplatform.cloud.ibm.com/projects/?context=wx>



IBM watsonx

Projects

Your active projects

Find a project

New project

Create a project

Start with a new, blank project or select from where to import an existing project.

New

Local file

Sample

Define details

Name

agentic-ai-wx-project

Description (optional)

for agentic ai app

Tags (optional)

Add tags

Add tags to make projects easier to find. To add tags, separate them with commas and press Enter.

Define storage

Select storage service

Target Cloud Object Storage Instance

watsonx-poc-wx-shared-cos-instance

Project includes integration with [Cloud Object Storage](#) for storing project assets.

Advanced settings

Cancel

Create

2.2. From the Manage tab > Services and integrations >Associate service, - Associate it with the watsonx.ai Runtime service.

Projects / agentic-ai-wx-project

Overview

Assets

Deployments

Jobs

Manage

Project

General

Access control

Environments

Resource usage

Services & integrations

Services & integrations

IBM services (1)

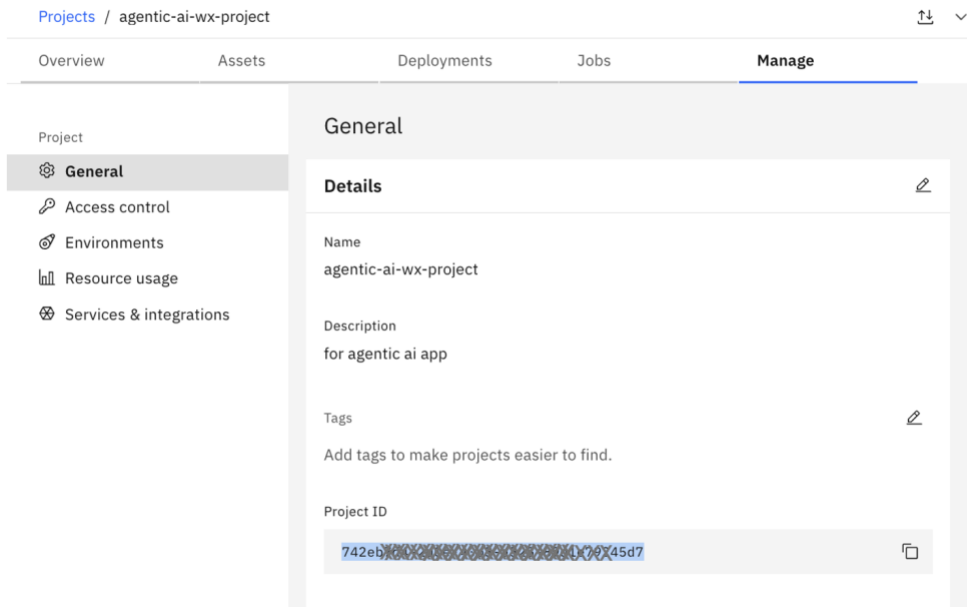
Third-party integrations

Find services

Associate service

Name	Service type
rag-common-watson-machine-learning-instance	watsonx.ai Runtime

### 2.3. From the Manage > General tab, capture the Project ID



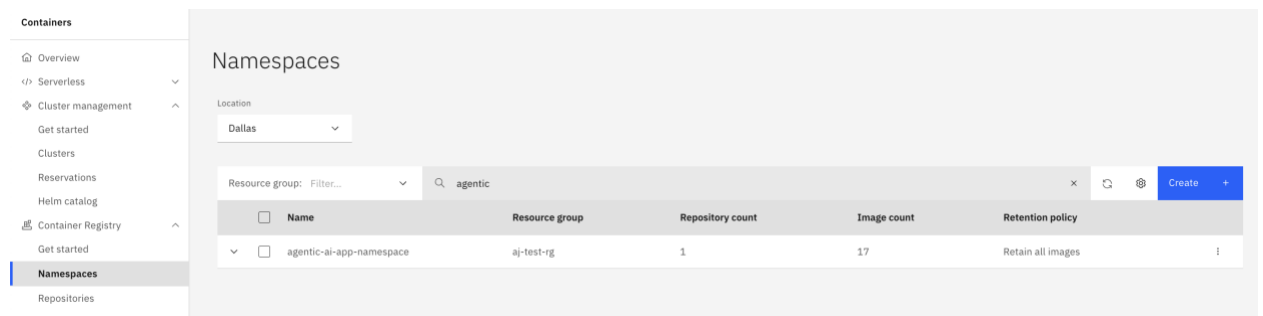
2.4. Note the Project Id. This will be the environment variable `WATSONX_PROJECT_ID`.

### 3. Create a namespace in Container Registry

Detailed steps for reference:

<https://cloud.ibm.com/containers/registry/namespaces>

Name: agentic-ai-app-namespace



## 4. Deploy application to Code Engine

Detailed steps for reference:

Refer section [Deploying your app from repository source code from the console](#)

Below is some information you will need to complete the steps.


### 4.1. Click Create Project


- Project name: agentic-ai-app-project, Select region and resource group.
- Component type. Select Application. Application name: agentic-ai-app
- Code. Select Build Container image from source.
- Code repo URL: <https://github.com/IBM/ai-agent-for-loan-risk.git>


Project

agentic-ai-app-project Dallas (us-south) ▼

Component type ⓘ

 **Application** ✓  
Run your code to serve HTTP requests.  
Examples: frontend server of a web app, micro-service as part of a solution.

 **Job**  
Run your code to complete tasks.  
Examples: payroll processing, PDF creation for mail attachments.

 **Function**  
Run your code with low latency on top of a runtime.  
Examples: web hook target, glue code snippet.

Name

agentic-ai-app-v1

Code

Specify a container image or build one from source code first. To learn more, [see the documentation](#).

Use an existing container image

Build container image from source code ✓

Code repo URL

https://github.com/IBM/ai-agent-for-loan-risk.git

Specify build details

Try this sample:

https://github.com/IBM/CodeEngine

## 4.2. Specify build details.

Mostly use defaults. But create new registry secret.

Specify build details

Learn more about [image builds](#).

Source

Strategy

Output

Code repo URL ⓘ  

https://github.com/IBM/ai-agent-for-loan-risk.git

Try this sample: 

https://github.com/IBM/CodeEngine

SSH secret ⓘ  

None

Select existing or create new SSH secret

Branch name ⓘ  

Example: main

Context directory ⓘ  

Optionally specify an alternate root folder

Specify build details

Learn more about [image builds](#).

Source

Strategy

Output

Strategy  

Dockerfile

Cloud Native Buildpack

Dockerfile  

Dockerfile

Timeout ⓘ  

10m

Build resources ⓘ  

M (1 vCPU / 4 GB)

## 4.3. Create registry secret.

Name agentic-ai-app-secret.

You will also need to select region, enter the user's API key, and select the namespace that was created earlier.

Create registry secret

Stores credentials to access a container registry.

Secret name  

agentic-ai-app-secret-v1

Secret contents  
Target registry  

IBM Container Registry

Docker Hub

Other

Location  

Dallas (private.us.icr.io)

Username  
iamapikey

IAM API key ⓘ  

.....

Email (optional)  

username@email.com

## Specify build details

×

Learn more about [image builds](#).

☒ Source

☒ Strategy

☐ Output

Registry server ⓘ

private.us.icr.io

× | ▾

Select registry or type in registry hostname

Registry secret ⓘ

agentic-ai-app-secret

▾

Select existing or create new registry secret

Namespace

agentic-ai-app-namespace

× | ▾

Select or type in namespace name

Repository (image name)

▾

Select or type in repository name

Tag

latest

▾

Select or type in version tag

### 4.4. Once done with Specify build details, this is what the Code section will look.

#### Code

Specify a container image or build one from source code first. To learn more, [see the documentation](#).

Use an existing  
container image

Build container image  
from source code ✓

#### Image build details

[Edit build details](#) ↗

Image reference	private.us.icr.io/agentic-ai-app-namespace/agentic-ai-app-image
Registry access secret	agentic-ai-app-secret-v1
Source code URL	https://github.com/IBM/ai-agent-for-loan-risk.git
Branch name	
Context directory	/

#### 4.5. Scroll down to optional settings and add environment variables.

- WATSONX\_AI\_APIKEY
- WATSONX\_PROJECT\_ID
- WATSONX\_SERVICE\_URL

### Optional settings

Environment variables

Define key-value pairs that can be used by your running code.

Search

Add

Name	Defined by	Value or reference
WATSONX_AI_APIKEY	Literal	U29wIH...XXXXXXXXXX
WATSONX_PROJECT_ID	Literal	2d1dfce2...da
WATSONX_SERVICE_URL	Literal	https://us-south.ml.cloud.ibm.com

#### 4.6. Create/deploy the application. Check status

Containers / Serverless / Projects / agentic-ai-app-project / Applications /

agentic-ai-app-v1 Deploying

Test application

Containers / Serverless / Projects / agentic-ai-app-project / Applications /

agentic-ai-app-v1 Ready

Test application

Overview Instances Configuration Domain mappings Service access

Active instances

View the active instances of this application, and their allocated resources.

Instances00

CPU (vCPU)00

Memory (GB)00

Number of instances

agentic-ai-app-v1-00001

Configuration revisions

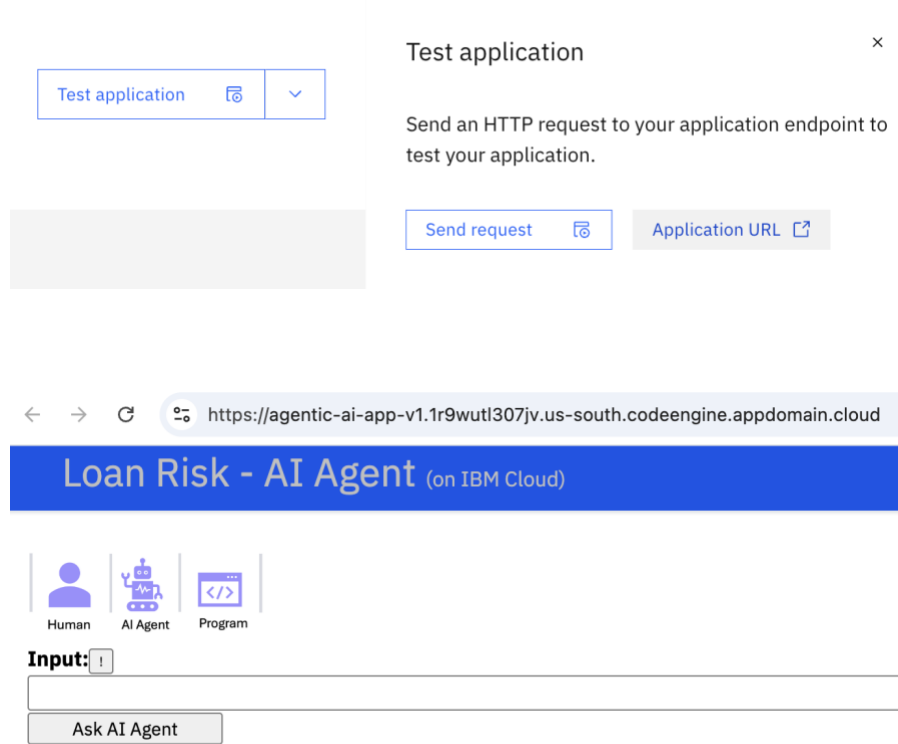
Applications have one or more revisions of the configuration properties. Each update creates a new revision.

Search

<input type="checkbox"/>	Name	Tag	Status	Traffic	Created
<input type="checkbox"/>	agentic-ai-app-v1-00001	latest	Ready	100%	about 3 hours ago



4.7. Click on Test application (right top) > Application URL to launch application.



## 5. Validate the application.

Ask “What is the interest rate for matt? Explain how it was determined?”. You should get a response explain high risk and 8% interest rate.

Refer to the application [usage section](#).

## 6. Enhance application with additional optional features.

Once initial application is deployed and running successfully, you can add enhancements.

- Using RAG LLM (Agentic RAG feature)
- Using watsonx Assistant/Orchestrate (Chat widget)

Refer to Optional Features section.

## Optional Features

Once initial application is deployed and running successfully, you can enhance with some additional optional features.

- Using RAG LLM (Agentic RAG feature)
- Using watsonx Assistant/Orchestrate (Chat widget)

### Using RAG LLM (Agentic RAG feature):

By default, the risk and interest rate tools of the AI agent simulate risk and interest rate determination. By adding this feature, the AI agent tool will make RAG query to retrieve relevant content from the bank documents and use that to determine the risk and interest rate. In the AI agent response you will be able to see the content from the document table and its interpretation by the AI agent.

- Create a new watsonx.ai Deployment Space with Deployment stage set as Production. ([refer](#))
- Create a vector index asset in the watsonx.ai Project using these [content PDF documents](#). To create the index use vector store - "In memory", embedding model - "allminilm-l6-v2", chunksize - "2000", chunk overlap - "200".
- Open the vector index you just created in Prompt Lab and set the generative AI model to "mistral-large". Test the index by asking some questions e.g., what is the risk for credit score 655 and account status closed?, what is the interest rate for medium risk?, and confirm answers are using RAG from the content PDF documents.
- Deploy the vector index as watsonx.ai Deployment on AI service for inferencing using the "fast path" option (use the "Deploy" button). ([refer-1](#)) ([refer-2](#)) and ([refer-3](#)).
- Capture the watsonx.ai Deployment private endpoint for the vector index for RAG inferencing (use non-stream; ends with ai\_service?version=...)
- On Code Engine add the following environment variables with the values captured above and redeploy the application (ENABLE\_RAG\_LLM=true and WATSONX\_RISK\_RAG\_LLM\_ENDPOINT=endpoint captured above)
- The application will now use the RAG content for risk and interest tools.

### Using watsonx Assistant/Orchestrate (Chat widget):

By adding this feature, you can get a more conversational/chat experience when asking questions in the watsonx Assistant chat widget. The conversation is single turn and the watsonx Assistant skills can be enhanced further if needed.

- Note the Code Engine URL for the deployed application.

- Open API file (agentic-ai-app-custom-ext-openapi.json) and update the URL with the deployed applicaiton URL.
- Create an action skill in watsonx Assistant instance
- Create and add a custom extension by importing the updated Open API file (agentic-ai-app-custom-ext-openapi.json).
- Import the zip file to set up the actions that use the custom extension (wx-asst-agentic-ai-app.zip)
- Open the watsonx Assistant Web chat configuration and note the integrationID, region and serviceInstanceID from the Embed script tab.
- On Code Engine open the deployed application configuration, add the following environment variables with the values captured above and redeploy the application (ENABLE\_WXASST=true, WXASST\_INTEGRATION\_ID, WXASST\_REGION, WXASST\_SERVICE\_INSTANCE\_ID captured above)
- The watsonx Assistant will become available on the page <application-url>/wx.html