Aman Raj

Superset ID: 6358186

**Kafka Integration with C#:**

**Introduction to Kafka**

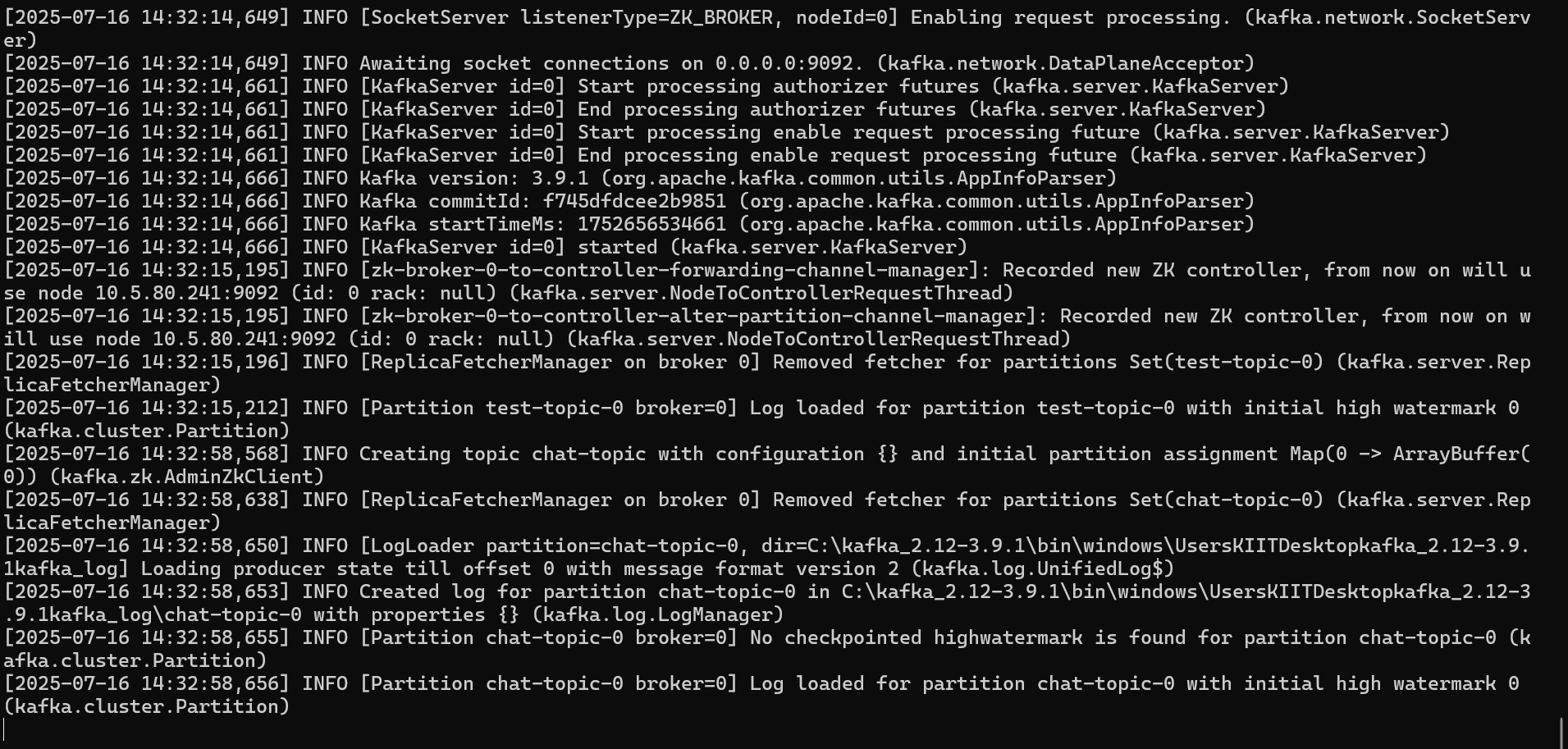
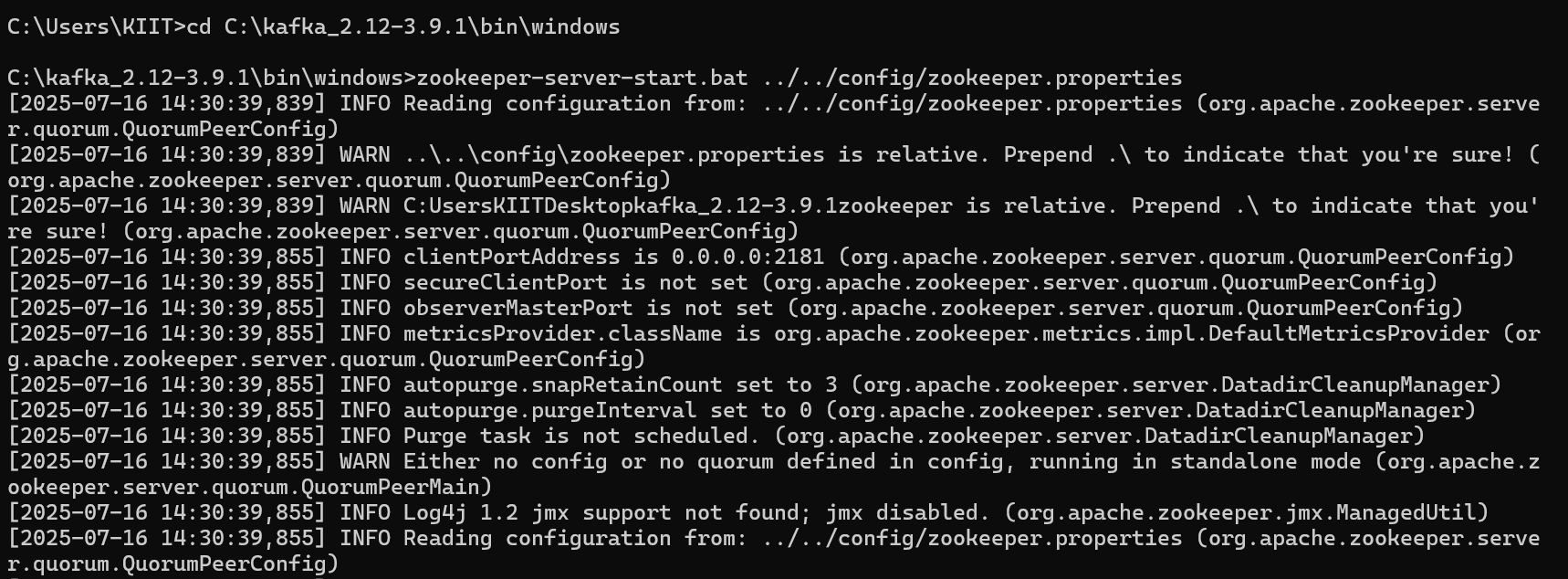
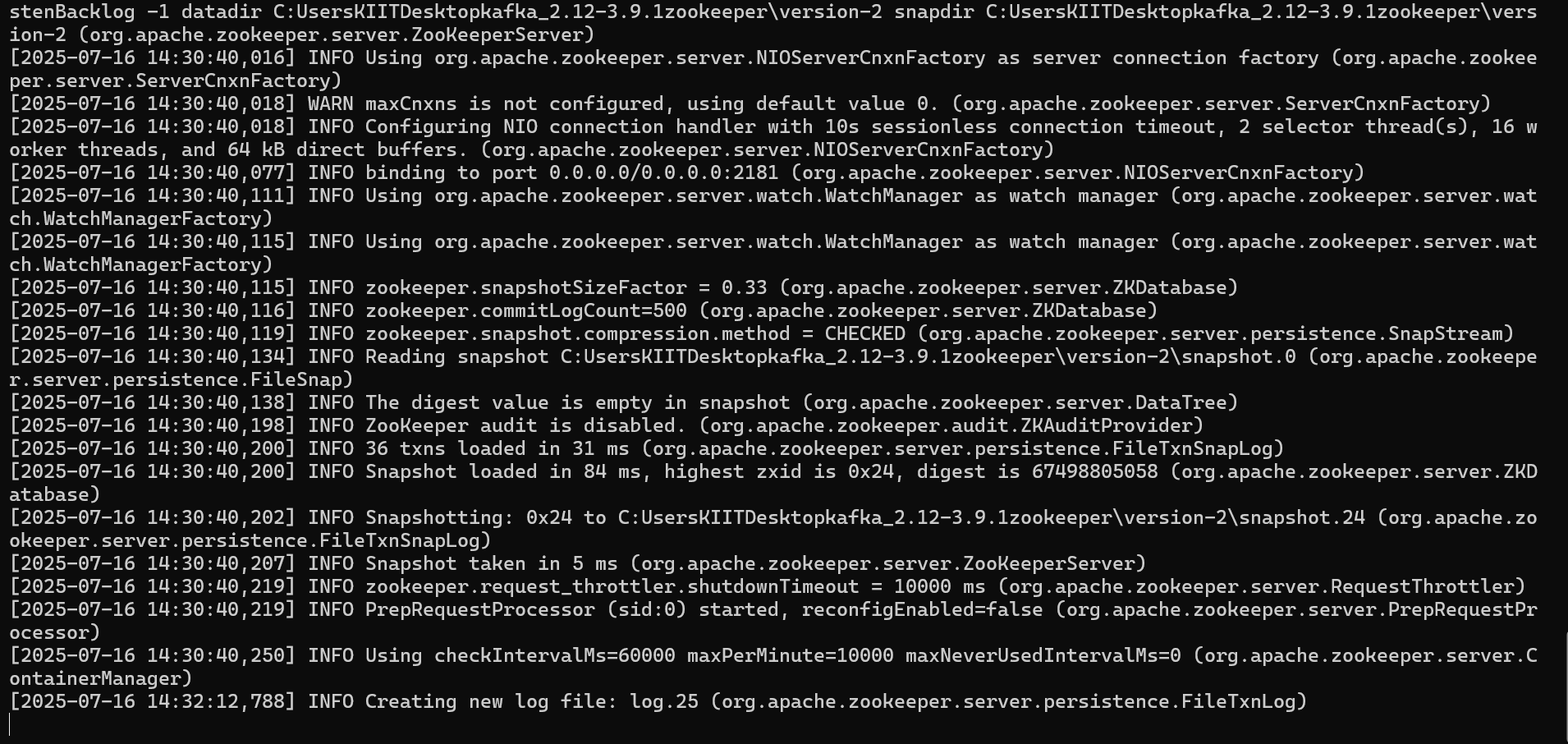
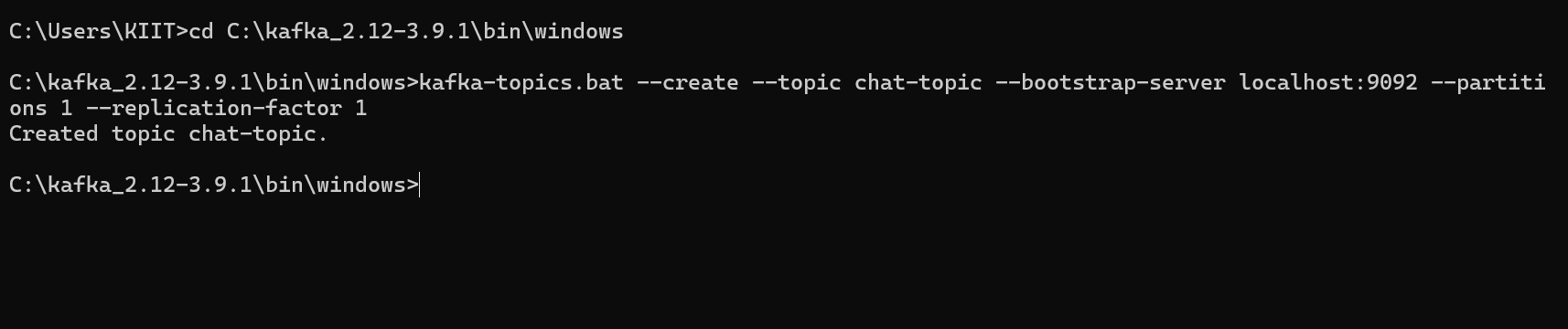
* Kafka is a distributed streaming platform.
* Built for scalability, durability, and fault tolerance.
* Commonly used for real-time data pipelines, streaming analytics, log aggregation, etc.

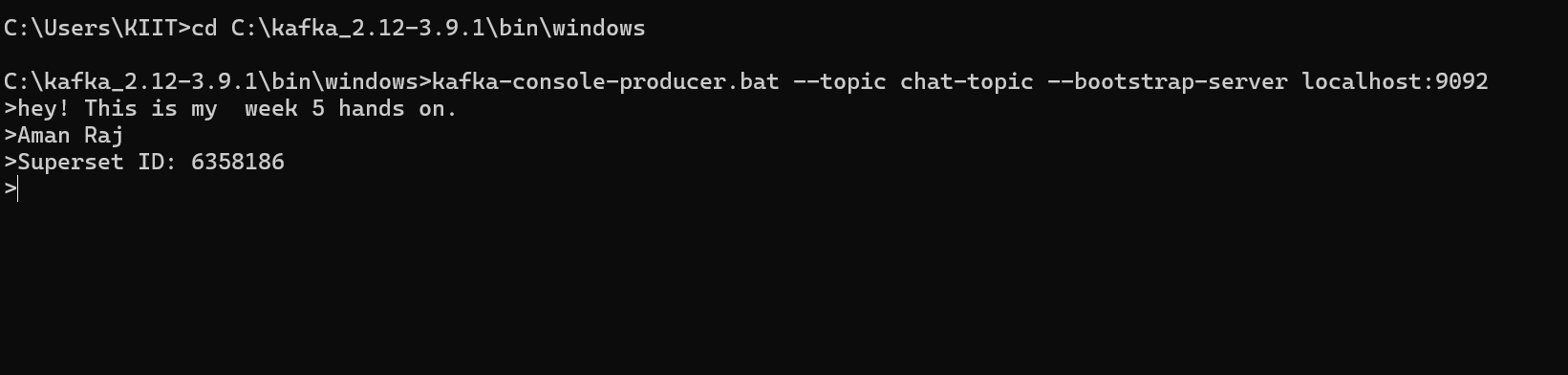
**Kafka Architecture**

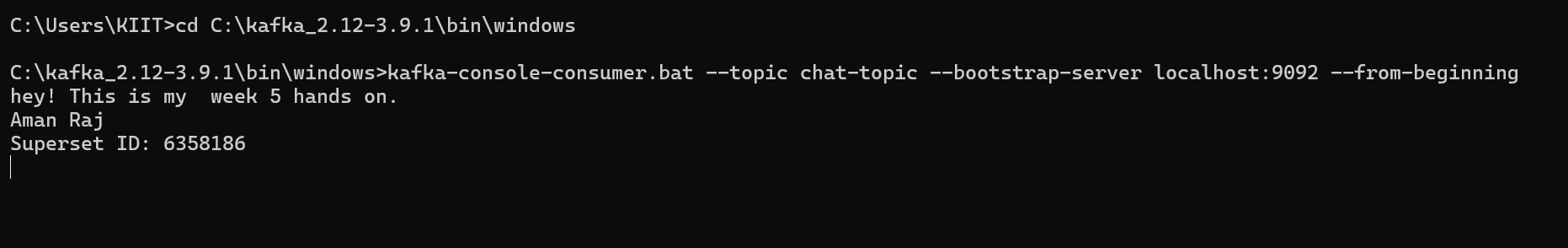
* Producer: Sends messages to Kafka.
* Consumer: Reads messages from Kafka.
* Broker: A Kafka server that stores messages.
* Zookeeper: Manages brokers, leader election, and cluster metadata.

1. **Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.**

**Outputs:-**

**Kafka Server Running:**  
  
  
  
  
  
**Zookeeper Running:**  
  
  
  
  
  
  
**Topic Created:**

**Creating Publisher in Command Prompt:**

**Client Application(Consumer):**

**2.Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

**Form1.cs**using System;

using System.Threading.Tasks;

using System.Windows.Forms;

using Confluent.Kafka;

namespace KafkaWindowsChatApp

{

public partial class Form1 : Form

{

private readonly string topic = "chat-topic";

private readonly string bootstrapServers = "localhost:9092";

private IConsumer<Ignore, string> consumer;

private Task consumerTask;

private bool running = true;

public Form1()

{

InitializeComponent();

StartConsumer();

}

private void btnSend\_Click(object sender, EventArgs e)

{

string message = txtMessage.Text.Trim();

if (!string.IsNullOrEmpty(message))

{

ProduceMessage(message);

lstChat.Items.Add("You: " + message);

txtMessage.Clear();

}

}

private async void ProduceMessage(string message)

{

var config = new ProducerConfig { BootstrapServers = bootstrapServers };

using (var producer = new ProducerBuilder<Null, string>(config).Build())

{

try

{

await producer.ProduceAsync(topic, new Message<Null, string> { Value = message });

}

catch (ProduceException<Null, string> ex)

{

MessageBox.Show($"Send failed: {ex.Message}");

}

}

}

private void StartConsumer()

{

var config = new ConsumerConfig

{

BootstrapServers = bootstrapServers,

GroupId = Guid.NewGuid().ToString(),

AutoOffsetReset = AutoOffsetReset.Earliest

};

consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe(topic);

consumerTask = Task.Run(() =>

{

while (running)

{

try

{

var cr = consumer.Consume();

Invoke((MethodInvoker)(() =>

{

lstChat.Items.Add("Other: " + cr.Message.Value);

}));

}

catch (Exception)

{

// Ignore exception on shutdown

}

}

});

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

running = false;

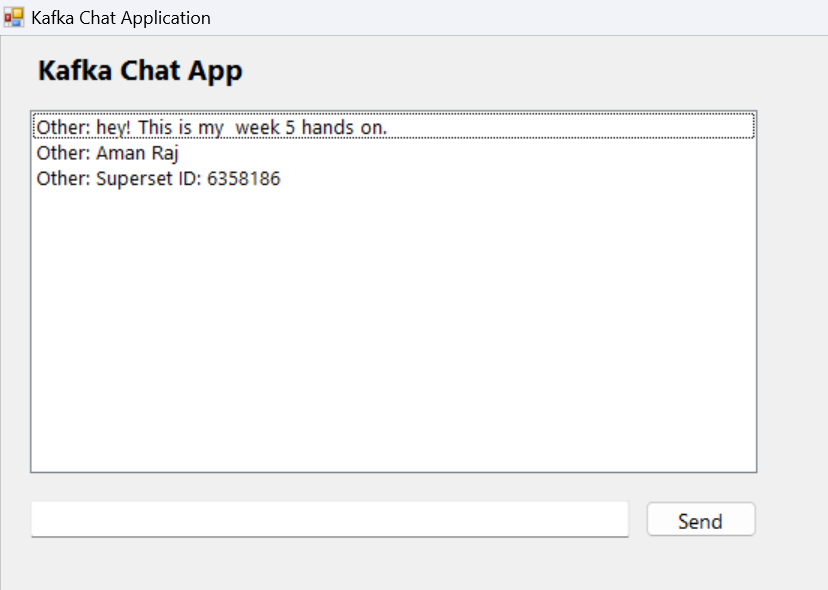
consumer.Close();

consumer.Dispose();

}

}

}

**Output  
  
**