**Unified Insurance Claims Processing: Integrating WatsonX Assistant with IBM Cloud Services**

Contents

[**1.** **Overview** 3](#_Toc191764343)

[**2.** **Why do we need such a system ?** 3](#_Toc191764344)

[**3.** **Tools and Technologies Used** 4](#_Toc191764345)

[**4.** **High Level Architecture diagram** 4](#_Toc191764346)

[**5.** **Solution Implementation Steps** 5](#_Toc191764347)

[**6.** **Workflow** 12](#_Toc191764348)

[**7.** **Appendix** 27](#_Toc191764349)

# **Overview**

This document outlines the architecture and processes for optimizing policy generation and claims management through the integration of **WatsonX Assistant** and **IBM Cloud Services**. It details the end-to-end workflow, from policy creation to claims processing, enabled by WatsonX Assistant’s AI-driven interactions.

The document further explores the **IBM Cloud solutions** leveraged to build and integrate the ecosystem, including:

* **Compute Services**.
* **Database Solutions**
* **AI/Machine Learning**

Additionally, it provides implementation guidance for integrating these services with the **.NET Core framework**, ensuring seamless interoperability, which will work as a baseline related to development in this area of business needs.

# **Why do we need such a system ?**

**Context and Challenges in Modern Insurance Claim Management**  
The insurance industry faces persistent challenges in streamlining complex Claim creation, management, and claims processing workflows. Existing systems often require intricate multi-platform integrations, resulting in fragmented customer experiences. Policyholders must engage in time-consuming manual processes, such as coordinating with support teams to verify policy details or navigate convoluted claim submission procedures. These inefficiencies lead to operational delays, increased costs, and customer dissatisfaction.

**Proposed Solution: AI-Driven Policy Lifecycle Automation**  
By integrating **IBM WatsonX Assistant** with **IBM Cloud services**, this ecosystem automates end-to-end policy generation and claims processing. Customers interact directly with WatsonX Assistant, an AI-driven interface that guides them through policy creation in minimal steps and simplifies claims submission. The system abstracts backend complexities, enabling:

* **Accelerated Policy Issuance**: AI-guided workflows reduce manual intervention, shortening policy creation timelines.
* **Seamless Claims Processing**: Real-time data validation and automated decision-making minimize delays.
* **Enhanced Customer Experience**: A unified interface eliminates redundant interactions and reduces errors.

**Technical Foundation: IBM Cloud Ecosystem**  
The solution leverages IBM Cloud’s robust infrastructure, including:

* **Compute and Database Services**: Scalable runtime environments and secure data storage.
* **AI/ML Capabilities**: WatsonX’s natural language processing (NLP) and predictive analytics for dynamic customer interactions.
* **Integration with .NET Core**: APIs and middleware ensure interoperability with existing enterprise systems.

**Competitive Advantage Over Traditional Cloud Platforms**  
Unlike generic cloud providers (e.g., Azure, AWS), IBM’s tailored integration of WatsonX Assistant with IBM Cloud delivers industry-specific advantages, such as pre-built insurance workflows, hybrid-cloud compliance, and enterprise-grade security. This reduces development overhead and accelerates time-to-value while maintaining scalability.

# **Tools and Technologies Used**

**Application Development Technology –**

* .Net 8
* C#
* Visual Studio 2022

**Code Versioning**

* GitHub

**Chat Assistant**

* IBM WatsonX Assistant

**Deployment and Integration –**

* **Cloudant** – IBM cloud JSON database
* **MQ –** IBM cloud message broker service
* **Kubernetes –** IBM Kubernetes Service for deployment
* **VPS –** Virtual Private Cloud on top of Kubernetes.

**Architecture –**

* **Microservice**

# **High Level Architecture diagram**

|  |
| --- |
|  |

# **Solution Implementation Steps**

**Step 1 – Create Policy and Claims Microservice**

I have used .Net 8 framework to build the microservices. One is Policy and another one is Claims.

The two services are small and independent of other services or databases and functions as independent service providers.

|  |
| --- |
|  |

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
|  |

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

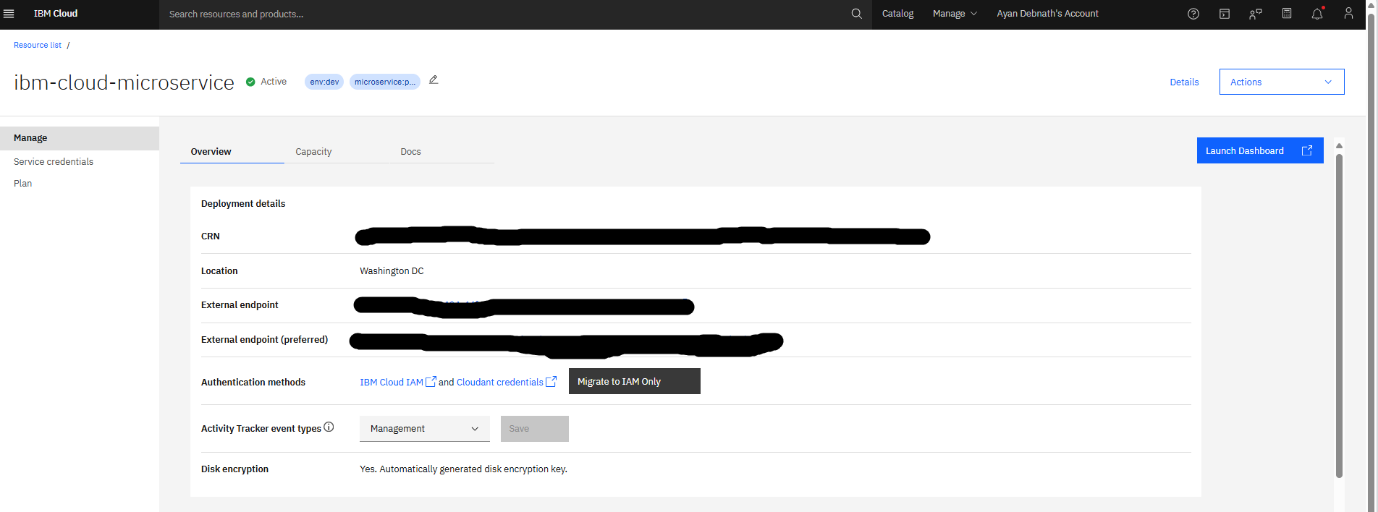
**Step 2 – Create Cloudant Account and Database**

Next, I will create one IBM cloudant DB for policy microservice and another IBM cloudant DB for claims microservice.

This system design will help me to create independent microservice, with different database.

**Policy Cloudant Account –**

**Database Name – “*policy-dev”***

****

**A screenshot of a computer

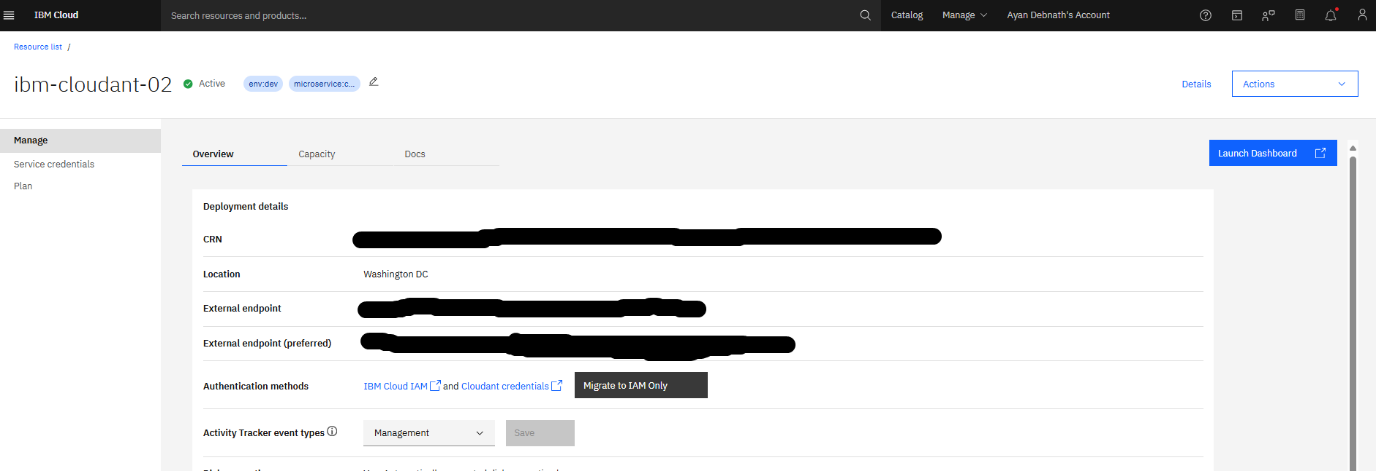
AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Claims Cloudant Account**

**Database Name - "claims-dev” and “policy-dev”.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 3 – Create IBM MQ Message Broker**

The microservice will communicate between each other by a message broker system build around IBM MQ.

The message broker system will reduce the dependency among the microservice as communication will happen between them via the IBM MQ.

The policy microservice will create a policy and publish a message to IBM MQ queue[DEV.QUEUE.1]. The claims microservice subscribes to the same queue, once the message is published the claims will add the policy details and save it in the DB.

Similarly, once a claim is submitted for a policy , the claims microservice pushes a message to another queue [DEV.QUEUE.2] , triggering policy microservice who is subscribing to the queue. The policy microservice will read the claim details and will update the record in its one database.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

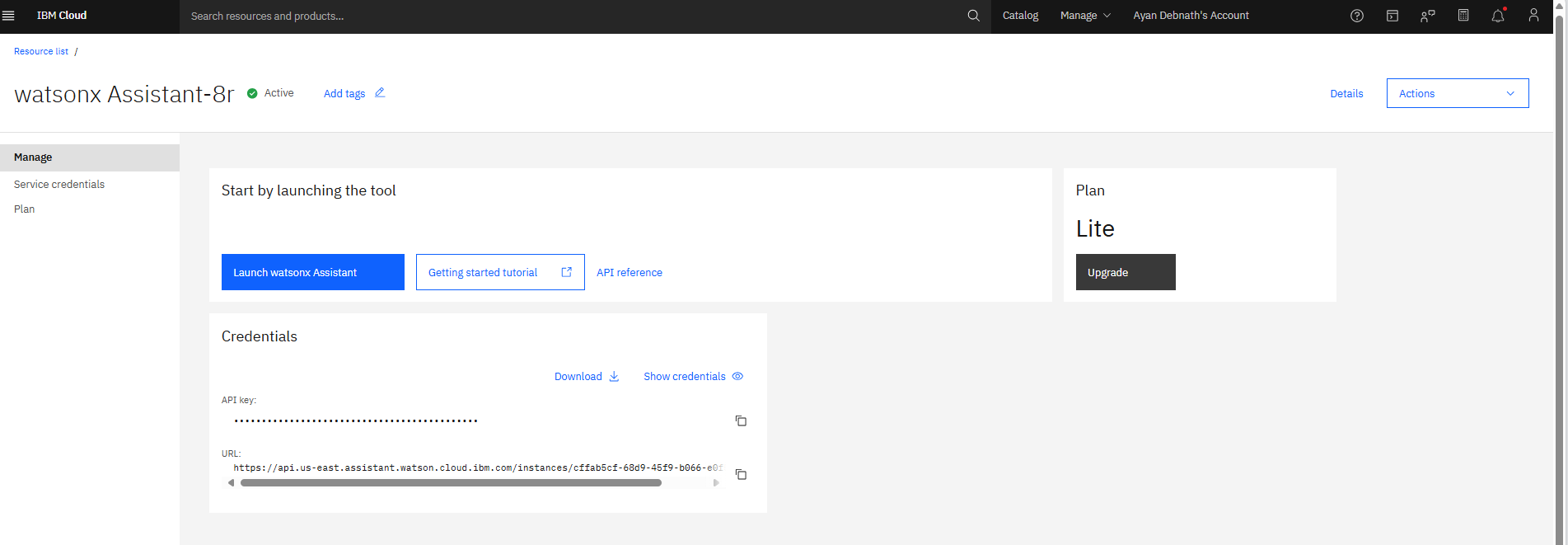
AI-generated content may be incorrect.

**Step 3 – Create Watson Assistant**

I have created a Watson X Assistant service, which has the 2 actions.

Policy Assistant – It will help to create a policy via the assistant

Claims Assistant – It will help to claim the created the policy

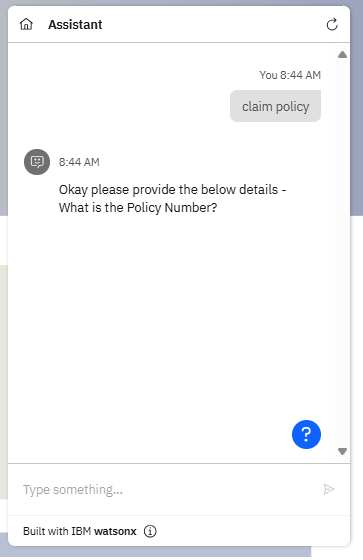
****

**A screenshot of a computer

AI-generated content may be incorrect.**

# **Workflow**

**Step 1 – Claim the policy via WatsonX assistant**

****

**A screenshot of a chat

AI-generated content may be incorrect.**

**A screenshot of a chat

AI-generated content may be incorrect.**

**A screenshot of a phone

AI-generated content may be incorrect.**

**Step 2 – Verify the claim details in the Claims DB**

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Step 3 – Claim MS will publish the data to IBM MQ**

Once the policy has been claimed via Watson X assistant, the claims MS will publish the message to IBM MQ queue (DEV.QUQUE.2).

Once the message has been published, this will trigger the policy MS , that will create the claims details in policy DB of the policy microservice.

A screenshot of a computer

AI-generated content may be incorrect.

# **Appendix**

Repo Link - <https://github.com/sourabhghatak/IBM-INS-MICROSERVICE-CLD>