Conducted By Kalyan Citizens' Education Society

B. K. Birla College of Arts, Science & Commerce, Kalyan

(Empowered Autonomous Status)

Affiliated to University of Mumbai

ISO 9001: 2015 Certified | Reaccredited by NAAC (4th Cycle) with 'A++' Grade (CGPA – 3.51)

B. K. Birla College Campus Road, Kalyan (West) - 421301

CERTIFICATE

This is to certify that **Mr./Ms.** ______________________ **Roll Number/Seat Number** _____________ a student of **Programme** ____________ has successfully completed the practical/project of the **Course** _________________ for **Semester** ________ under the valuable guidance of ______________________ during the **A.Y.** __________.

Date:

Place:

Internal Examiner **External Examiner**

Seal

INDEX

Programme: ______________ Semester: ____________

Course: ____

Sr. No.	**Prac. No.**	**Particulars**	**Date**	**Teacher Signature**
1.	1.			
2.	2.			
3.	3.			
4.	4.			
5.	5.			
6.	6.			
7.	7.			
8.	8.			

Case Study 1: AI-Powered Chatbots on Cloud for Customer Support Automation

Objective:

To analyze the architecture, implementation, and business impact of deploying AI-powered chatbots on cloud platforms. This case study explores how services like **AWS Lex**, **Google Dialogflow**, and **Azure Bot Service** leverage Natural Language Processing (NLP) and serverless computing to create scalable and intelligent customer support solutions.

Background:

In today's digital-first landscape, customers expect immediate, 24/7 support. Traditional support models, relying on human agents, are often costly and struggle to scale during peak demand. AI-powered chatbots address this challenge by automating responses to frequently asked questions and handling routine tasks. By hosting these bots on the cloud, companies can offload complex infrastructure management, benefit from pay-as-you-go pricing, and integrate with a vast ecosystem of cloud services.

Technical Architecture Deep Dive:

A modern cloud chatbot architecture is a decoupled, event-driven system composed of several managed services.

1. **Frontend/Client Interface:** This is the user-facing component. It can be a web chat widget (using JavaScript), a mobile application (iOS/Android), or an integration with third-party platforms like Slack, WhatsApp, or Facebook Messenger.
2. **API Gateway:** Services like **Amazon API Gateway** or **Azure API Management** act as the front door for all incoming user requests. It provides security, throttling, and routes requests to the appropriate backend service.
3. **NLP/NLU Engine (The "Brain"):** This is the core AI service that processes the user's text.
   - **AWS Lex, Google Dialogflow, or Azure LUIS (Language Understanding)** are used here.
   - **Natural Language Processing (NLP):** The service first processes the raw text.
   - **Natural Language Understanding (NLU):** It then performs two key tasks:
     - **Intent Recognition:** Identifies the user's goal (e.g., CheckOrderStatus, ResetPassword).
     - **Entity Extraction:** Pulls out key pieces of information (e.g., order_id: "12345", username: "john.doe").
4. **Business Logic Layer (Serverless Functions):** Once the intent and entities are identified, the NLP service triggers a backend function to execute the required action.
   - **AWS Lambda**, **Google Cloud Functions**, or **Azure Functions** are perfect for this.
   - This function contains the custom code to, for instance, query a database, call an external API, or perform a calculation.
5. **Data Layer (Cloud Database):** The backend function often needs to interact with a database to retrieve or store information.
   - **NoSQL Databases** like **Amazon DynamoDB** or **Google Firestore** are commonly used for their high speed and scalability, which is ideal for storing conversation logs or user data.
   - **Relational Databases** like **Amazon RDS** or **Azure SQL Database** might be used to access existing business data, such as customer order information.

6. **Response Generation:** The backend function formats a response, which is sent back through the API Gateway to the user on the frontend.

Workflow Example: Checking an Order Status:

User: "What is the status of my order number 98765?"

1. **Frontend:** The chat UI sends the user's message as a JSON payload to the API Gateway endpoint.
2. **API Gateway:** Authenticates the request and forwards it to the NLP Engine (e.g., Dialogflow).
3. **NLP Engine (Dialogflow):**
    - Recognizes the **Intent** as CheckOrderStatus.
    - Extracts the **Entity** orderNumber with the value "98765".
    - Triggers the associated serverless function (e.g., a Google Cloud Function).
4. **Logic Layer (Cloud Function):**
    - The function receives the intent (CheckOrderStatus) and the entity (orderNumber: "98765").
    - It executes code to query the company's Orders Database (e.g., an RDS instance) using the provided order number.
    - The database returns the status: "Shipped".
5. **Response Generation:** The function constructs a user-friendly response string: "Your order #98765 has been shipped and is on its way!"
6. **Return Path:** The response is sent back through the API Gateway to the user's chat window.

Advantages

- **24/7 Availability:** Provides instant, round-the-clock automated support without human intervention.
- **Hyper-Scalability:** Leverages serverless functions and managed services that automatically scale from zero to thousands of requests per second.
- **Cost Reduction:** Dramatically reduces the cost per interaction compared to human agents, freeing them up for more complex issues.
- **Data-Driven Insights:** Captures and analyzes user queries to identify common pain points, popular products, or gaps in service.
- **Omnichannel Integration:** A single chatbot backend can be easily deployed across multiple platforms (web, mobile, social media, voice assistants).
- **Consistent Service:** Ensures every customer receives the same quality and accuracy of information, aligning with brand guidelines.

Challenges and Considerations

- **Handling Ambiguity:** Chatbots can struggle with complex, ambiguous, or slang-filled user queries. A robust fallback mechanism to escalate to a human agent is crucial.
- **State Management:** Maintaining conversation context (remembering what was said earlier) can be complex and requires careful design.
- **Security:** As chatbots often handle sensitive user data, ensuring secure authentication, authorization, and data encryption is paramount.
- **Maintenance Overhead:** Intents, entities, and response scripts must be continuously updated and trained as business processes and customer queries evolve.

Example in Practice: Domino’s Pizza

The Domino’s Pizza chatbot ("Dom") is a prime example of cloud AI in action. Built with **Google Dialogflow**, it allows customers to order pizzas through natural conversation via text, social media, or voice.

- **Intent:** OrderPizza
- **Entities:** pizzaType (Pepperoni), size (Large), crust (Thin Crust), and address (User's saved address).
- **Backend Logic:** The Dialogflow agent integrates with Domino's backend ordering system via APIs to place the order, process payment, and provide a real-time delivery tracker link.

Conclusion

Cloud-based AI chatbots represent a fundamental shift in customer service, moving from a reactive, human-dependent model to a proactive, automated one. By leveraging the power of managed NLP services and serverless computing, organizations can build intelligent, scalable, and cost-effective solutions that significantly enhance customer experience and drive operational efficiency.

Case Study 2: AI Document Analysis using Cloud OCR (Invoice Extraction)

Objective

To investigate how cloud-based AI services for Optical Character Recognition (OCR) and document analysis, such as **AWS Textract**, **Azure Form Recognizer**, and **Google Document AI**, can automate the extraction of structured data from documents like invoices, receipts, and forms.

Background

Finance, logistics, healthcare, and legal departments in most organizations are inundated with vast quantities of semi-structured documents. The manual process of reading these documents and keying data into systems like ERPs or databases is slow, expensive, and prone to human error. Intelligent Document Processing (IDP) systems built on cloud AI services provide a scalable and accurate solution, transforming unstructured document images into structured, machine-readable data.

Technical Architecture Deep Dive

This process is typically implemented as an event-driven, serverless pipeline on the cloud, ensuring scalability and cost-efficiency.

1. **Ingestion Point (Cloud Storage):** Documents (e.g., PDF, JPG, PNG) are uploaded to a secure, scalable object storage service.
   - **Services: Amazon S3**, **Azure Blob Storage**, **Google Cloud Storage**.
   - An upload event (e.g., S3:ObjectCreated:*) is configured to automatically trigger the next step in the pipeline.
2. **Processing Trigger (Serverless Function):** A lightweight, serverless function is initiated by the storage event. Its role is to orchestrate the analysis.
   - **Services: AWS Lambda**, **Azure Functions**, **Google Cloud Functions**.
   - This function receives the document's location (e.g., bucket name and file key) as input.
3. **Intelligent OCR & Data Extraction (The "Brain"):** The serverless function calls the core AI document analysis service. This is more advanced than traditional OCR because it understands document structure.
   - **Services: AWS Textract**, **Azure Form Recognizer**, **Google Document AI**.
   - **Functionality:**
     - **Raw Text OCR:** Extracts all plain text from the document.
     - **Forms Extraction:** Identifies and extracts data from key-value pairs (e.g., "Invoice Number": "INV-1234").
     - **Table Extraction:** Recognizes tabular data and preserves the row and column structure.
     - **Handwriting Recognition:** Can often process handwritten text in addition to printed text.
   - The service returns a structured JSON object containing the extracted data, confidence scores, and bounding box coordinates for each piece of information.

4. **Post-Processing & Validation (Logic Layer):** The same serverless function (or a subsequent one) receives the JSON output from the AI service.
    - **Parsing:** It parses the JSON to isolate critical fields (e.g., vendor name, total amount, due date).
    - **Validation:** It can perform business rule checks, such as verifying the date format or ensuring the sum of line items matches the total amount.
    - **Enrichment:** It might enrich the data by cross-referencing the vendor name with an existing database to get a vendor ID.
5. **Data Persistence (Cloud Database):** Once validated, the clean, structured data is stored in a suitable database.
    - **NoSQL Databases (e.g., Amazon DynamoDB):** Ideal for storing the flexible JSON output for quick retrieval.
    - **Relational Databases (e.g., Amazon Aurora/RDS):** Used when the data needs to be integrated into existing structured systems like an ERP or accounting platform.
6. **Visualization and Integration (Downstream Systems):** The structured data is now ready for use.
    - **Dashboards:** Tools like **Power BI**, **Tableau**, or **Amazon QuickSight** can connect to the database to visualize spending trends, payment cycles, etc.
    - **ERP/Billing Systems:** The data can be programmatically pushed into systems like SAP, Oracle, or QuickBooks to automate the accounts payable process.

Workflow Example: Processing a Supplier Invoice

An accounts payable clerk uploads a supplier's PDF invoice into a designated Amazon S3 bucket.

1. **Ingestion:** The PDF file lands in the invoices-pending S3 bucket.
2. **Trigger:** An S3 PutObject event automatically invokes an **AWS Lambda** function.
3. **Extraction:**
    - The Lambda function calls the **AWS Textract** StartDocumentAnalysis API, pointing to the location of the PDF in S3.
    - Textract processes the document, identifying the invoice number, line items in a table, and the final amount due.
    - It returns a detailed JSON object with all this structured information.
4. **Post-Processing:**
    - The Lambda function parses the JSON, extracting the values for vendor_name, invoice_id, due_date, and total_amount.
    - It checks if total_amount is a valid number and if due_date is a valid date.
5. **Persistence:** The function then writes a new item to a **DynamoDB** table called Invoices, containing the clean, validated data.
6. **Integration:** A separate process (or another event trigger on the DynamoDB table) picks up the new record and automatically creates a draft bill in the company's accounting software via an API call.

Advantages

- **Eliminates Manual Data Entry:** Drastically reduces the time, cost, and errors associated with manual keying.
- **Accelerated Business Cycles:** Speeds up processes like accounts payable, claims processing, and customer onboarding from days to minutes.
- **High Accuracy & Confidence:** AI models provide confidence scores for each extracted field, allowing for a human-in-the-loop review only for low-confidence results.
- **Scalability:** The serverless architecture effortlessly scales to process thousands of documents per hour during peak periods without any infrastructure management.
- **Creates Searchable Archives:** Digitized documents and their metadata can be easily indexed, making them fully searchable for compliance and auditing purposes.

Challenges and Considerations

- **Template Variation:** The models work best with consistent layouts. Performance can degrade if invoices from different vendors have wildly different formats. Custom models may need to be trained for specific, complex documents.
- **Image Quality:** The accuracy of OCR is highly dependent on the quality of the source document (e.g., resolution, lighting, skew, handwritten notes).
- **Cost Management:** While cost-effective, processing millions of pages can become expensive. It's important to monitor API usage and optimize the workflow.
- **Data Privacy:** Documents may contain sensitive PII (Personally Identifiable Information). It's crucial to implement proper security controls, encryption, and access policies within the cloud environment.

Conclusion

Cloud-based Intelligent Document Processing is a transformative technology that allows organizations to unlock the value trapped in their unstructured documents. By building automated pipelines with services like AWS Textract or Azure Form Recognizer, companies can achieve significant gains in operational efficiency, data accuracy, and business agility, ultimately freeing up human capital to focus on higher-value tasks.

Case Study 3: Real-Time Cloud Analytics for Smart Cities

Objective

To examine the end-to-end cloud architecture required to build a real-time analytics platform for smart city management. This case study focuses on how the integration of IoT (Internet of Things) devices, streaming data services, and AI/ML models on the cloud enables proactive urban management and data-driven decision-making.

Background

Urbanization is leading to increasingly complex challenges in traffic management, public safety, resource allocation, and environmental sustainability. Smart cities leverage a network of IoT sensors—embedded in traffic lights, public utilities, waste bins, and environmental monitors—to collect vast amounts of real-time data. By harnessing cloud AI platforms, city planners and operators can move from a reactive to a predictive model, anticipating issues like traffic bottlenecks or air quality degradation before they become critical.

Technical Architecture Deep Dive

A smart city analytics platform is a multi-layered, highly available system designed to handle high-velocity data streams.

1. **Sensing & Edge Layer:** This layer consists of the physical IoT devices distributed across the city.
    - **Devices:** GPS sensors on buses, video cameras for traffic monitoring, air quality sensors, smart water meters, and smart lighting controllers.
    - **Edge Gateways:** These devices aggregate data from multiple sensors locally, perform initial filtering or compression, and securely transmit it to the cloud using efficient protocols like **MQTT** or **HTTP/S**.
2. **Ingestion & Messaging Layer:** This is the secure entry point for all IoT data into the cloud.
    - **Services: AWS IoT Core**, **Azure IoT Hub**, or **Google Cloud IoT Core**.
    - **Functionality:** These managed services can handle millions of device connections, provide device authentication and authorization, and offer a rules engine to route incoming data to different cloud services.
3. **Stream Processing Layer (The "Hot Path"):** Data that requires immediate analysis flows through this real-time pipeline.
    - **Services: Amazon Kinesis**, **Azure Stream Analytics**, or **Google Cloud Dataflow**.
    - **Functionality:** These services can perform real-time data transformation, aggregation, and filtering on the fly. For example, they can calculate the average traffic speed over a 1-minute window or detect an anomalous spike in pollution levels.
4. **AI / Machine Learning Layer:** The processed real-time data is fed into machine learning models for predictive insights.
    - **Services: Amazon SageMaker**, **Azure Machine Learning**, or **Google AI Platform**.
    - **Model Types:**
        - **Forecasting:** Time-series models to predict traffic congestion 30 minutes in advance.

        - **Anomaly Detection:** Identifying unusual patterns, like a sudden drop in water pressure indicating a pipe leak.
        - **Classification:** Analyzing camera feeds to classify available vs. occupied parking spots.

5. **Data Storage Layer:** The platform utilizes a dual-approach to data storage.
    - **Hot Storage (Real-Time):** Data for immediate dashboarding is stored in a high-speed, low-latency database. Examples include **Amazon Timestream** (a time-series database) or **Redis**.
    - **Cold Storage (Long-Term):** All raw and processed data is archived in a cost-effective data lake for historical analysis and training future ML models. Services include **Amazon S3**, **Azure Data Lake Storage**, or **Google Cloud Storage**.
6. **Presentation & Action Layer:** This is where insights are consumed and acted upon.
    - **Dashboards:** Real-time dashboards built with **Amazon QuickSight**, **Microsoft Power BI**, or **Grafana** display key metrics for city operators.
    - **Alerting:** Automated alerts are triggered based on predefined rules (e.g., air quality index > 150). Services like **Amazon Simple Notification Service (SNS)** can send notifications via email, SMS, or to a mobile app.
    - **Actuation:** In advanced systems, insights can trigger automated actions, such as dynamically changing traffic light timings to ease congestion.

Workflow Example: Dynamic Traffic Management

A traffic accident occurs on a major city highway during rush hour.

1. **Sensing:** Roadside cameras and embedded magnetic loop sensors detect a sudden, complete stop in traffic flow in one lane.
2. **Ingestion:** The data (video snippets and sensor readings showing zero velocity) is sent via an edge gateway to **AWS IoT Core**.
3. **Processing:** IoT Core's rules engine forwards the data to **Amazon Kinesis**, which identifies this pattern as a high-priority anomaly.
4. **AI Analysis:** The anomaly data is fed to a pre-trained **SageMaker** model, which confirms the event is an accident (not just a traffic jam) and predicts a 90% probability of severe congestion spreading to adjacent routes within 15 minutes.
5. **Action & Presentation:**
    - An automated alert is sent via **Amazon SNS** to the traffic management center and emergency services.
    - The operator's real-time dashboard in **QuickSight** immediately shows the affected road segment in red.
    - The system automatically triggers a command to update electronic road signs on feeder roads, suggesting alternative routes to drivers.

Advantages

- **Proactive Decision-Making:** Enables authorities to anticipate and mitigate problems, rather than just reacting to them.
- **Optimized Resource Allocation:** Data-driven insights help optimize public transport schedules, waste collection routes, and energy distribution.
- **Enhanced Public Safety:** Faster emergency response times through real-time incident detection and improved infrastructure monitoring.
- **Environmental Sustainability:** Helps manage pollution levels, conserve water, and reduce the city's carbon footprint through smart energy grids.
- **Scalability and Reliability:** The cloud infrastructure can seamlessly scale to accommodate an increasing number of sensors and data loads.

Challenges and Considerations

- **Data Privacy & Security:** Collecting city-wide data raises significant privacy concerns. Strong anonymization, encryption, and strict access control policies are essential.
- **High Initial Investment:** Deploying thousands of sensors and building the cloud infrastructure requires significant upfront investment.
- **Data Interoperability:** Integrating data from a wide variety of sensors and systems from different manufacturers can be a major technical hurdle.
- **Network Reliability:** The entire system depends on a reliable communication network (e.g., 5G, LoRaWAN) to transmit data from sensors to the cloud.

Conclusion

Cloud AI is the engine that powers the modern smart city. By providing a scalable and intelligent platform for IoT data analytics, it transforms raw sensor readings into actionable intelligence. This allows municipal governments to create safer, more efficient, and more sustainable urban environments for their citizens, fundamentally changing the way cities are managed.

Case Study 4: AI-Based Healthcare Diagnostics on Cloud

Objective

To analyze the architecture and impact of cloud-hosted artificial intelligence models in medical diagnostics. This case study explores how cloud platforms facilitate the entire lifecycle—from data ingestion and model training to secure clinical deployment—of AI tools that assist healthcare professionals in detecting diseases from medical imagery.

Background

Radiology and pathology are cornerstone diagnostic fields that rely on the interpretation of complex visual data (X-rays, CT scans, MRIs, tissue slides). The global shortage of specialists, combined with an increasing volume of scans, creates significant workload pressure and can lead to diagnostic delays or errors. Cloud AI offers a powerful solution by providing a secure, scalable infrastructure to train and deploy deep learning models. These models act as an assistive tool for clinicians, helping to prioritize critical cases, improve detection accuracy, and provide quantitative analysis. Critically, these systems must operate within strict regulatory frameworks like **HIPAA** (in the US) to ensure patient data privacy and security.

Technical Architecture Deep Dive

Building a clinical-grade AI diagnostic tool involves a compliant and robust cloud architecture.

1. **Secure Data Ingestion & Anonymization:**
    - **Source:** Medical images, typically in **DICOM** format, are exported from a hospital's **PACS** (Picture Archiving and Communication System) or EMR (Electronic Medical Record) system.
    - **Anonymization:** Before being uploaded, a crucial preprocessing step is performed to strip all Protected Health Information (PHI) from the DICOM metadata to comply with privacy regulations.
    - **Cloud Storage:** The anonymized images are uploaded to a secure, compliant object storage service like **Amazon S3** (using S3 Object Lock for immutability) or **Azure Blob Storage** with HIPAA-compliant configurations.
2. **Data Lake & Labeling:**
    - The anonymized images are stored in a centralized data lake. This repository becomes the single source of truth for model training.
    - **Data Labeling:** To train a supervised learning model, a dataset must be labeled by medical experts. Services like **Amazon SageMaker Ground Truth** or **Google Cloud's Data Labeling Service** provide workflows where radiologists can draw bounding boxes around tumors or classify images (e.g., 'pneumonia present' vs. 'normal').
3. **Model Training, Tuning, & Validation (The "Brain"):**
    - **Model Choice: Convolutional Neural Networks (CNNs)** are the state-of-the-art models for image analysis tasks like classification, object detection, and segmentation.
    - **Training Platform:** Managed services like **Amazon SageMaker**, **Azure Machine Learning**, or **Google's Vertex AI** are used to train these complex models on powerful GPU instances. These platforms handle the underlying infrastructure, allowing data scientists to focus on the model itself.

    - **Validation:** The model is rigorously tested against a separate, unseen dataset to validate its accuracy, sensitivity, and specificity before it can be considered for deployment.

4. **Secure Deployment & Inference:**
    - **API Endpoint:** Once validated, the trained model is deployed as a highly available, scalable, and secure API endpoint using services like **Amazon SageMaker Endpoints** or a containerized application on **Azure Kubernetes Service (AKS)**.
    - **Inference:** For a new patient case, the anonymized medical image is sent to this endpoint via a secure API call. The model processes the image and returns its prediction (e.g., a probability score of malignancy) in a structured format like JSON, usually within seconds.
5. **Clinical Integration & Presentation Layer:**
    - The model's output is not a final diagnosis but a piece of decision-support information for the clinician.
    - **Dashboard:** The results, sometimes including a "heatmap" that visually highlights suspicious regions on the image, are displayed on a secure web-based dashboard for the doctor to review.
    - **PACS/EMR Integration:** In advanced setups, the AI results are sent back and integrated directly into the hospital's native radiology software (PACS), appearing as a secondary analysis alongside the original image.

Workflow Example: AI-Assisted Lung Nodule Detection

A radiologist is reviewing a patient's chest CT scan for signs of lung cancer.

1. **Image Upload:** The anonymized CT scan is securely uploaded from the hospital's PACS to a designated S3 bucket.
2. **Trigger & Inference:** The upload triggers an **AWS Lambda** function, which calls the deployed **SageMaker** model endpoint, passing the location of the new scan.
3. **AI Analysis:** The CNN model analyzes hundreds of images in the CT scan series and detects a small, suspicious nodule (8mm) that could be an early-stage tumor.
4. **Result Generation:** The model returns a JSON response containing:
    - The probability of malignancy (e.g., 0.85).
    - The coordinates and slice number of the detected nodule.
    - A link to a generated heatmap image.
5. **Clinical Review:**
    - The case is automatically flagged and moved to the top of the radiologist's worklist.
    - When the radiologist opens the case in their viewer, they see the original CT scan alongside the AI's findings: the highlighted nodule and the 85% malignancy score.
    - The radiologist uses this information to pay closer attention to the flagged area, ultimately confirming the finding and recommending a follow-up biopsy. The AI acted as a vigilant "second reader."

Advantages

- **Improved Diagnostic Accuracy & Speed:** Acts as a second pair of eyes for clinicians, reducing the risk of missed findings and accelerating the review process.
- **Early Disease Detection:** AI models can often detect subtle patterns that are imperceptible to the human eye, enabling earlier diagnosis and better patient outcomes.
- **Workload Triage:** Automatically prioritizes critical cases (e.g., brain hemorrhage, pulmonary embolism), ensuring that the most urgent patients receive immediate attention.
- **Democratization of Expertise:** Makes expert-level analysis available to smaller clinics or in remote areas that may not have on-site specialists.
- **Scalable and Secure:** Cloud platforms provide the necessary compute power for training large models and offer robust security and compliance (e.g., HIPAA, GDPR) for handling sensitive medical data.

Challenges and Considerations

- **Regulatory Approval:** AI diagnostic tools are considered medical devices and require stringent validation and approval from regulatory bodies like the **FDA** in the US or an **CE mark** in Europe.
- **Data Bias and Generalizability:** A model trained primarily on data from one demographic may perform poorly on others. Ensuring diverse and representative training data is a major ethical and technical challenge.
- **The "Black Box" Problem:** It can be difficult to understand exactly *why* a deep learning model made a particular prediction, which can be a barrier to clinical trust and adoption.
- **Integration with Hospital IT:** Integrating the AI solution seamlessly into existing and often legacy hospital IT systems (like PACS and EMRs) is complex.

Conclusion

Cloud-based AI is poised to revolutionize medical diagnostics by empowering clinicians with powerful, data-driven insights. While significant challenges around regulation, ethics, and integration remain, these tools have the potential to enhance diagnostic accuracy, improve efficiency, and ultimately make high-quality healthcare more accessible and effective on a global scale.

Practical 1: Building a Cloud Chatbot using Amazon Lex

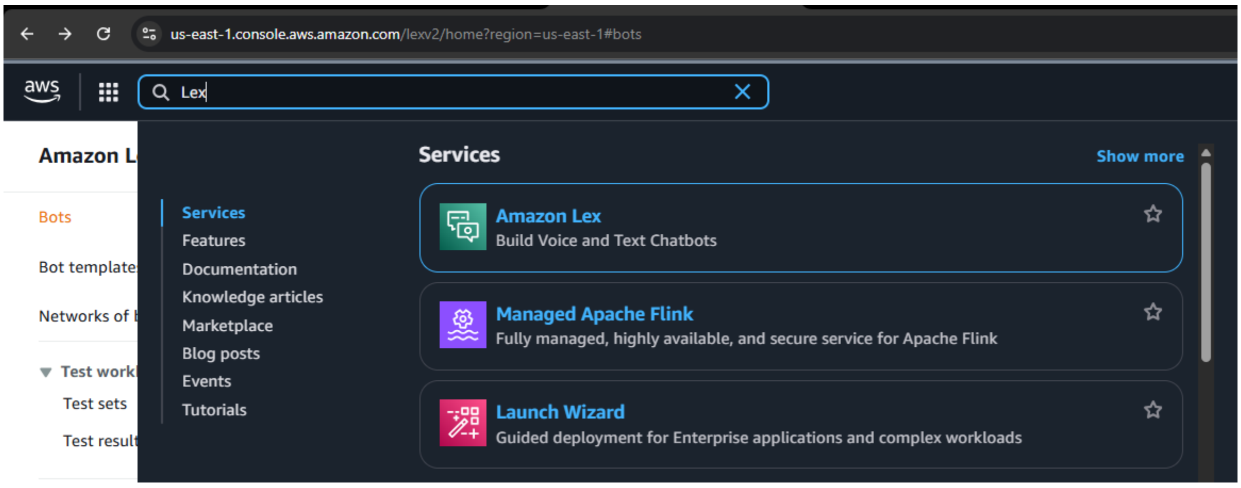
Aim: To create and test a basic conversational chatbot using Amazon Lex to handle simple, automated customer support queries.

Tools Used:

- AWS Management Console
- **Amazon Lex**
- **AWS Lambda** (Optional, for advanced logic)

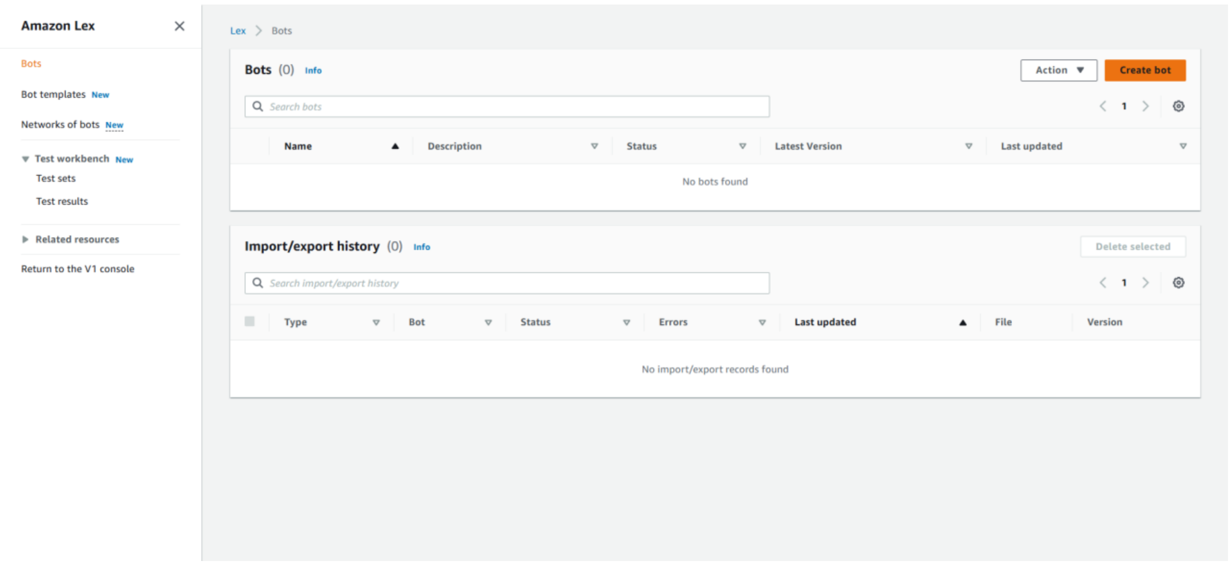
Procedure:

1. **Sign in** to the AWS Management Console and navigate to the **Amazon Lex** service.

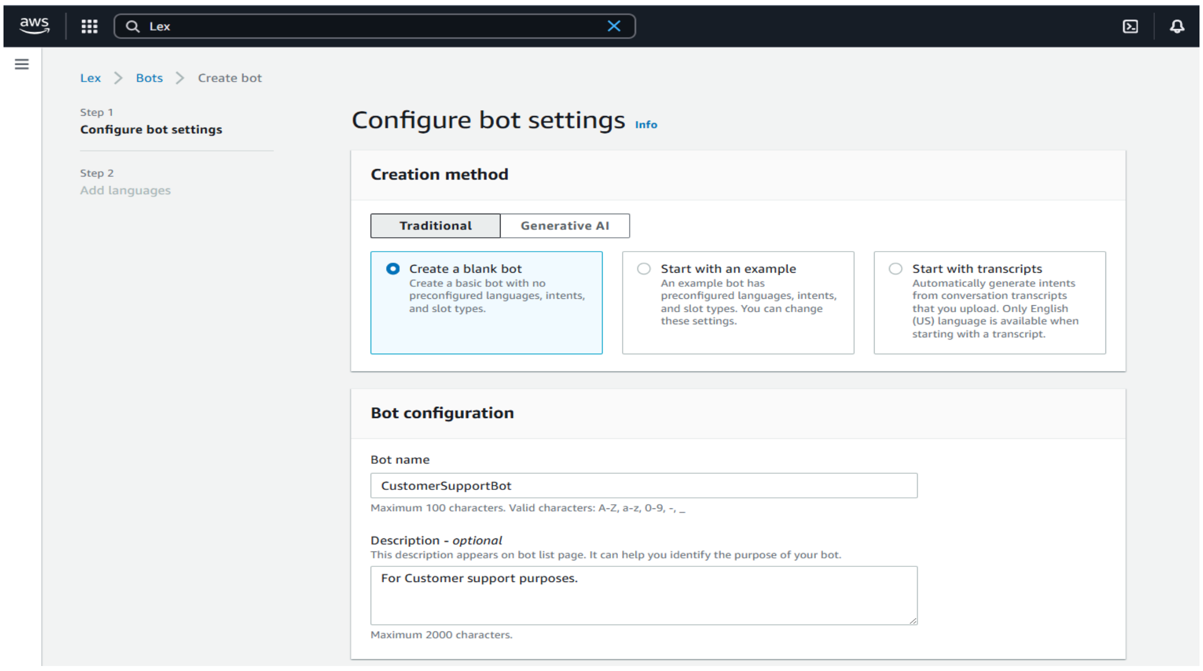


us-east-1.console.aws.amazon.com/lexv2/home?region=us-east-1#bots
aws
Lex
Amazon L
Bots
Bot template
Networks of b
Test work
Test sets
Test result
Services
Features
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Services
Show more
Amazon Lex
Build Voice and Text Chatbots
Managed Apache Flink
Fully managed, highly available, and secure service for Apache Flink
Launch Wizard
Guided deployment for Enterprise applications and complex workloads

2. Click "**Create bot**" and choose the "Create a blank bot" option.



3. **Name your bot** (e.g., CustomerSupportBot) and configure the required IAM permissions.


Lex > Bots > Create bot

Step 1
Configure bot settings

Step 2
Add languages

Configure bot settings Info

Creation method

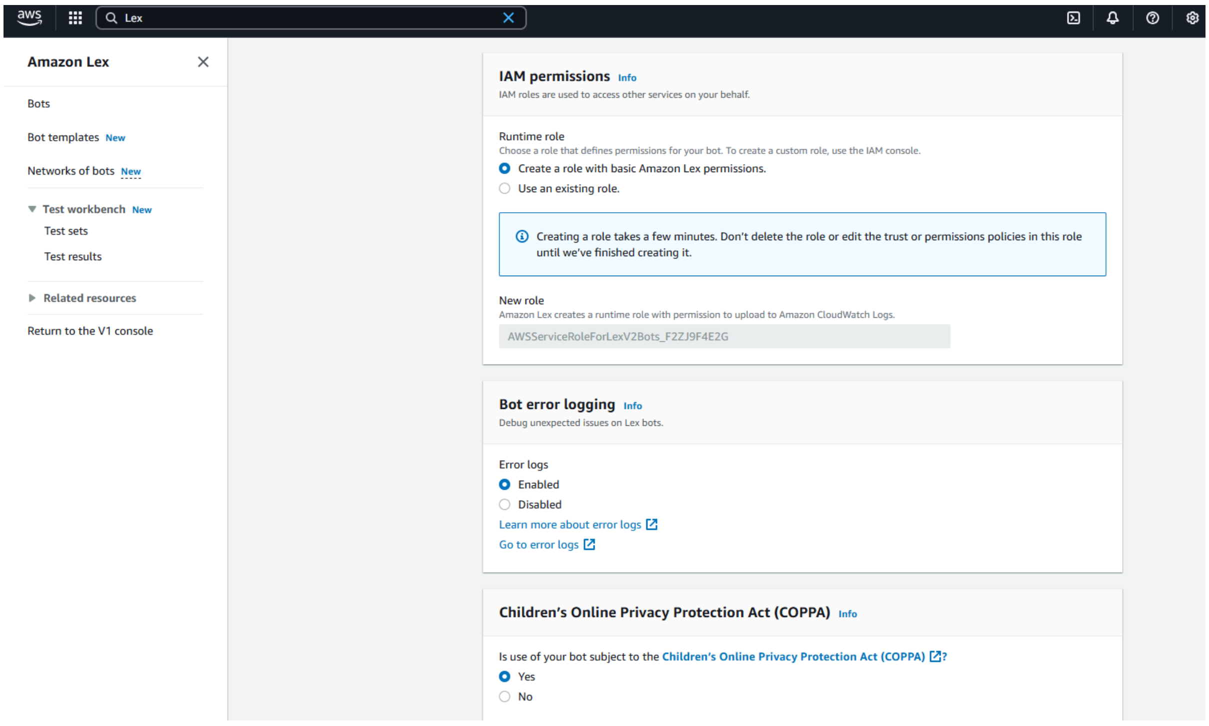
Traditional | Generative AI

- [✓ Create a blank bot] — Create a basic bot with no preconfigured languages, intents, and slot types.
- Start with an example — An example bot has preconfigured languages, intents, and slot types. You can change these settings.
- Start with transcripts — Automatically generate intents from conversation transcripts that you upload. Only English (US) language is available when starting with a transcript.

Bot configuration

Bot name
CustomerSupportBot
Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Description - *optional*
This description appears on bot list page. It can help you identify the purpose of your bot.
For Customer support purposes.
Maximum 2000 characters.



IAM permissions Info
IAM roles are used to access other services on your behalf.

Runtime role
Choose a role that defines permissions for your bot. To create a custom role, use the IAM console.
- [✓ Create a role with basic Amazon Lex permissions.]
- Use an existing role.

Creating a role takes a few minutes. Don't delete the role or edit the trust or permissions policies in this role until we've finished creating it.

New role
Amazon Lex creates a runtime role with permission to upload to Amazon CloudWatch Logs.
AWSServiceRoleForLexV2Bots_F2ZJ9F4E2G

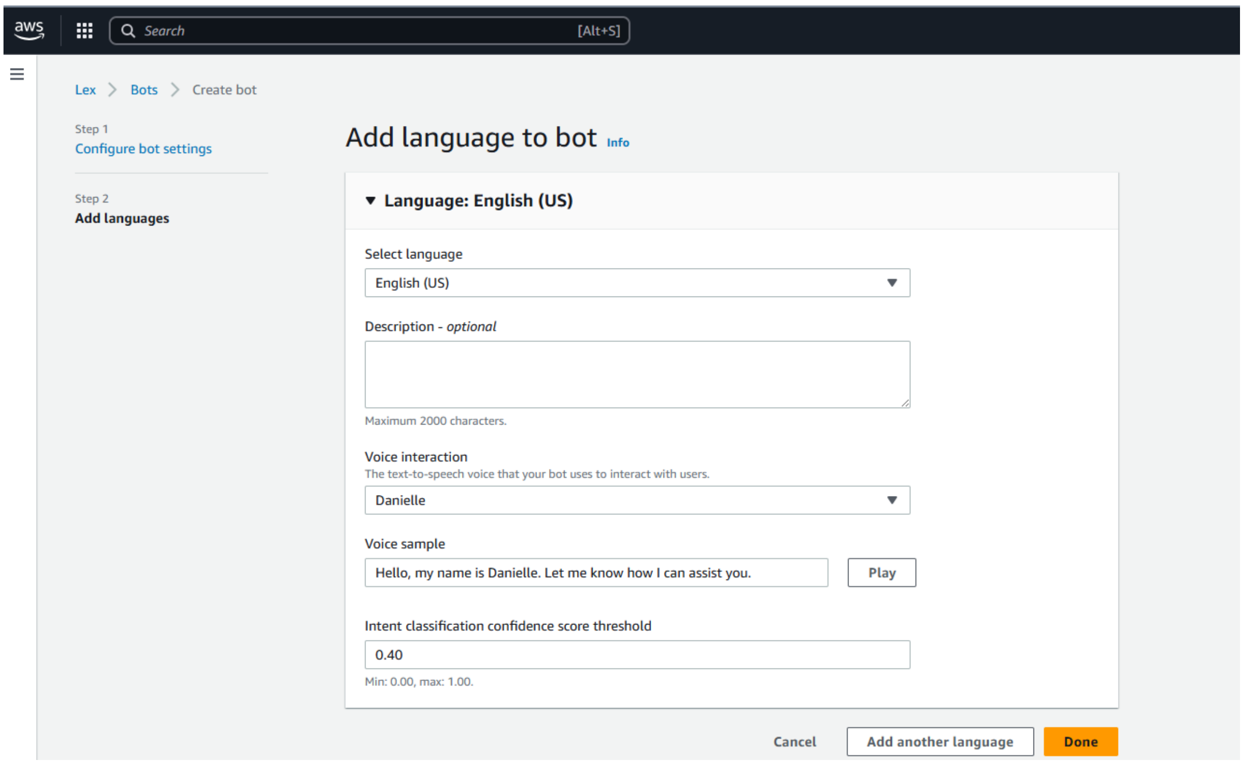
Bot error logging Info
Debug unexpected issues on Lex bots.

Error logs
- [✓ Enabled]
- Disabled

Learn more about error logs
Go to error logs

Children's Online Privacy Protection Act (COPPA) Info

Is use of your bot subject to the Children's Online Privacy Protection Act (COPPA)?
- [✓ Yes]
- No


Lex > Bots > Create bot

Step 1
Configure bot settings

Step 2
Add languages

Add language to bot Info

Language: English (US)

Select language
English (US)

Description - *optional*

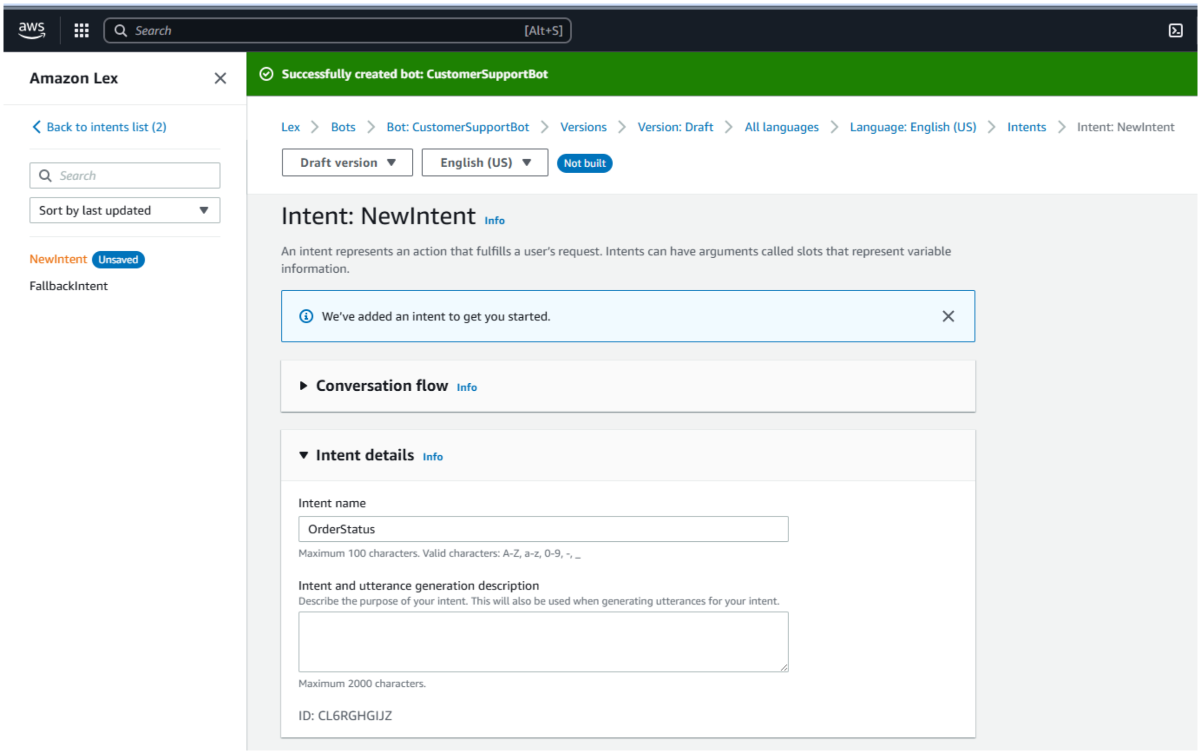
Maximum 2000 characters.

Voice interaction
The text-to-speech voice that your bot uses to interact with users.
Danielle

Voice sample
Hello, my name is Danielle. Let me know how I can assist you.

Intent classification confidence score threshold
0.40
Min: 0.00, max: 1.00.

4. Once the bot is created, add a new **intent**. Name it something like OrderStatus.


Successfully created bot: CustomerSupportBot

Amazon Lex
Back to intents list (2)
Sort by last updated
NewIntent Unsaved
FallbackIntent

Lex > Bots > Bot: CustomerSupportBot > Versions > Version: Draft > All languages > Language: English (US) > Intents > Intent: NewIntent

Draft version | English (US) | Not built

Intent: NewIntent Info

An intent represents an action that fulfills a user's request. Intents can have arguments called slots that represent variable information.

We've added an intent to get you started.

Conversation flow Info

Intent details Info

Intent name
OrderStatus
Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

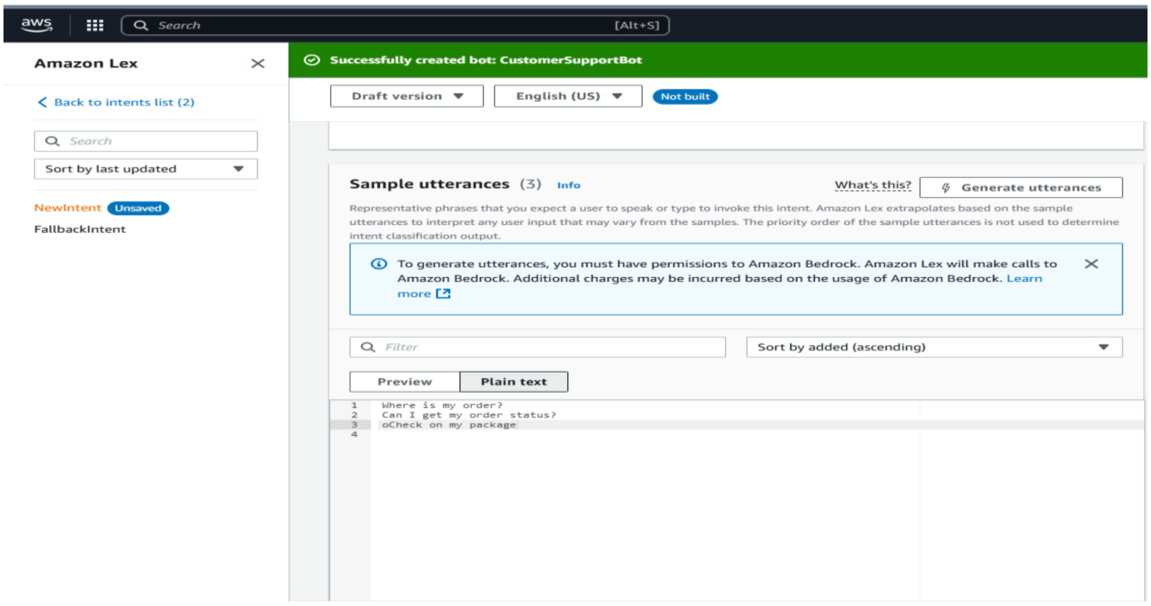
Intent and utterance generation description
Describe the purpose of your intent. This will also be used when generating utterances for your intent.

Maximum 2000 characters.

ID: CL6RGHGIJZ

5. In the intent configuration, add several **sample utterances** (training phrases) that a user might say, such as:

    - Where is my order?
    - Can I get my order status?
    - Check on my package



Successfully created bot: CustomerSupportBot

Sample utterances (3) Info

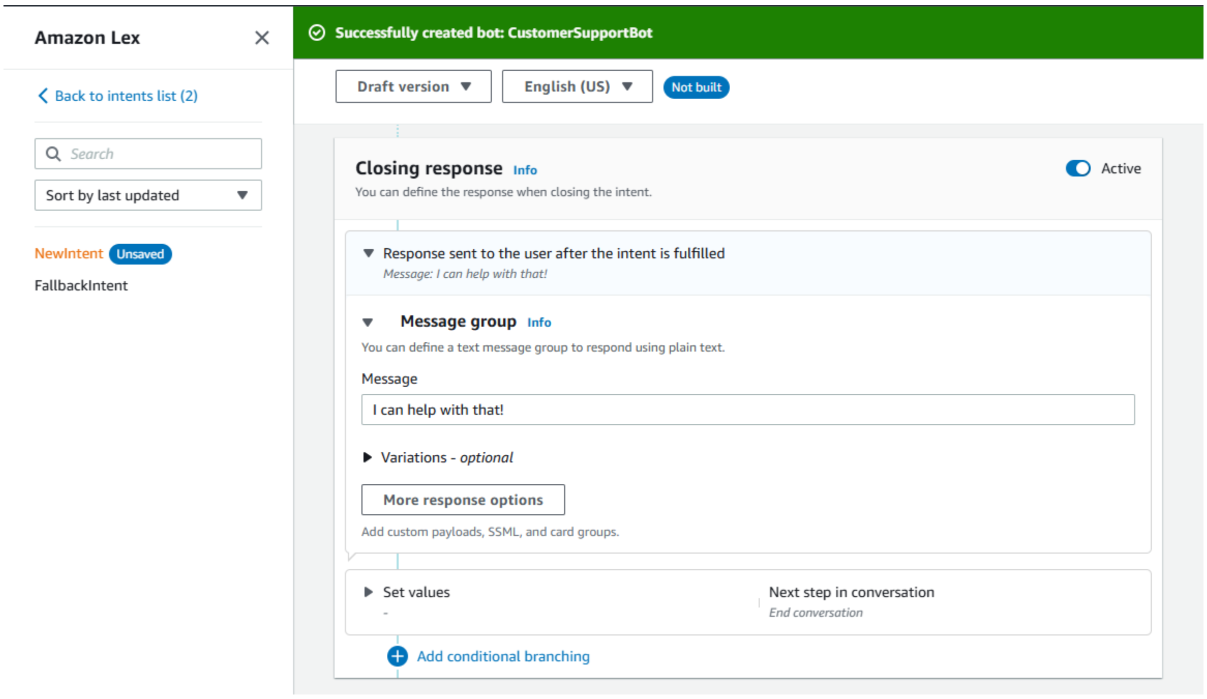
Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

To generate utterances, you must have permissions to Amazon Bedrock. Amazon Lex will make calls to Amazon Bedrock. Additional charges may be incurred based on the usage of Amazon Bedrock. Learn more

Sort by added (ascending)

```
Where is my order?
Can I get my order status?
oCheck on my package
```

6. Scroll down to the "**Closing response**" section and add a message that the bot will reply with, like I can help with that! Please provide your order ID to check its status.



Successfully created bot: CustomerSupportBot

Closing response Info

You can define the response when closing the intent.

Response sent to the user after the intent is fulfilled

Message: I can help with that!

Message group Info

You can define a text message group to respond using plain text.

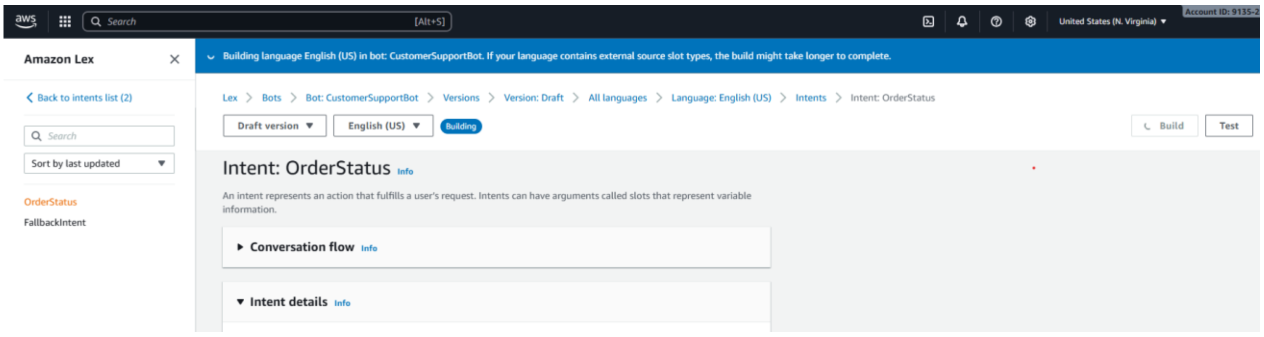
Message: I can help with that!

Add custom payloads, SSML, and card groups.

Set values: -

Next step in conversation: *End conversation*

7. **Save** the intent, and then click the "**Build**" button to train the bot's NLP model.

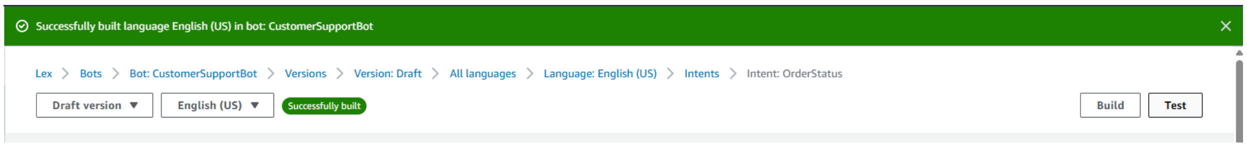


Building language English (US) in bot: CustomerSupportBot. If your language contains external source slot types, the build might take longer to complete.

Intent: OrderStatus Info

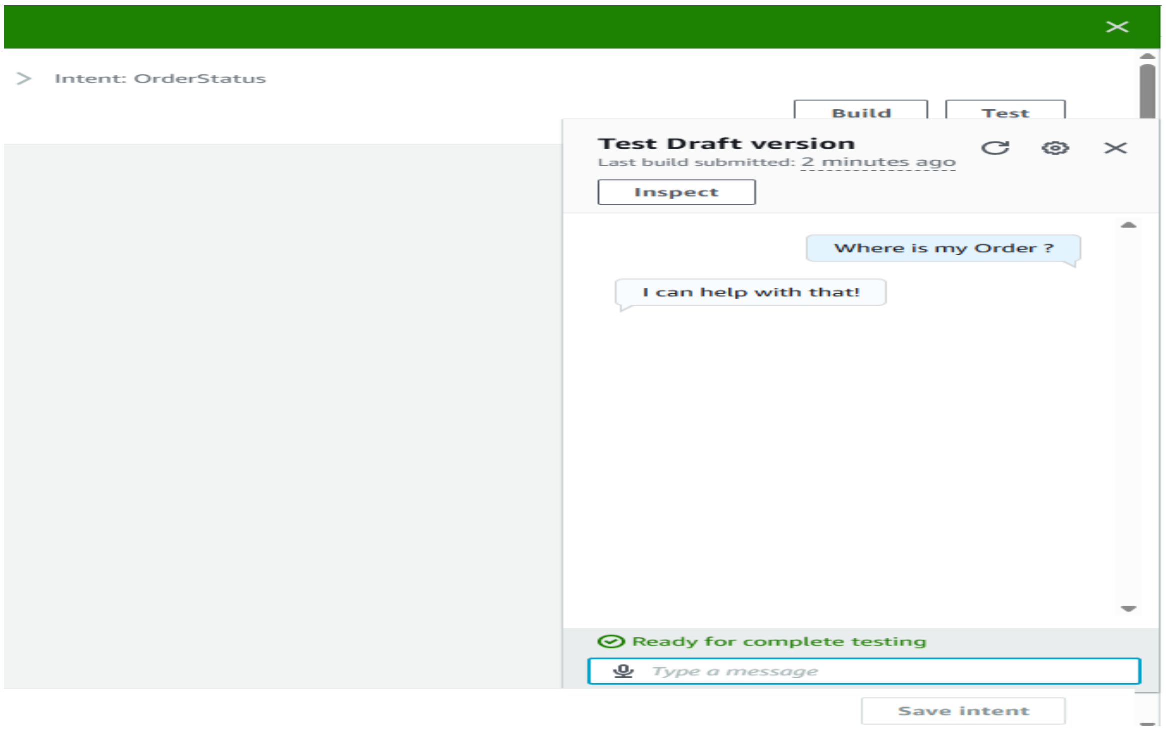
An intent represents an action that fulfills a user's request. Intents can have arguments called slots that represent variable information.

8. After the build is complete, use the "**Test**" window within the Lex console to interact with your bot and verify its responses.


Successfully built language English (US) in bot: CustomerSupportBot

Lex > Bots > Bot: CustomerSupportBot > Versions > Version: Draft > All languages > Language: English (US) > Intents > Intent: OrderStatus

Expected Output: When a user types a phrase like "Where is my order?" into the test chat window, the bot should respond with your predefined closing response, "I can help with that! Please provide your order ID to check its status."


> Intent: OrderStatus

Test Draft version

Last build submitted: 2 minutes ago

Where is my Order ?

I can help with that!

Ready for complete testing

Conclusion: Amazon Lex provides a powerful, fully managed service for building conversational AI interfaces using Natural Language Processing (NLP), allowing for the creation of sophisticated chatbots without deep AI expertise.

Practical 2: Text Extraction from Documents using AWS Textract

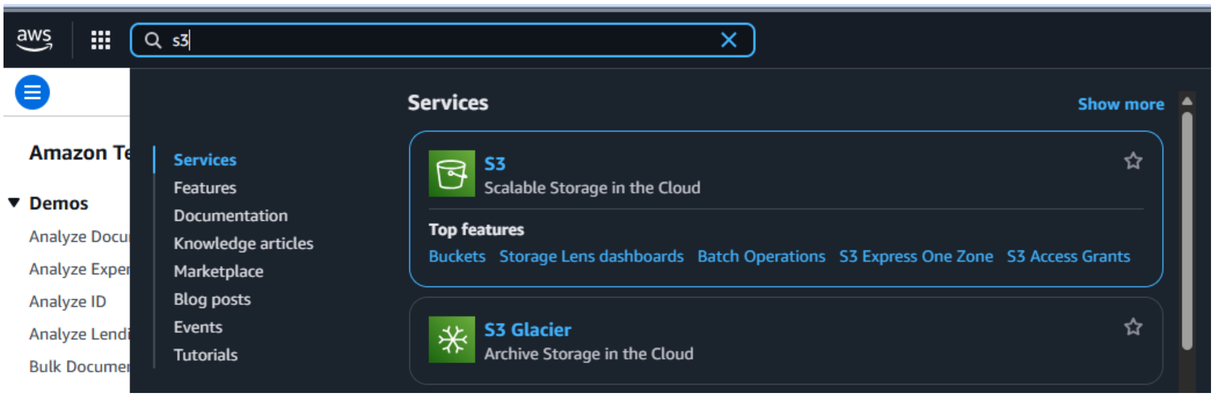
Aim: To use AWS Textract to automatically extract text, forms (key-value pairs), and tables from a scanned document like an invoice or a report.

Tools Used:

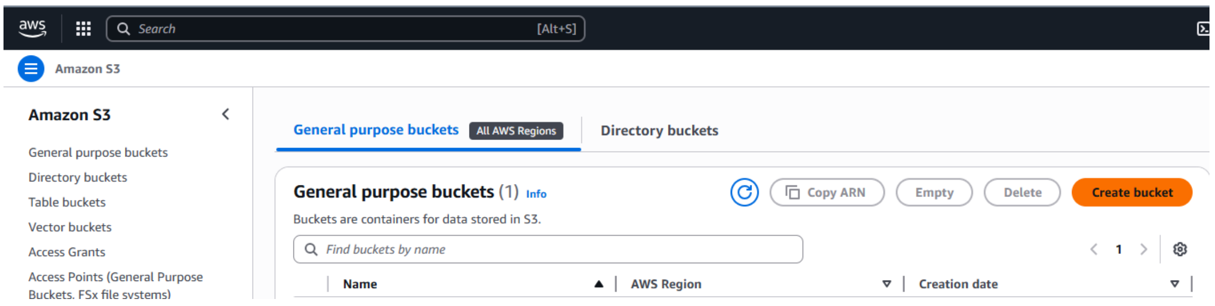
- AWS Management Console
- **Amazon S3** (Simple Storage Service)
- **AWS Textract**

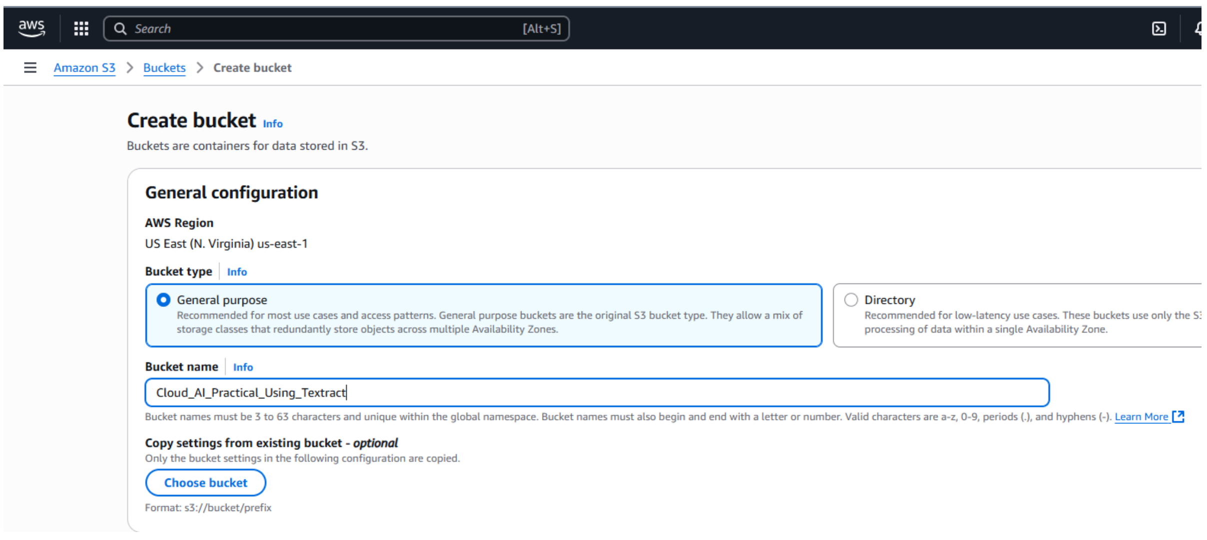
Procedure:

1. **Sign in** to the AWS Management Console and navigate to the **Amazon S3** service.



2. Create a new S3 bucket or use an existing one. **Upload** a sample document (e.g., an invoice in PDF, JPG, or PNG format) to your bucket.





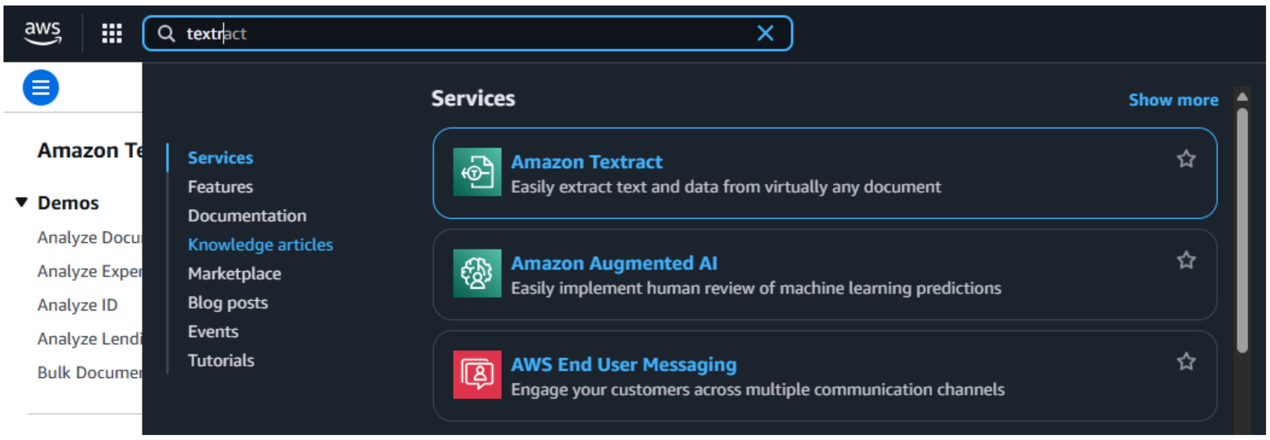
AWS Region
US East (N. Virginia) us-east-1

Bucket type
General purpose — Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory — Recommended for low-latency use cases. These buckets use only the S3 processing of data within a single Availability Zone.

Bucket name
Cloud_AI_Practical_Using_Textract
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Format: s3://bucket/prefix

3. Navigate to the **AWS Textract** service.



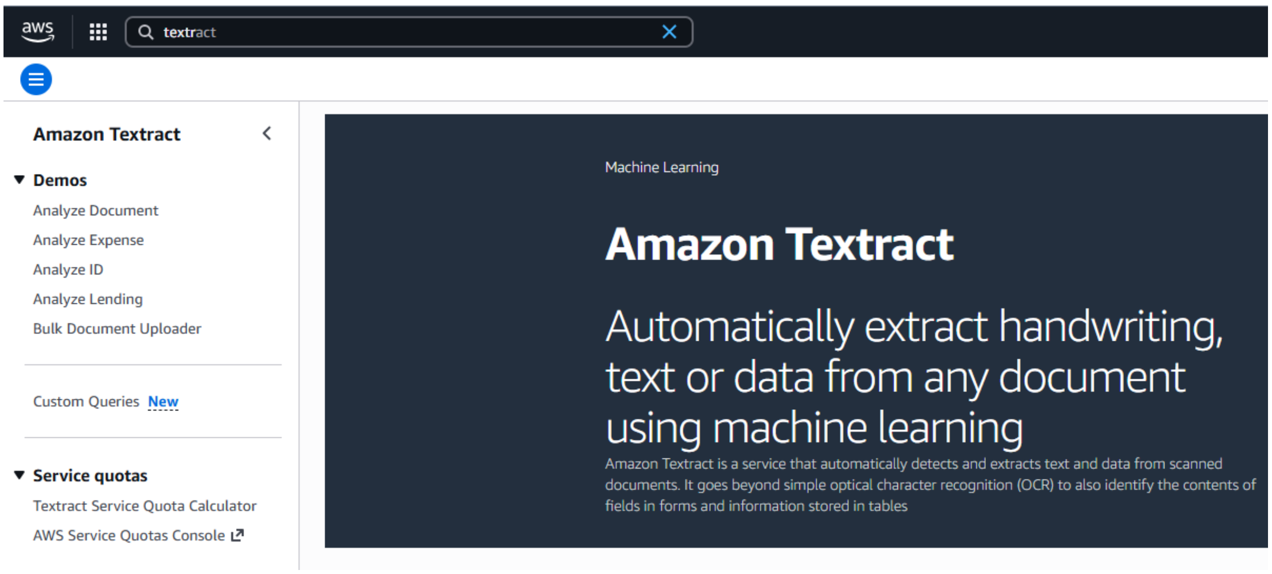
Services

Amazon Textract
Easily extract text and data from virtually any document

Amazon Augmented AI
Easily implement human review of machine learning predictions

AWS End User Messaging
Engage your customers across multiple communication channels

4. In the Textract console, select "**Analyze Document**" from the left-hand menu.



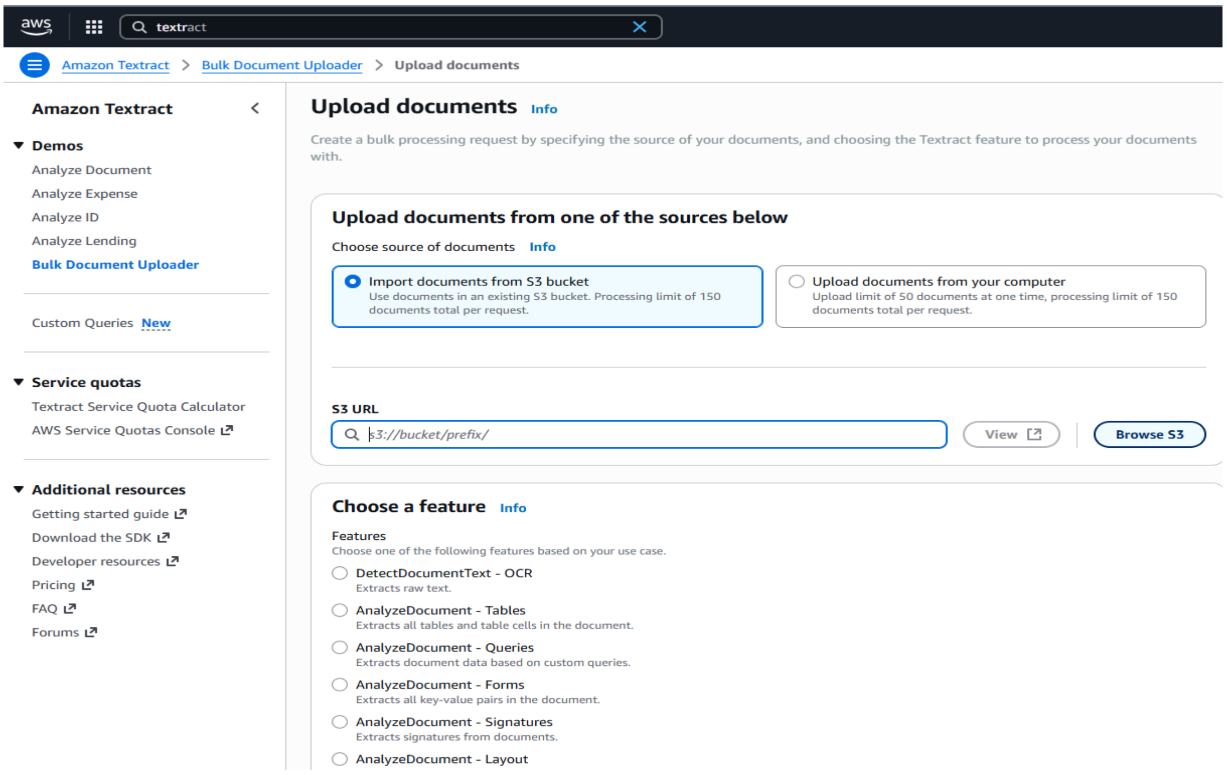
Machine Learning

Amazon Textract

Automatically extract handwriting, text or data from any document using machine learning

Amazon Textract is a service that automatically detects and extracts text and data from scanned documents. It goes beyond simple optical character recognition (OCR) to also identify the contents of fields in forms and information stored in tables

5. Choose the option to select a document from your Amazon S3 bucket and **browse** to the file you just uploaded.



Upload documents Info

Create a bulk processing request by specifying the source of your documents, and choosing the Textract feature to process your documents with.

Upload documents from one of the sources below

Choose source of documents Info

[✓ Import documents from S3 bucket] Use documents in an existing S3 bucket. Processing limit of 150 documents total per request.

Upload documents from your computer — Upload limit of 50 documents at one time, processing limit of 150 documents total per request.

S3 URL

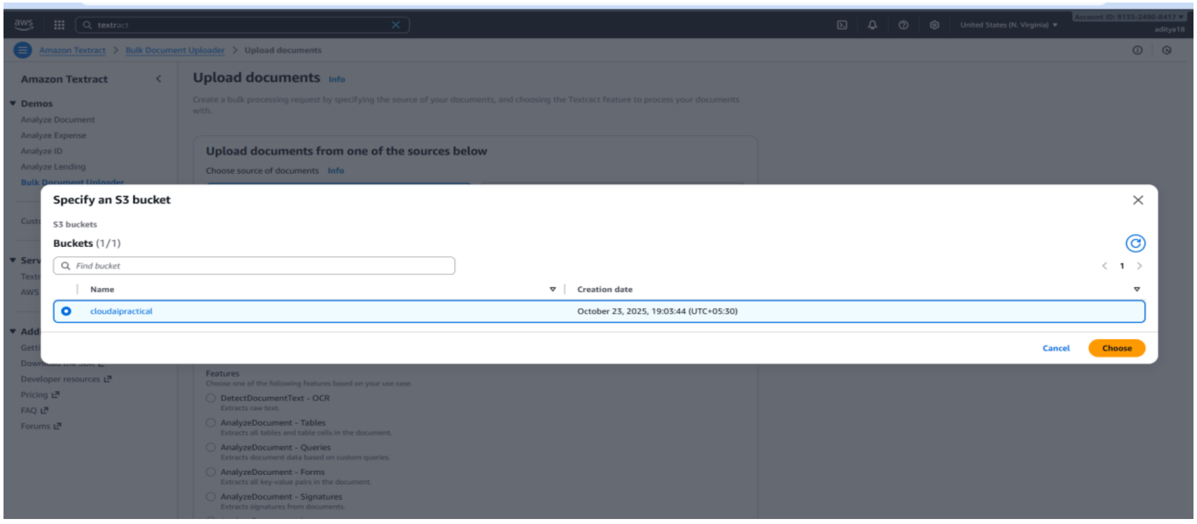
s3://bucket/prefix/

View | Browse S3

Choose a feature Info

Features
Choose one of the following features based on your use case.

- DetectDocumentText - OCR
  Extracts raw text.
- AnalyzeDocument - Tables
  Extracts all tables and table cells in the document.
- AnalyzeDocument - Queries
  Extracts document data based on custom queries.
- AnalyzeDocument - Forms
  Extracts all key-value pairs in the document.
- AnalyzeDocument - Signatures
  Extracts signatures from documents.
- AnalyzeDocument - Layout


Amazon Textract > Bulk Document Uploader > Upload documents

Upload documents Info

Create a bulk processing request by specifying the source of your documents, and choosing the Textract feature to process your documents with.

Upload documents from one of the sources below

Choose source of documents Info

Specify an S3 bucket

S3 buckets

Buckets (1/1)

Find bucket

	Name	Creation date
●	cloudaipractical	October 23, 2025, 19:03:44 (UTC+05:30)

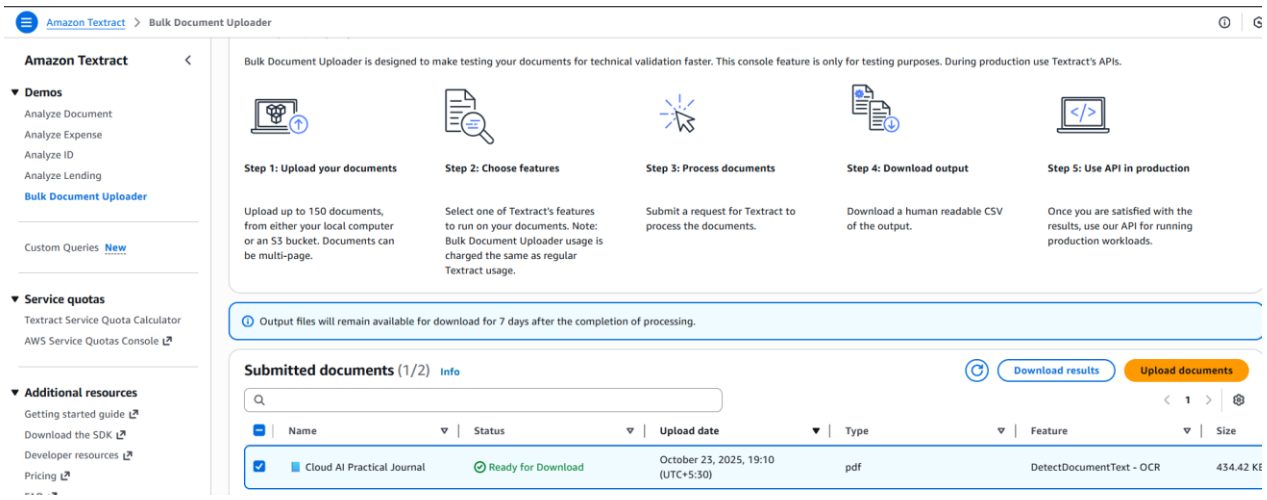
Cancel Choose

Features

Choose one of the following features based on your use case.

- DetectDocumentText - OCR
  Extracts raw text.
- AnalyzeDocument - Tables
  Extracts all tables and table cells in the document.
- AnalyzeDocument - Queries
  Extracts document data based on custom queries.
- AnalyzeDocument - Forms
  Extracts all key-value pairs in the document.
- AnalyzeDocument - Signatures
  Extracts signatures from documents.

6. Textract will automatically process the document. On the results screen, you can view the extracted **raw text**, **forms** (key-value pairs like "Invoice Number": "INV-101"), and **tables**.


Amazon Textract > Bulk Document Uploader

Bulk Document Uploader is designed to make testing your documents for technical validation faster. This console feature is only for testing purposes. During production use Textract's APIs.

Step 1: Upload your documents
Upload up to 150 documents, from either your local computer or an S3 bucket. Documents can be multi-page.

Step 2: Choose features
Select one of Textract's features to run on your documents. Note: Bulk Document Uploader usage is charged the same as regular Textract usage.

Step 3: Process documents
Submit a request for Textract to process the documents.

Step 4: Download output
Download a human readable CSV of the output.

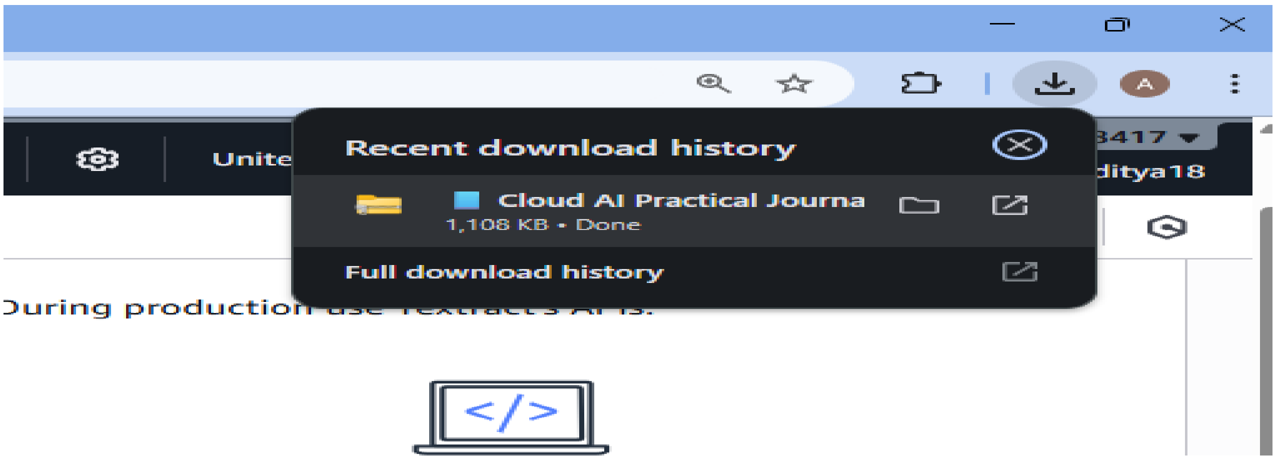
Step 5: Use API in production
Once you are satisfied with the results, use our API for running production workloads.

Output files will remain available for download for 7 days after the completion of processing.

Submitted documents (1/2) Info

Download results Upload documents

	Name	Status	Upload date	Type	Feature	Size
✓	Cloud AI Practical Journal	Ready for Download	October 23, 2025, 19:10 (UTC+5:30)	pdf	DetectDocumentText - OCR	434.42 K


Recent download history

Cloud AI Practical Journa
1,108 KB • Done

Full download history

During production use Textract's APIs.

Expected Output: The service will display the document's contents, correctly identifying and separating fields like "Invoice Number," "Due Date," "Total Amount," and any line items within a table.

Conclusion: AWS Textract is more than just OCR; it's an intelligent document processing service that understands document structure, enabling the automation of data entry and document analysis workflows.

Practical 3: Text Sentiment Analysis using AWS Comprehend

Aim

To perform sentiment and entity analysis on text data using **AWS Comprehend**, a cloud-based Natural Language Processing (NLP) service.

Objective

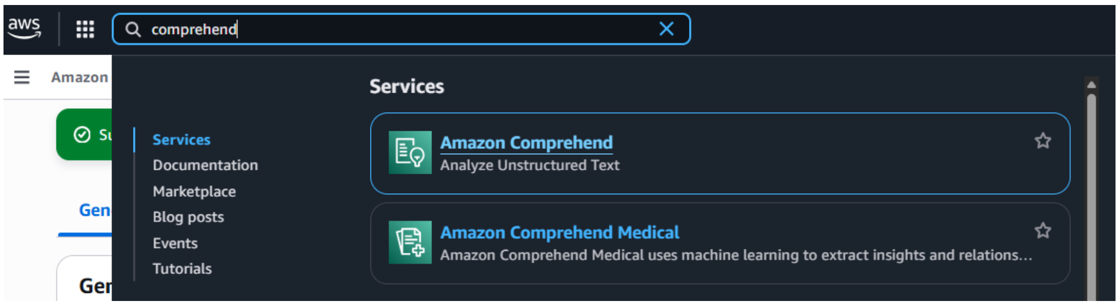
To understand how AWS Comprehend analyzes unstructured text and extracts useful information such as sentiment, entities, and key phrases through pre-trained AI models.

Tools / Services Used

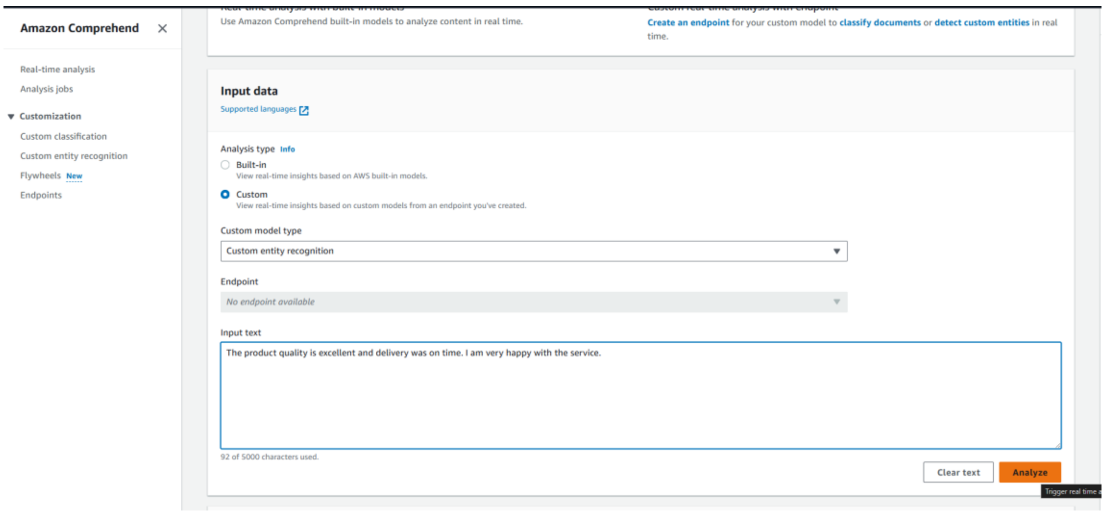
- AWS Management Console
- AWS Comprehend Service
- Sample Text Dataset

Detailed Procedure

1. **Login to AWS Console:**
   - Open https://aws.amazon.com/console and log in.
2. **Open the Comprehend Service:**
   - In the search bar, type **"Comprehend"** and open the service dashboard.


aws
comprehend
Amazon
Services
Services
Documentation
Marketplace
Blog posts
Events
Tutorials
Amazon Comprehend
Analyze Unstructured Text
Amazon Comprehend Medical
Amazon Comprehend Medical uses machine learning to extract insights and relations...

3. **Select the Analysis Type:**
   - Choose **"Sentiment Analysis"** to detect emotions in text.
   - Optionally, explore **"Entity Recognition"** and **"Key Phrases Extraction."**



Use Amazon Comprehend built-in models to analyze content in real time.

Create an endpoint for your custom model to classify documents or detect custom entities in real time.

Input data

Supported languages

Analysis type Info

Built-in
View real-time insights based on AWS built-in models.

[✓ Custom]
View real-time insights based on custom models from an endpoint you've created.

Custom model type: Custom entity recognition

Endpoint: No endpoint available

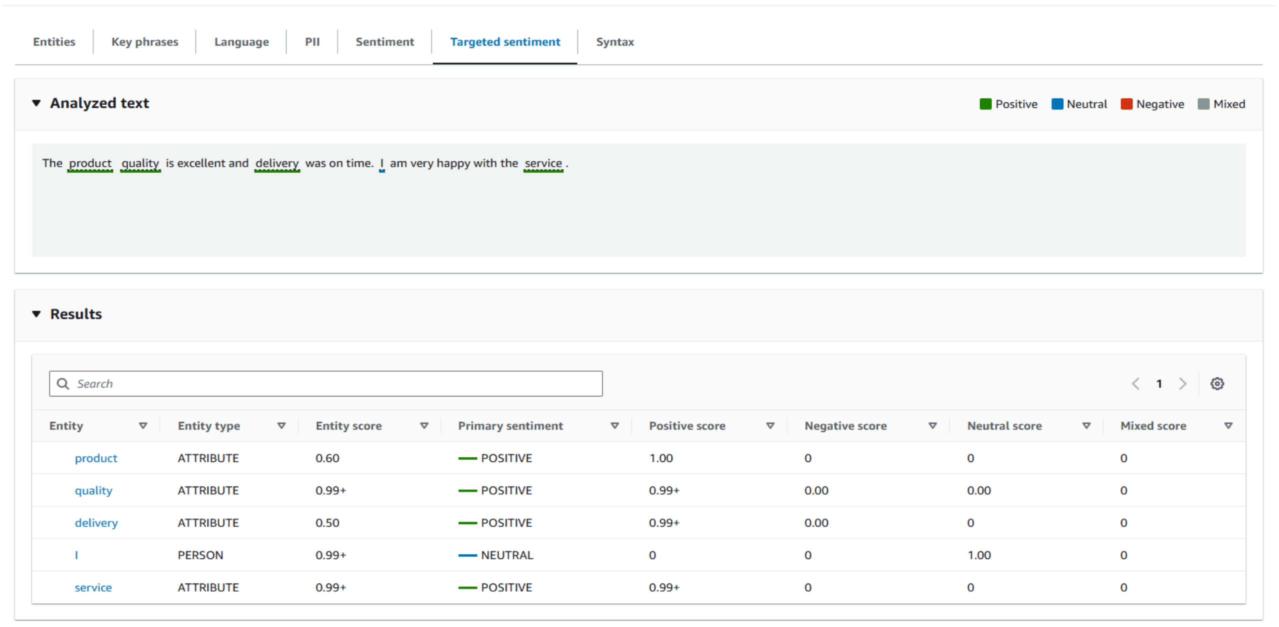
Input text: The product quality is excellent and delivery was on time. I am very happy with the service.

92 of 5000 characters used.

4. **Input Sample Text:**
   - Enter a paragraph such as:
     "The product quality is excellent and delivery was on time. I am very happy with the service."
5. **Run Analysis:**
   - Click on **"Analyze"** or **"Run"** to begin text processing.
   - AWS Comprehend uses its NLP models to evaluate sentiment and entities.

View Results:

- The output displays:
  - Overall Sentiment: *Positive*
  - Sentiment Scores: (Positive = 0.98, Negative = 0.01, Neutral = 0.01)
  - Key Entities: *Product, Delivery, Service*
  - Key Phrases: *Excellent quality, On time, Happy with service*



The product quality is excellent and delivery was on time. I am very happy with the service .

Entity	Entity type	Entity score	Primary sentiment	Positive score	Negative score	Neutral score	Mixed score
product	ATTRIBUTE	0.60	POSITIVE	1.00	0	0	0
quality	ATTRIBUTE	0.99+	POSITIVE	0.99+	0.00	0.00	0
delivery	ATTRIBUTE	0.50	POSITIVE	0.99+	0.00	0	0
I	PERSON	0.99+	NEUTRAL	0	0	1.00	0
service	ATTRIBUTE	0.99+	POSITIVE	0.99+	0	0	0

Expected Output

AWS Comprehend classifies text into sentiment categories and highlights entities and key phrases with corresponding confidence scores.

Conclusion

AWS Comprehend enables automatic sentiment and entity analysis without requiring custom NLP development. It can be applied in industries like e-commerce, customer service, and marketing to derive actionable insights from text data.

Practical 4: Image Object and Face Detection using AWS Rekognition

Aim

To analyze and detect objects, scenes, and faces in images using **AWS Rekognition**, an AI-based image and video analysis service.

Objective

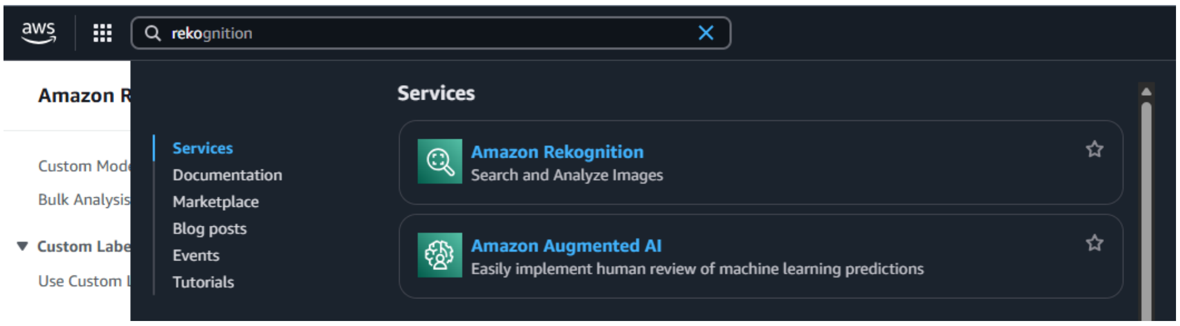
To understand how cloud-based computer vision systems perform object detection and facial recognition using pre-trained AI models.

Tools / Services Used

- AWS Management Console
- AWS Rekognition Service

Detailed Procedure (Step-by-Step)

1. **Login to AWS Console:**
   - Go to https://aws.amazon.com/console and sign in.
2. **Open Rekognition Service:**
   - Type **“Rekognition”** in the search bar and select the service.



3. **Select the Use Case:**
   - From the dashboard, choose **“Image Properties.”**

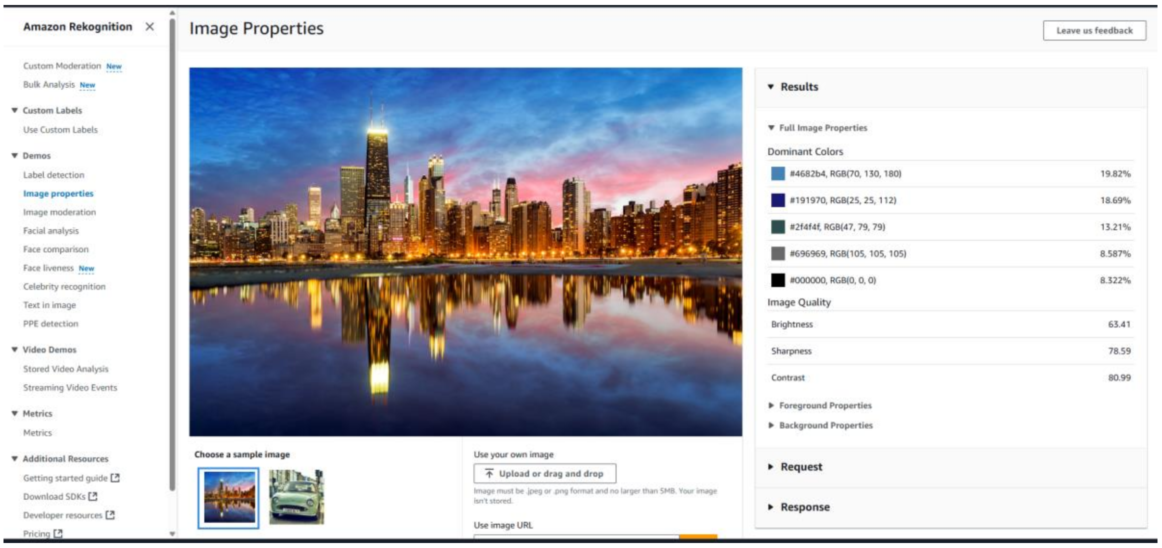


Image must be .jpeg or .png format and no larger than 5MB. Your image isn't stored.

Results

Full Image Properties

Dominant Colors

#4682b4, RGB(70, 130, 180)	19.82%
#191970, RGB(25, 25, 112)	18.69%
#2f4f4f, RGB(47, 79, 79)	13.21%
#696969, RGB(105, 105, 105)	8.587%
#000000, RGB(0, 0, 0)	8.322%

Image Quality

Brightness	63.41
Sharpness	78.59
Contrast	80.99

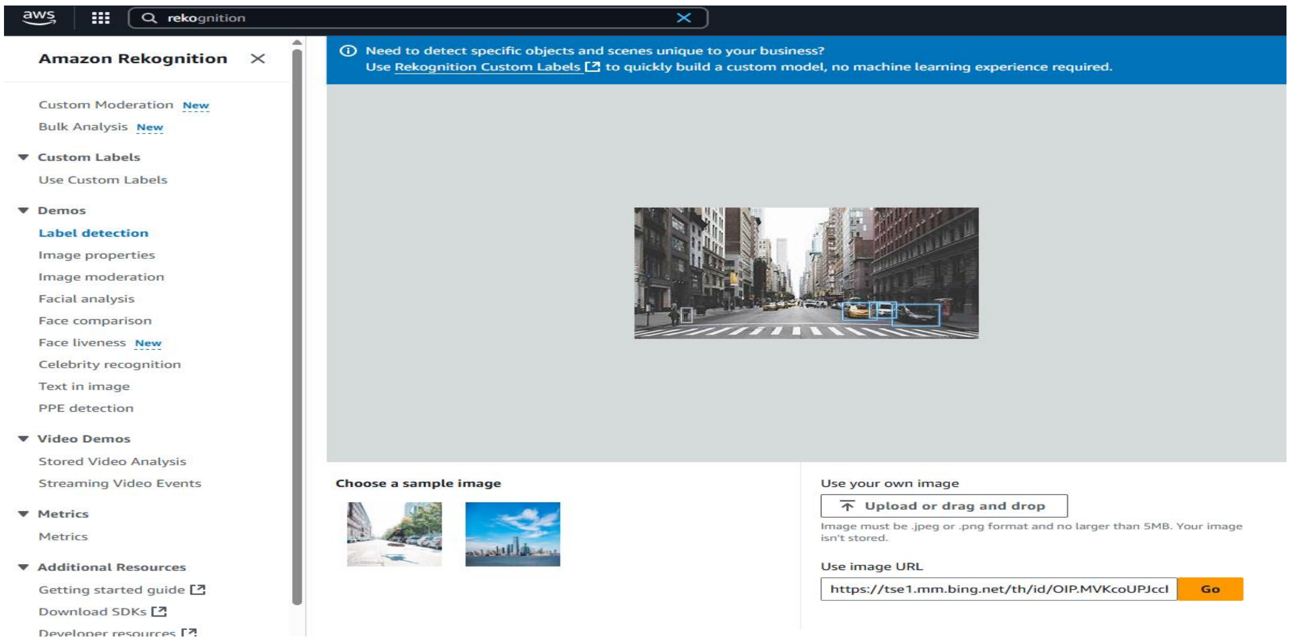
Foreground Properties

Background Properties

4. **Upload or Select Image:**

- Upload a local image
- Use a clear image containing people or objects (e.g., a group photo or street scene).

5. **Run Object and Scene Detection:**
   - Click **“Detect Labels.”**
   - Rekognition processes the image and lists detected items (e.g., “Person,” “Car,” “Building,” “Tree”).

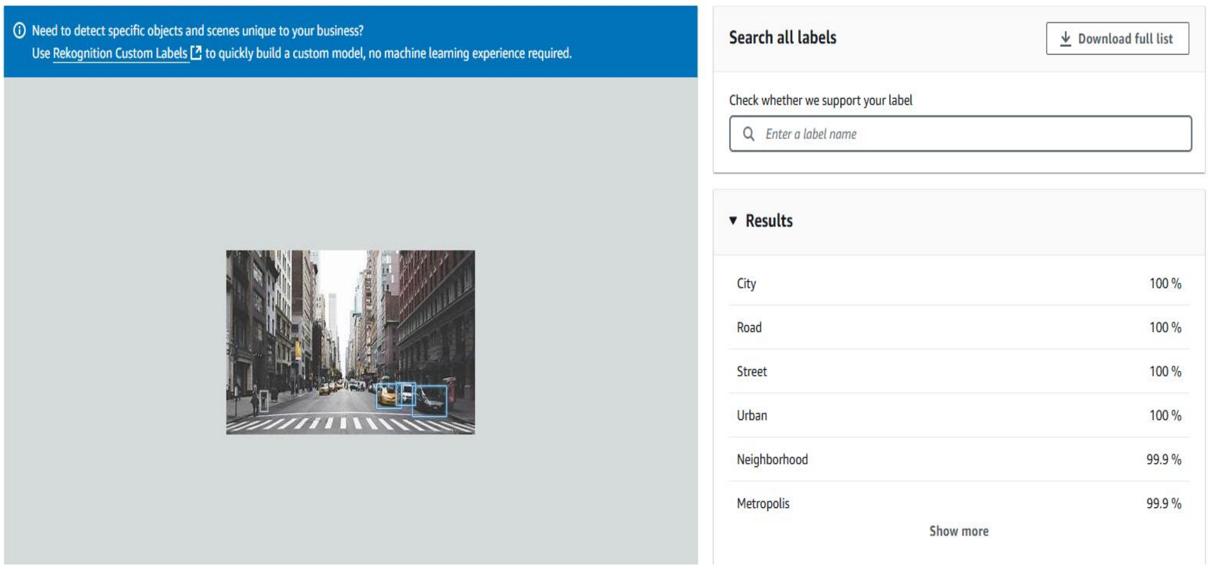


Need to detect specific objects and scenes unique to your business?
Use Rekognition Custom Labels to quickly build a custom model, no machine learning experience required.

Image must be .jpeg or .png format and no larger than 5MB. Your image isn't stored.

https://tse1.mm.bing.net/th/id/OIP.MVKcoUPJccl

6. **View Confidence Scores:**



Need to detect specific objects and scenes unique to your business?
Use Rekognition Custom Labels to quickly build a custom model, no machine learning experience required.

City	100 %
Road	100 %
Street	100 %
Urban	100 %
Neighborhood	99.9 %
Metropolis	99.9 %

   - Each label is displayed with a confidence percentage (e.g., Person – 99.8%, Car – 97.2%).

Conclusion

AWS Rekognition demonstrates the power of cloud-based computer vision systems in recognizing and classifying image content. It simplifies the development of intelligent visual applications for industries like security, retail, and automation.