**🎯 Simple AI Use Case: CPI Log Failure Classification**

**Goal:**  
Given basic CPI message log fields, classify whether the log represents a **Success** or **Failure**, and if failure, assign a **bucket** (e.g., Connectivity, Mapping, Security).

**Why this is simple**

* **Input:** A few columns from CPI log (e.g., ARTIFACT\_NAME, ORIGIN\_COMPONENT\_NAME, LOG\_LEVEL, CUSTOM\_STATUS).
* **Model:** A basic scikit-learn classifier (Logistic Regression / RandomForest).
* **Output:** SUCCESS or one of 3–4 failure buckets.
* **BTP Pieces:**
  + **AI Core** → Train + Serve the model.
  + **Object Store** → Store dataset (CSV).
  + **HANA Cloud** → Store history of predictions.
  + **AI Launchpad** → Monitor executions/deployments.

Absolutely—here’s a **concise, end-to-end checklist** to train and deploy your CPI classifier on **SAP BTP AI Core** using **AWS S3** for data/model storage.

**0) Prereqs (one-time)**

* **AI Core / AI Launchpad** tenant is available.
* **Runtime** is ready (your admin wires the cluster; you don’t need cluster access).
* **Container registry** you can push to (e.g., ECR/GCR/Docker Hub).
* **AWS S3 bucket** with permissions:
  + Read for training dataset (e.g., s3://your-bucket/cpi/cpi\_logs.csv)
  + (Optional) Write for saving trained model (e.g., s3://your-bucket/cpi/models/model.pkl)
* **Secrets**
  + Image pull secret in the runtime (ask admin if private registry).
  + S3 creds (can be provided as env vars in Executions/Serving):
    - AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, *(optional)* AWS\_SESSION\_TOKEN
    - AWS\_DEFAULT\_REGION (e.g., us-east-1)

**1) Build & push images**

From your project root (with src/ and docker/ in place):

# Trainer + batch summary image

docker build -f docker/Dockerfile.train -t <registry>/<repo>:cpi-triage-ai-train .

docker push <registry>/<repo>:cpi-triage-ai-train

# Online serving image (FastAPI/KServe)

docker build -f docker/Dockerfile.serve -t <registry>/<repo>:cpi-triage-ai-serve .

docker push <registry>/<repo>:cpi-triage-ai-serve

**2) Put your dataset in S3**

* Upload a CSV (e.g., cpi\_logs.csv) to your bucket:
  + Must include the fields you shared (e.g., ARTIFACT\_NAME, ORIGIN\_COMPONENT\_NAME, LOG\_LEVEL, and label column like CUSTOM\_STATUS).
* Note the URI: s3://your-bucket/cpi/cpi\_logs.csv.

**3) Import the Applications YAMLs (Executables + Serving)**

In **AI Launchpad → Applications → Add → YAML**, import:

1. **Training Executable** (WorkflowTemplate)
   * applications/cpi-triage-ai-train.yaml
   * It has labels/annotations so it appears under scenario **CPI-Triage-AI**.
2. **(Optional) Daily Summary Executable**
   * applications/cpi-triage-ai-daily-summary.yaml (batch predict + summary files)
3. **Serving Executable** (ServingTemplate / KServe)
   * applications/cpi-triage-ai-serving.yaml
   * Creates a “Serving Executable” to deploy your FastAPI.

**Replace placeholders** in the YAMLs before importing:

* <your-registry>/<repo>
* <your-docker-secret> (if needed)
* Keep scenario id consistent: cpi-triage-ai

**4) Run a training Execution**

In **Launchpad → Executions → New**:

* Pick **Executable**: “**Train CPI classifier**”.
* **Parameters**:
  + S3\_DATA\_URI = s3://your-bucket/cpi/cpi\_logs.csv
  + *(Optional)* S3\_MODEL\_URI = s3://your-bucket/cpi/models/model.pkl
  + TARGET\_COLUMN = CUSTOM\_STATUS (or your label)
  + AWS\_DEFAULT\_REGION = us-east-1
* **Env vars** (if not already in the runtime as secrets):
  + AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, *(optional)* AWS\_SESSION\_TOKEN
* **Output**: AI Core will persist a **model artifact** named cpimodel (also optionally write to S3 if S3\_MODEL\_URI set).

✅ When the run finishes, you should see cpimodel attached to the execution.

**5) Create a Serving from the Serving Executable**

In **Scenarios → CPI-Triage-AI → Serving Executables → cpi-model-serving → Create**:

* **Input artifact**: select cpimodel from the training run.
* **Resource plan**: choose a suitable plan (e.g., starter).
* (If private image) **imagePullSecrets**: select your secret.
* **Environment** (the ServingTemplate already injects):
  + STORAGE\_URI / S3\_MODEL\_URI ← AI Core fills with the cpimodel artifact URI
  + AWS\_DEFAULT\_REGION (ensure it matches)
  + (If needed) AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, AWS\_SESSION\_TOKEN

Create it → wait until it’s **Ready** and shows an **endpoint URL**.

**6) Test the endpoint**

# Health

curl -s https://<endpoint>/health

# Predict (example)

curl -s -X POST https://<endpoint>/predict \

-H "Content-Type: application/json" \

-d '{"ARTIFACT\_NAME":"InvoiceProcessing","ORIGIN\_COMPONENT\_NAME":"HTTP","LOG\_LEVEL":"ERROR"}'

Response:

{"prediction": "<bucket>", "confidence": 0.93}

**7) (Optional) Schedule daily summary**

Use the **Daily summary executable** to:

* Input: latest day’s dataset (S3 or attached dataset)
* Input: cpimodel artifact from training
* Output: summaryout (JSON/CSV with bucket counts)
* Optionally configure SLACK\_WEBHOOK\_URL / TEAMS\_WEBHOOK\_URL for notifications

**Common gotchas (quick fixes)**

* **No executables appear**: check YAML has
  + labels: { scenarios.ai.sap.com/id: "cpi-triage-ai", ai.sap.com/version: "4.0" }
  + Required annotations under metadata.annotations (scenario/executable/artifacts).
* **Image can’t pull**: set a valid **imagePullSecret** in the ServingTemplate / runtime.
* **S3 read fails**: missing AWS creds/region or blocked network path. Add creds as env vars in Execution/Serving; confirm bucket policy/allow-list/VPC endpoint if used.
* **Model not found at serve**: ensure ServingTemplate passes STORAGE\_URI/S3\_MODEL\_URI and your container loads from it (the provided serve\_cpi\_btpcore.py does).
* **Wrong port**: serving image must expose and listen on **8080** (or match the ServingTemplate).

**Minimal things you must change**

* Set **your registry image names** in all three YAMLs.
* Provide **S3 URIs** for dataset/model.
* Provide **AWS credentials** either in the runtime or as env vars in Executions/Serving.

If you paste your **registry path** (e.g., docker.io/yourname/cpi) and tell me whether you need an **image pull secret** name, I can return the **three ready-to-import YAMLs** with everything filled in for your environment.

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info.

Absolutely—here’s a **concise, end-to-end checklist** to train and deploy your CPI classifier on **SAP BTP AI Core** using **AWS S3** for data/model storage.

**0) Prereqs (one-time)**

* **AI Core / AI Launchpad** tenant is available.
* **Runtime** is ready (your admin wires the cluster; you don’t need cluster access).
* **Container registry** you can push to (e.g., ECR/GCR/Docker Hub).
* **AWS S3 bucket** with permissions:
  + Read for training dataset (e.g., s3://your-bucket/cpi/cpi\_logs.csv)
  + (Optional) Write for saving trained model (e.g., s3://your-bucket/cpi/models/model.pkl)
* **Secrets**
  + Image pull secret in the runtime (ask admin if private registry).
  + S3 creds (can be provided as env vars in Executions/Serving):
    - AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, *(optional)* AWS\_SESSION\_TOKEN
    - AWS\_DEFAULT\_REGION (e.g., us-east-1)

**1) Build & push images**

From your project root (with src/ and docker/ in place):

# Trainer + batch summary image

docker build -f docker/Dockerfile.train -t <registry>/<repo>:cpi-triage-ai-train .

docker push <registry>/<repo>:cpi-triage-ai-train

# Online serving image (FastAPI/KServe)

docker build -f docker/Dockerfile.serve -t <registry>/<repo>:cpi-triage-ai-serve .

docker push <registry>/<repo>:cpi-triage-ai-serve

**2) Put your dataset in S3**

* Upload a CSV (e.g., cpi\_logs.csv) to your bucket:
  + Must include the fields you shared (e.g., ARTIFACT\_NAME, ORIGIN\_COMPONENT\_NAME, LOG\_LEVEL, and label column like CUSTOM\_STATUS).
* Note the URI: s3://your-bucket/cpi/cpi\_logs.csv.

**3) Import the Applications YAMLs (Executables + Serving)**

In **AI Launchpad → Applications → Add → YAML**, import:

1. **Training Executable** (WorkflowTemplate)
   * applications/cpi-triage-ai-train.yaml
   * It has labels/annotations so it appears under scenario **CPI-Triage-AI**.
2. **(Optional) Daily Summary Executable**
   * applications/cpi-triage-ai-daily-summary.yaml (batch predict + summary files)
3. **Serving Executable** (ServingTemplate / KServe)
   * applications/cpi-triage-ai-serving.yaml
   * Creates a “Serving Executable” to deploy your FastAPI.

**Replace placeholders** in the YAMLs before importing:

* <your-registry>/<repo>
* <your-docker-secret> (if needed)
* Keep scenario id consistent: cpi-triage-ai

**4) Run a training Execution**

In **Launchpad → Executions → New**:

* Pick **Executable**: “**Train CPI classifier**”.
* **Parameters**:
  + S3\_DATA\_URI = s3://your-bucket/cpi/cpi\_logs.csv
  + *(Optional)* S3\_MODEL\_URI = s3://your-bucket/cpi/models/model.pkl
  + TARGET\_COLUMN = CUSTOM\_STATUS (or your label)
  + AWS\_DEFAULT\_REGION = us-east-1
* **Env vars** (if not already in the runtime as secrets):
  + AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, *(optional)* AWS\_SESSION\_TOKEN
* **Output**: AI Core will persist a **model artifact** named cpimodel (also optionally write to S3 if S3\_MODEL\_URI set).

✅ When the run finishes, you should see cpimodel attached to the execution.

**5) Create a Serving from the Serving Executable**

In **Scenarios → CPI-Triage-AI → Serving Executables → cpi-model-serving → Create**:

* **Input artifact**: select cpimodel from the training run.
* **Resource plan**: choose a suitable plan (e.g., starter).
* (If private image) **imagePullSecrets**: select your secret.
* **Environment** (the ServingTemplate already injects):
  + STORAGE\_URI / S3\_MODEL\_URI ← AI Core fills with the cpimodel artifact URI
  + AWS\_DEFAULT\_REGION (ensure it matches)
  + (If needed) AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, AWS\_SESSION\_TOKEN

Create it → wait until it’s **Ready** and shows an **endpoint URL**.

**6) Test the endpoint**

# Health

curl -s https://<endpoint>/health

# Predict (example)

curl -s -X POST https://<endpoint>/predict \

-H "Content-Type: application/json" \

-d '{"ARTIFACT\_NAME":"InvoiceProcessing","ORIGIN\_COMPONENT\_NAME":"HTTP","LOG\_LEVEL":"ERROR"}'

Response:

{"prediction": "<bucket>", "confidence": 0.93}

**7) (Optional) Schedule daily summary**

Use the **Daily summary executable** to:

* Input: latest day’s dataset (S3 or attached dataset)
* Input: cpimodel artifact from training
* Output: summaryout (JSON/CSV with bucket counts)
* Optionally configure SLACK\_WEBHOOK\_URL / TEAMS\_WEBHOOK\_URL for notifications

**Common gotchas (quick fixes)**

* **No executables appear**: check YAML has
  + labels: { scenarios.ai.sap.com/id: "cpi-triage-ai", ai.sap.com/version: "4.0" }
  + Required annotations under metadata.annotations (scenario/executable/artifacts).
* **Image can’t pull**: set a valid **imagePullSecret** in the ServingTemplate / runtime.
* **S3 read fails**: missing AWS creds/region or blocked network path. Add creds as env vars in Execution/Serving; confirm bucket policy/allow-list/VPC endpoint if used.
* **Model not found at serve**: ensure ServingTemplate passes STORAGE\_URI/S3\_MODEL\_URI and your container loads from it (the provided serve\_cpi\_btpcore.py does).
* **Wrong port**: serving image must expose and listen on **8080** (or match the ServingTemplate).

**Minimal things you must change**

* Set **your registry image names** in all three YAMLs.
* Provide **S3 URIs** for dataset/model.
* Provide **AWS credentials** either in the runtime or as env vars in Executions/Serving.