TRIANZ℠

Deployment Instructions

AI-Powered Policy Agent

Trianz_AWS Hackathon '25

trianz.com

TABLE OF CONTENTS

Pre-Deployment Requirements

AWS Account Setup: You need an AWS account with Bedrock access enabled in your target region (us-east-1 recommended for full Nova model availability). Ensure your account has sufficient service quotas for Bedrock model invocations.

IAM Permissions: Your deployment user or role requires:

- Bedrock full access for model invocation
- S3 full access for document storage and status tracking
- CloudWatch Logs for monitoring and debugging
- AgentCore deployment permissions
- STS assume role capabilities for credential management
- Kiro IDE

Local Development Environment: Install Python 3.9 or higher, pip for package management, and the AgentCore CLI tool. Ensure you have AWS CLI configured with appropriate credentials.

S3 Bucket Preparation: Create a dedicated S3 bucket for your deployment (e.g., trianz-underwriting-documents-121409194654). Configure appropriate lifecycle policies for automated cleanup of old sessions.

Deployment Commands

AWS Bedrock AgentCore - Command Reference

Initial Setup Commands

Install AWS CLI (if not already installed)

bash

```
# For Linux/macOS
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
# For Windows
# Use Kiro IDE
# Download and run the MSI installer from:
# https://awscli.amazonaws.com/AWSCLIV2.msi
```

Configure AWS Credentials

bash

```
# Configure default profile
aws configure
# You'll be prompted for:
# AWS Access Key ID: [Enter your access key]
# AWS Secret Access Key: [Enter your secret key]
# Default region name: us-east-1
# Default output format: json
# Verify configuration
aws sts get-caller-identity
# Expected output shows your Account ID, UserId, and ARN
```

Install Python and Dependencies

bash

```
# Check Python version (requires 3.9+)
python --version
# or
python3 --version
# Create virtual environment (recommended)
python3 -m venv venv
# Activate virtual environment
# On Linux/macOS:
source venv/bin/activate
# On Windows:
venv\Scripts\activate
# Install application dependencies
pip install -r requirements.txt
# Verify key packages installed
pip list | grep bedrock-agentcore
pip list | grep strands-agents
pip list | grep boto3
```

Install AgentCore CLI

bash

```
# Install AgentCore command-line interface
pip install bedrock-agentcore-cli
# Verify installation
agentcore --version
# Check available commands
agentcore --help
```

S3 Bucket Setup Commands

Create S3 Bucket

bash

```
# Create bucket in us-east-1 region
aws s3 mb s3://trianz-underwriting-documents-121409194654 --region us-east-1
# Verify bucket created
aws s3 ls | grep trianz-underwriting
# Enable versioning (recommended for production)
aws s3api put-bucket-versioning \
  --bucket trianz-underwriting-documents-121409194654 \
  --versioning-configuration Status=Enabled
```

Test S3 Access

bash

```
# Upload test file
echo "Test file" > test.txt
```

```
aws s3 cp test.txt s3://trianz-underwriting-documents-121409194654/
# List bucket contents
aws s3 ls s3://trianz-underwriting-documents-121409194654/
# Download test file
aws s3 cp s3://trianz-underwriting-documents-121409194654/test.txt ./test-downloaded.txt
# Clean up test file
aws s3 rm s3://trianz-underwriting-documents-121409194654/test.txt
rm test.txt test-downloaded.txt
```

AgentCore Deployment Commands

Prepare Application for Deployment

bash

```
# Navigate to application directory
cd /path/to/your/sdis-application
# Verify all required files exist
ls -la agentcore_main.py underwriting_agents.py config.py models.py requirements.txt
# Run pre-deployment validation (if available)
python -m py_compile agentcore_main.py
python -m py_compile underwriting_agents.py
# Check for syntax errors
python agentcore_main.py --help 2>&1 | head -n 5
```

Deploy to AgentCore

bash

```
# Basic deployment command
agentcore deploy \
  --name trianz-underwriting-system \
  --region us-east-1 \
  --entry-point agentcore_main.py
# Deployment with environment variables
agentcore deploy \
  --name trianz-underwriting-system \
  --region us-east-1 \
  --entry-point agentcore_main.py \
  --env AWS_REGION=us-east-1 \
  --env S3_BUCKET=trianz-underwriting-documents-121409194654
# Deployment with custom timeout (in seconds)
agentcore deploy \
  --name trianz-underwriting-system \
  --region us-east-1 \
  --entry-point agentcore_main.py \
  --timeout 1800 \
  --memory 2048
# View deployment progress
# Output shows:
# - Packaging application
# - Uploading to AWS
# - Creating IAM roles
# - Configuring CloudWatch
# - Deployment URL and invocation details
```

Verify Deployment

bash

```
# List all deployed AgentCore applications
agentcore list --region us-east-1
# Get specific application details
agentcore describe \
  --name trianz-underwriting-system \
  --region us-east-1
# Check application status
agentcore status \
  --name trianz-underwriting-system \
  --region us-east-1
```

Test AgentCore Deployment

bash

```
# Test with simple health check payload
agentcore invoke '{"request_type": "get_status", "session_id": "test-session"}' \
  --name trianz-underwriting-system \
  --region us-east-1
# Test session creation
agentcore invoke '{"request_type": "create_session"}' \
  --name trianz-underwriting-system \
  --region us-east-1

# Expected output shows session_id and s3_bucket information
```

Running the Frontend Demo

Start Flask Application

bash

```
# Ensure you're in the application directory
cd /path/to/your/sdis-application
# Activate virtual environment (if not already active)
source venv/bin/activate  # Linux/macOS
# or
venv\Scripts\activate  # Windows
# Set environment variables (if not in .env file)
export AWS_REGION=us-east-1
export AWS_DEFAULT_REGION=us-east-1
export S3_BUCKET=trianz-underwriting-documents-121409194654
# Start the Flask application
```

```
python run.py
# Expected output:
# ============================================================
# NOVA SONIC VOICE INTEGRATION policy generation
# ============================================================
# [INFO] AWS Region configured: us-east-1
# [INFO] S3 Bucket: trianz-underwriting-documents-121409194654
# [SUCCESS] S3 bucket accessible: trianz-underwriting-documents-121409194654
# ============================================================
# [INFO] Starting Flask-SocketIO with Nova Sonic integration
# [INFO] Access: http://127.0.0.1:8080
# [INFO] Health check: http://127.0.0.1:8080/health
# ============================================================
```

Access the Application

```bash
# Open in default browser (Linux/macOS)
open http://127.0.0.1:8080
# Open in default browser (Windows)
start http://127.0.0.1:8080
# Or manually navigate to:
# http://127.0.0.1:8080
```

Test Health Endpoint

```bash
# Check application health
curl http://127.0.0.1:8080/health | jq
# Expected output:
# {
#   "status": "healthy",
#   "timestamp": "2025-01-20T14:30:00",
#   "service": "TRIANZ Underwriting System - S3 Frontend with Nova Sonic",
#   "version": "3.0.0",
#   "s3_bucket": "trianz-underwriting-documents-121409194654",
#   "s3_status": "connected",
#   "nova_sonic": "enabled"
# }
```

Testing Commands

Create Test Session

```bash
# Using curl to create session
curl -X POST http://127.0.0.1:8080/api/create-session | jq
# Using AgentCore CLI
agentcore invoke '{"request_type": "create_session"}' \
  --name trianz-underwriting-system \
  --region us-east-1 | jq
# Save session_id from response for subsequent commands
SESSION_ID="session_2025-01-20_14-30-45_abc12345"
```

Upload Test Documents

bash

```
# Prepare test ZIP file with sample PDFs
zip -r test-documents.zip sample_policies/*.pdf
# Upload using curl
curl -X POST http://127.0.0.1:8080/upload/$SESSION_ID \
  -F "zipFileInput=@test-documents.zip"
# Verify upload in S3
aws s3 ls s3://trianz-underwriting-documents-121409194654/$SESSION_ID/
# Expected output shows uploaded ZIP file
```

Trigger AgentCore Processing

bash

```
# Create processing payload
cat > process-payload.json <<EOF
{
  "request_type": "s3_process",
  "s3_bucket": "trianz-underwriting-documents-121409194654",
  "s3_key": "$SESSION_ID/${SESSION_ID}_upload.zip",
  "session_id": "$SESSION_ID"
}
EOF
# Invoke AgentCore processing
agentcore invoke "$(cat process-payload.json)" \
  --name trianz-underwriting-system \
  --region us-east-1
# This triggers the 8-agent workflow
```

Monitor Processing Status

bash

```
# Check status via HTTP endpoint
curl http://127.0.0.1:8080/status/$SESSION_ID | jq
# Check agent status directly from S3
aws s3 cp s3://trianz-underwriting-documents-
121409194654/$SESSION_ID/agent_status.json - | jq
# Watch for status updates (Linux/macOS)
watch -n 5 "aws s3 cp s3://trianz-underwriting-documents-
121409194654/$SESSION_ID/agent_status.json - | jq '.agents | to_entries[] | {agent: .key,
status: .value.status}'"
# Manual polling (all platforms)
while true; do
  curl -s http://127.0.0.1:8080/status/$SESSION_ID | jq '.agents | to_entries[] | {agent: .key,
status: .value.status}'
  sleep 5
```

```
done
```

Retrieve Processing Results

bash

```
# Get comprehensive summary
curl http://127.0.0.1:8080/status/$SESSION_ID | jq '.agents.summary_generation.analysis'
# Download generated policy PDF
curl http://127.0.0.1:8080/download_policy/$SESSION_ID \
  --output policy_$SESSION_ID.pdf
# View policy in browser
curl http://127.0.0.1:8080/view_policy/$SESSION_ID
# Get policy status
curl http://127.0.0.1:8080/policy_status/$SESSION_ID | jq
```

Monitoring and Logging Commands

View CloudWatch Logs

bash

```
# List log groups for AgentCore application
aws logs describe-log-groups \
  --log-group-name-prefix /aws/agentcore/trianz-underwriting-system
# Get latest log stream
LOG_STREAM=$(aws logs describe-log-streams \
  --log-group-name /aws/agentcore/trianz-underwriting-system \
  --order-by LastEventTime \
  --descending \
  --max-items 1 \
  --query 'logStreams[0].logStreamName' \
  --output text)
echo "Latest log stream: $LOG_STREAM"
# View recent logs
aws logs tail /aws/agentcore/trianz-underwriting-system \
  --follow \
  --format short
# Search logs for specific session
aws logs filter-log-events \
  --log-group-name /aws/agentcore/trianz-underwriting-system \
  --filter-pattern "$SESSION_ID" \
  --start-time $(date -u -d '1 hour ago' +%s)000
# Search for errors
aws logs filter-log-events \
  --log-group-name /aws/agentcore/trianz-underwriting-system \
  --filter-pattern "ERROR" \
  --start-time $(date -u -d '1 hour ago' +%s)000
```

Monitor S3 Session Activity

bash

```
# List all active sessions
aws s3 ls s3://trianz-underwriting-documents-121409194654/ | grep session_
# Count total sessions
```

```
aws s3 ls s3://trianz-underwriting-documents-121409194654/ | grep session_ | wc -l
# Get session details
aws s3 ls s3://trianz-underwriting-documents-121409194654/$SESSION_ID/ --recursive --
human-readable
# Monitor session folder size
aws s3 ls s3://trianz-underwriting-documents-121409194654/$SESSION_ID/ --recursive --
summarize
# Download all session files
aws s3 sync s3://trianz-underwriting-documents-121409194654/$SESSION_ID/
./session_backup/$SESSION_ID/
```

Application Performance Monitoring

bash

```
# Check AgentCore metrics
aws cloudwatch get-metric-statistics \
  --namespace AWS/AgentCore \
  --metric-name Invocations \
  --dimensions Name=ApplicationName,Value=trianz-underwriting-system \
  --start-time $(date -u -d '1 hour ago' --iso-8601) \
  --end-time $(date -u --iso-8601) \
  --period 300 \
  --statistics Sum
# Check error rate
aws cloudwatch get-metric-statistics \
  --namespace AWS/AgentCore \
  --metric-name Errors \
  --dimensions Name=ApplicationName,Value=trianz-underwriting-system \
  --start-time $(date -u -d '1 hour ago' --iso-8601) \
  --end-time $(date -u --iso-8601) \
  --period 300 \
  --statistics Sum
# Check execution duration
aws cloudwatch get-metric-statistics \
  --namespace AWS/AgentCore \
  --metric-name Duration \
  --dimensions Name=ApplicationName,Value=trianz-underwriting-system \
  --start-time $(date -u -d '1 hour ago' --iso-8601) \
  --end-time $(date -u --iso-8601) \
  --period 300 \
  --statistics Average,Maximum
```

Update and Maintenance Commands

Update AgentCore Application

bash

```
# Make code changes, then redeploy
agentcore deploy \
  --name trianz-underwriting-system \
  --region us-east-1 \
  --entry-point agentcore_main.py \
  --update
# Deploy new version without replacing current
agentcore deploy \
  --name trianz-underwriting-system \
  --region us-east-1 \
  --entry-point agentcore_main.py \
  --version v2
# Switch traffic to new version
agentcore update-alias \
  --name trianz-underwriting-system \
  --alias production \
  --version v2 \
  --region us-east-1
```

Rollback Commands

bash

```
# List all versions
agentcore list-versions \
  --name trianz-underwriting-system \
  --region us-east-1
# Rollback to previous version
agentcore update-alias \
  --name trianz-underwriting-system \
  --alias production \
  --version v1 \
  --region us-east-1
# Delete specific version
agentcore delete-version \
  --name trianz-underwriting-system \
  --version v2 \
  --region us-east-1
```

TRIANZ℠

Thank You

reach@trianz.com