




AWS Lambda + OpenAI Integration for Replit Upload Center

INTEGRATION OVERVIEW

You already have:

-  AWS Lambda function set up
-  OpenAI API integrated
-  Document analysis capabilities

Replit needs to: Connect the upload workflow to your existing AWS Lambda endpoint.

FOR REPLIT AGENT - EXACT INSTRUCTIONS

Tell Replit:

"I have an existing AWS Lambda function with OpenAI API that analyzes documents. After files are uploaded in the Upload Center, call my Lambda endpoint to get real AI analysis results. Here's how to integrate it:"

Step 1: Upload Workflow Integration

JSX

```
const handleFileUpload = async (files) => {
  // 1. Upload files first
  const uploadedFiles = await uploadFiles(files);

  // 2. Open LEFT sidebar immediately
  setDocuments(uploadedFiles);
  setSidebarOpen(true);

  // 3. Call AWS Lambda for each document
  uploadedFiles.forEach(async (doc) => {
    // Show "Analyzing..." status
    updateDocumentStatus(doc.id, 'analyzing');

    // Call your AWS Lambda endpoint
    const analysisResult = await callAWSLambda(doc);

    // Update with real AI results
    updateDocumentStatus(doc.id, 'analyzed', analysisResult);
  });
}
```

```
});  
};
```

Step 2: AWS Lambda API Call

JSX

```
const callAWSLambda = async (document) => {  
  try {  
    // Call your existing Lambda endpoint  
    const response = await fetch('YOUR_LAMBDA_ENDPOINT_URL', {  
      method: 'POST',  
      headers: {  
        'Content-Type': 'application/json',  
        'Authorization': 'Bearer YOUR_API_KEY' // if needed  
      },  
      body: JSON.stringify({  
        documentUrl: document.url,  
        documentType: document.type,  
        fileName: document.name  
      })  
    });  
  
    const result = await response.json();  
  
    return {  
      extractedText: result.extractedText,  
      keyValuePairs: result.keyValuePairs,  
      documentType: result.documentType,  
      confidence: result.confidence,  
      summary: result.summary  
    };  
  
  } catch (error) {  
    console.error('Lambda analysis failed:', error);  
    return {  
      extractedText: 'Analysis failed',  
      keyValuePairs: [],  
      documentType: 'unknown',  
      confidence: 0  
    };  
  }  
};
```

Step 3: Display Real AI Results

JSX

```
const DocumentItem = ({ document }) => {
  return (
    <div className="document-item">
      <div className="document-info">
        <h4>{document.name}</h4>
        <p className="status">
          {document.status === 'analyzing' && '⚡ Analyzing...'}
          {document.status === 'analyzed' && '✅ Analysis Complete'}
        </p>
      </div>

      {/* Lightning bolt appears after real analysis */}
      {document.status === 'analyzed' && (
        <button
          className="lightning-bolt"
          onClick={() => showDetailsModal(document)}
        >
          ⚡ Details {document.analysisResult?.keyValuePairs?.length || 0}
        </button>
      )}
    </div>
  );
};
```

Step 4: Details Modal with Real Data

JSX

```
const DetailsModal = ({ document, onClose }) => {
  const analysis = document.analysisResult;

  return (
    <div className="details-modal">
      <div className="modal-header">
        <h3>📄 {document.name}</h3>
        <button onClick={onClose}>✕</button>
      </div>

      <div className="modal-content">
        {/* Document Type */}
        <div className="analysis-section">
          <h4>Document Type</h4>
          <p>{analysis.documentType}</p>
          <span className="confidence">Confidence: {analysis.confidence}%</span>
        </div>
      </div>
    </div>
  );
};
```

```

</div>

{/* Extracted Key-Value Pairs */}
<div className="analysis-section">
  <h4>Extracted Information</h4>
  {analysis.keyValuePairs.map((pair, index) => (
    <div key={index} className="key-value-pair">
      <span className="key">{pair.key}</span>
      <span className="value">{pair.value}</span>
      <span className="confidence">{pair.confidence}%</span>
    </div>
  ))}
</div>

{/* AI Summary */}
{analysis.summary && (
  <div className="analysis-section">
    <h4>AI Summary</h4>
    <p>{analysis.summary}</p>
  </div>
)}

{/* Extracted Text Preview */}
<div className="analysis-section">
  <h4>Extracted Text</h4>
  <div className="extracted-text">
    {analysis.extractedText.substring(0, 500)}...
  </div>
</div>
</div>

<div className="modal-actions">
  <button className="accept-btn" onClick={() =>
acceptDocument(document)}>
     Accept & Route
  </button>
  <button className="dismiss-btn" onClick={onClose}>
     Dismiss
  </button>
</div>
</div>
);
};

```



CONFIGURATION NEEDED

Ask the user for:

1. **Lambda Endpoint URL** - Your AWS Lambda function URL
2. **API Authentication** - Any API keys or tokens needed
3. **Request Format** - Expected input format for your Lambda
4. **Response Format** - What your Lambda returns

EXAMPLE LAMBDA INTEGRATION

JSX

```
// Complete integration example
const UploadCenterWithLambda = () => {
  const [documents, setDocuments] = useState([]);
  const [sidebarOpen, setSidebarOpen] = useState(false);

  const handleUpload = async (files) => {
    // Upload files
    const uploadedDocs = await uploadFiles(files);

    // Open sidebar
    setDocuments(uploadedDocs);
    setSidebarOpen(true);

    // Process each document with Lambda
    uploadedDocs.forEach(async (doc) => {
      // Update status to analyzing
      setDocuments(prev => prev.map(d =>
        d.id === doc.id ? { ...d, status: 'analyzing' } : d
      ));

      // Call your AWS Lambda
      const analysis = await fetch('YOUR_LAMBDA_URL', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          documentUrl: doc.url,
          fileName: doc.name
        })
      }).then(res => res.json());

      // Update with results
      setDocuments(prev => prev.map(d =>
        d.id === doc.id ? {
          ...d,
          status: 'analyzed',
          analysisResult: analysis
        } : d
      ));
    });
  };
};
```

```

        } : d
      ));
    });
  };

  return (
    <div className="upload-center">
      { /* Upload buttons */ }
      <UploadButtons onUpload={handleUpload} />

      { /* Sidebar with real AI results */ }
      <LeftSidebar
        isOpen={sidebarOpen}
        documents={documents}
        onClose={() => setSidebarOpen(false)}
      />
    </div>
  );
};

```



TELL REPLIT:

"Integrate my existing AWS Lambda + OpenAI setup into the upload workflow. After files upload, call my Lambda endpoint for real document analysis, then show the results in the lightning bolt details modal. I'll provide the Lambda URL and any authentication details needed."

This connects your real AI analysis to the Trustworthy workflow!