Prerequisites

- **AWS EC2 instance** (Ubuntu-based recommended) with security group allowing SSH and outbound SRT/RTMP traffic.
- **SSH key pair** (`stream-relay-kp.pem`) with correct file permissions (`chmod 400 stream-relay-kp.pem`).

GStreamer installed on the EC2 instance:

```
sudo apt update
sudo apt install -y gstreamer1.0-tools gstreamer1.0-plugins-base \
gstreamer1.0-plugins-good gstreamer1.0-plugins-bad \
gstreamer1.0-plugins-ugly gstreamer1.0-libav
```

-
- Camera RTSP credentials and network access.
- AWS IVS / MediaLive ingest endpoint (SRT or RTMP).

Step 1: Connect to the EC2 Instance

```
ssh -i stream-relay-kp.pem ubuntu@34.243.156.239
```

Tapo Cameras

TAPO C246D Dual Cam

```
#!/usr/bin/env bash
set -euo pipefail

# ------------------------------------------------------------------
# CONFIGURATION (edit only these lines)
# ------------------------------------------------------------------
```

```
ROLE_NAME="LiveStreamRole"                # IAM role attached to EC2
STREAM_NAME="TAPO-C246D-Stream"           # KVS stream name
RTSP_URL="rtsp://cctv-dual-tapo:squirrel@192.168.9.241:554/stream2"
AWS_REGION="eu-west-1"
METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                   # 6 hours (max for IMDSv2)
REFRESH_BEFORE_EXPIRY=300                 # Refresh 5 min early
# ------------------------------------------------------------------

# ---------- Helper: get a fresh token ----------
get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

# ---------- Helper: get credentials for the role ----------
get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

# ---------- Export fresh env vars ----------
export_credentials() {
    local creds_json="$1"
    export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq -r
.AccessKeyId)
    export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq -r
.SecretAccessKey)
    export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq -r .Token)
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(echo "$creds_json" | jq -r
.Expiration)" +%s)
}

# ---------- Build the GStreamer command ----------
build_gst_cmd() {
    cat <<EOF
```

```
GST_DEBUG=kvssink:2 \
gst-launch-1.0 -e rtspsrc location="$RTSP_URL" latency=0 name=src \
  src. ! application/x-rtp,media=video,encoding-name=H264 !
rtph264depay ! h264parse ! avdec_h264 \
       ! videoconvert \
       ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
                 ! video/x-h264,profile=baseline \
       ! kvssink stream-name="$STREAM_NAME" aws-region="$AWS_REGION"
storage-size=512
EOF
}

# ---------- Start (or restart) the pipeline ----------
start_pipeline() {
    echo "[$(date)] Starting pipeline with fresh credentials (expire
$(date -d "@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &
    GST_PID=$!
    echo $GST_PID > /tmp/kvs_pipeline.pid
}

# ---------- Main credential-refresh loop ----------
main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        REFRESH_AT=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY ))

        SLEEP_SECONDS=$(( REFRESH_AT - NOW ))
        if (( SLEEP_SECONDS > 0 )); then
            echo "[$(date)] Sleeping $SLEEP_SECONDS s until credential
refresh..."
            sleep "$SLEEP_SECONDS"
```

```
        fi

        # Refresh credentials
        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"

        # Graceful restart
        if [[ -f /tmp/kvs_pipeline.pid ]] && kill -0 $(cat
/tmp/kvs_pipeline.pid) 2>/dev/null; then
            echo "[$(date)] Sending SIGTERM to old pipeline (PID $(cat
/tmp/kvs_pipeline.pid))"
            kill $(cat /tmp/kvs_pipeline.pid) || true
            wait $(cat /tmp/kvs_pipeline.pid) 2>/dev/null || true
        fi

        start_pipeline
    done
}

trap 'echo "[$(date)] Received signal, cleaning up..."; kill $(cat
/tmp/kvs_pipeline.pid) 2>/dev/null || true; exit' SIGINT SIGTERM

main
```

How to Use It

```
# 1. Save the script
cat > run-kvs-pipeline.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x run-kvs-pipeline.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./run-kvs-pipeline.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                                # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./run-kvs-pipeline.sh > kvs.log 2>&1 & tail -f kvs.log
```

TAPO C500

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION (edit only these lines)
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"                  # IAM role attached to EC2
STREAM_NAME="TAPO-C500-Stream"              # KVS stream name
RTSP_URL="rtsp://cctvhelpngo:squirrel@192.168.8.149:554/stream2"
PROTOCOLS="tcp"                             # force TCP (avoids UDP
packet loss on EC2)
LATENCY_MS=200                              # 200 ms buffer
AWS_REGION="eu-west-1"
METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                     # 6 h
REFRESH_BEFORE_EXPIRY=300                   # 5 min early
# ---------------------------------------------------------------------

# ---------- Helper: get a fresh IMDSv2 token ----------
get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

# ---------- Helper: get credentials for the role ----------
get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

# ---------- Export fresh env vars ----------
export_credentials() {
    local creds_json="$1"
    export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq -r
.AccessKeyId)
    export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq -r
.SecretAccessKey)
```

```
    export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq -r .Token)
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(echo "$creds_json" | jq -r
.Expiration)" +%s)
}

# ---------- Build the GStreamer command ----------
build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:2 \
gst-launch-1.0 -e \
  rtspsrc location="$RTSP_URL" \
          protocols=$PROTOCOLS latency=$LATENCY_MS name=src \
  ! queue \
  ! application/x-rtp,media=video,encoding-name=H264 \
  ! rtph264depay \
  ! h264parse \
  ! avdec_h264 \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
  ! video/x-h264,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            aws-region="$AWS_REGION" \
            storage-size=512 \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

# ---------- Start (or restart) the pipeline ----------
start_pipeline() {
    echo "[$(date)] Starting TAPO C500 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &
    GST_PID=$!
    echo $GST_PID > /tmp/kvs_c500_pipeline.pid
```

```
}

# ---------- Main credential-refresh loop ----------
main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        REFRESH_AT=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY ))
        SLEEP_SECONDS=$(( REFRESH_AT - NOW ))
        (( SLEEP_SECONDS > 0 )) && {
            echo "[$(date)] Sleeping $SLEEP_SECONDS s until
refresh..."
            sleep "$SLEEP_SECONDS"
        }

        # Refresh credentials
        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"

        # Graceful restart
        if [[ -f /tmp/kvs_c500_pipeline.pid ]] && kill -0 $(cat
/tmp/kvs_c500_pipeline.pid) 2>/dev/null; then
            echo "[$(date)] Stopping old pipeline (PID $(cat
/tmp/kvs_c500_pipeline.pid))..."
            kill $(cat /tmp/kvs_c500_pipeline.pid) || true
            wait $(cat /tmp/kvs_c500_pipeline.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] SIGTERM → killing pipeline..."; kill $(cat
/tmp/kvs_c500_pipeline.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
```

```
main
```

How to Use It

```
# 1. Save the script
cat > tapoc500-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```
then
```
chmod +x tapoc500-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./tapoc500-kvs.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                              # show the log live
```
One-Liner (Start + Live Logs)
```
nohup ./tapoc500-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Hikvision Cameras

Hikvision 2MP Dome 1

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"                  # IAM role on the EC2
STREAM_NAME="HIKVision-2MP-Dome-1-Stream"   # exact KVS stream name
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.x4j8eVyrF6mO.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                   # 5 min early refresh
# ---------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                     # 6 h

# ---------- IMDSv2 token ----------
```

```
get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

# ---------- Role credentials ----------
get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

# ---------- Export env vars ----------
export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

# ---------- GStreamer pipeline ----------
build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:6 \
gst-launch-1.0 -v -e \
  uridecodebin uri="$HLS_URL" name=dec \
  ! queue max-size-buffers=0 max-size-bytes=0 max-size-time=0 \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=veryfast
key-int-max=60 threads=4 \
  ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
```

```
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

# ---------- Start pipeline ----------
start_pipeline() {
    echo "[$(date)] Starting Hikvision Dome-1 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/hik-dome1-gst.log &
    echo $! > /tmp/kvs_hik_dome1.pid
}

# ---------- Main loop ----------
main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until credential
refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"

        # Graceful restart
        if [[ -f /tmp/kvs_hik_dome1.pid ]] && kill -0 $(cat
/tmp/kvs_hik_dome1.pid) 2>/dev/null; then
            echo "[$(date)] Stopping old pipeline (PID $(cat
/tmp/kvs_hik_dome1.pid))..."
```

```
            kill $(cat /tmp/kvs_hik_dome1.pid)
            wait $(cat /tmp/kvs_hik_dome1.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_hik_dome1.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > hikvision-dome1-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x hikvision-dome1-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./hikvision-dome1-to-kvs.sh > kvs.log 2>&1 &   # start in
background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./hikvision-dome1-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Hikvision 2MP Dome 2

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"                  # IAM role on the EC2
STREAM_NAME="HIKVision-2MP-Dome-2-Stream"   # exact KVS stream name
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.NnIXfhJeqX6i.m3u8"
AWS_REGION="eu-west-1"
```

```
REFRESH_BEFORE_EXPIRY=300                   # 5 min early refresh
# ------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                     # 6 h

# ---------- IMDSv2 token ----------
get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

# ---------- Role credentials ----------
get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

# ---------- Export env vars ----------
export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

# ---------- GStreamer pipeline ----------
build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:6 \
gst-launch-1.0 -v -e \
  uridecodebin uri="$HLS_URL" name=dec \
  ! queue max-size-buffers=0 max-size-bytes=0 max-size-time=0 \
  ! videoconvert \
```

```
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=veryfast
key-int-max=60 threads=4 \
  ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

# ---------- Start pipeline ----------
start_pipeline() {
    echo "[$(date)] Starting Hikvision Dome-2 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/hik-dome2-gst.log &
    echo $! > /tmp/kvs_hik_dome2.pid
}

# ---------- Main loop ----------
main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until credential
refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
```

```
        export_credentials "$CREDS"

        # Graceful restart
        if [[ -f /tmp/kvs_hik_dome2.pid ]] && kill -0 $(cat
/tmp/kvs_hik_dome2.pid) 2>/dev/null; then
            echo "[$(date)] Stopping old pipeline (PID $(cat
/tmp/kvs_hik_dome2.pid))..."
            kill $(cat /tmp/kvs_hik_dome2.pid)
            wait $(cat /tmp/kvs_hik_dome2.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_hik_dome2.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > hikvision-dome2-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x hikvision-dome2-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./hikvision-dome2-to-kvs.sh > kvs.log 2>&1 &   # start in
background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./hikvision-dome2-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Hikvision 2MP Bullet 1

```
#!/usr/bin/env bash
set -euo pipefail
# ------------------------------------------------------------------
```

```
# CONFIGURATION — edit only these 5 lines
# ------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="HIKVision-2MP-Bullet-1-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.vP2wVuCEN5vi.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                   # 5 min early
# ------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                     # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:6 \
gst-launch-1.0 -v -e \
```

```
  uridecodebin uri="$HLS_URL" name=dec \
  ! queue max-size-buffers=0 max-size-bytes=0 max-size-time=0 \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=veryfast
key-int-max=60 threads=4 \
  ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Hikvision Bullet-1 → KVS (expire $(date
-d "@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/hik-bullet1-gst.log &
    echo $! > /tmp/kvs_hik_bullet1.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
```

```
        export_credentials "$CREDS"

        if [[ -f /tmp/kvs_hik_bullet1.pid ]] && kill -0 $(cat
/tmp/kvs_hik_bullet1.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_hik_bullet1.pid))..."
            kill $(cat /tmp/kvs_hik_bullet1.pid)
            wait $(cat /tmp/kvs_hik_bullet1.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_hik_bullet1.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > hikvision-bullet1-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x hikvision-bullet1-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./hikvision-bullet1-to-kvs.sh > kvs.log 2>&1 &   # start in
background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./hikvision-bullet1-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Hikvision 2MP Bullet 2

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
```

```
# ------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="HIKVision-2MP-Bullet-2-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.SnPNqUVLHinu.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                  # 5 min early
# ------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                    # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:6 \
gst-launch-1.0 -v -e \
  uridecodebin uri="$HLS_URL" name=dec \
```

```
  ! queue max-size-buffers=0 max-size-bytes=0 max-size-time=0 \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=veryfast
key-int-max=60 threads=4 \
  ! video/x-h264,stream-format=avc,alignment=au,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Hikvision Bullet-2 → KVS (expire $(date
-d "@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/hik-bullet2-gst.log &
    echo $! > /tmp/kvs_hik_bullet2.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"
```

```
        if [[ -f /tmp/kvs_hik_bullet2.pid ]] && kill -0 $(cat
/tmp/kvs_hik_bullet2.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_hik_bullet2.pid))..."
            kill $(cat /tmp/kvs_hik_bullet2.pid)
            wait $(cat /tmp/kvs_hik_bullet2.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_hik_bullet2.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > hikvision-bullet2-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x hikvision-bullet2-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./hikvision-bullet2-to-kvs.sh > kvs.log 2>&1 &   # start in
background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./hikvision-bullet2-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Dahua Cameras

Dahua 2MP Dome 1

```
#!/usr/bin/env bash
set -euo pipefail
```

```
# ---------------------------------------------------------------------
# CONFIGURATION — edit only these 5 lines
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="Dahua-2MP-Dome-1-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.YMpY5kku8IXg.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                   # 5 min early
# ---------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                     # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:5 \
```

```
gst-launch-1.0 -v -e \
  souphttpsrc location="$HLS_URL" \
  ! hlsdemux \
  ! decodebin \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
  ! video/x-h264,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Dahua Dome-1 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/dahua-dome1-gst.log &
    echo $! > /tmp/kvs_dahua_dome1.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }
```

```
        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"

        if [[ -f /tmp/kvs_dahua_dome1.pid ]] && kill -0 $(cat
/tmp/kvs_dahua_dome1.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_dahua_dome1.pid))..."
            kill $(cat /tmp/kvs_dahua_dome1.pid)
            wait $(cat /tmp/kvs_dahua_dome1.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_dahua_dome1.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > dahua-dome1-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x dahua-dome1-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./dahua-dome1-to-kvs.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./dahua-dome1-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Dahua 2MP Dome 2

```
#!/usr/bin/env bash
set -euo pipefail
# ------------------------------------------------------------------
```

```
# CONFIGURATION — edit only these 5 lines
# ------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="Dahua-2MP-Dome-2-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.yyzi5SqLTftn.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                  # 5 min early
# ------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                    # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:5 \
gst-launch-1.0 -v -e \
```

```
  souphttpsrc location="$HLS_URL" \
  ! hlsdemux \
  ! decodebin \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
  ! video/x-h264,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Dahua Dome-2 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/dahua-dome2-gst.log &
    echo $! > /tmp/kvs_dahua_dome2.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
```

```
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"

        if [[ -f /tmp/kvs_dahua_dome2.pid ]] && kill -0 $(cat
/tmp/kvs_dahua_dome2.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_dahua_dome2.pid))..."
            kill $(cat /tmp/kvs_dahua_dome2.pid)
            wait $(cat /tmp/kvs_dahua_dome2.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_dahua_dome2.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > dahua-dome2-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x dahua-dome2-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./dahua-dome2-to-kvs.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./dahua-dome2-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Dahua 2MP Bullet 1

```
#!/usr/bin/env bash
set -euo pipefail
# ----------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
```

```
# ------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="Dahua-2MP-Bullet-1-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.YTOYlpGjviLB.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                  # 5 min early
# ------------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                    # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:5 \
gst-launch-1.0 -v -e \
  souphttpsrc location="$HLS_URL" \
```

```
  ! hlsdemux \
  ! decodebin \
  ! videoconvert \
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
  ! video/x-h264,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Dahua Bullet-1 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/dahua-bullet1-gst.log &
    echo $! > /tmp/kvs_dahua_bullet1.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
```

```
        export_credentials "$CREDS"

        if [[ -f /tmp/kvs_dahua_bullet1.pid ]] && kill -0 $(cat
/tmp/kvs_dahua_bullet1.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_dahua_bullet1.pid))..."
            kill $(cat /tmp/kvs_dahua_bullet1.pid)
            wait $(cat /tmp/kvs_dahua_bullet1.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_dahua_bullet1.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main

# 1. Save the script
cat > dahua-bullet1-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x dahua-bullet1-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./dahua-dome1-to-kvs.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./dahua-bullet1-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Dahua 2MP Bullet 2

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
STREAM_NAME="Dahua-2MP-Bullet-2-Stream"
```

```
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.cLVBa3vAV4yf.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                  # 5 min early
# ---------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                    # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:5 \
gst-launch-1.0 -v -e \
  souphttpsrc location="$HLS_URL" \
  ! hlsdemux \
  ! decodebin \
  ! videoconvert \
```

```
  ! x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 \
  ! video/x-h264,profile=baseline \
  ! kvssink stream-name="$STREAM_NAME" \
            storage-size=512 \
            aws-region="$AWS_REGION" \
            access-key="\${AWS_ACCESS_KEY_ID}" \
            secret-key="\${AWS_SECRET_ACCESS_KEY}" \
            session-token="\${AWS_SESSION_TOKEN}"
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Dahua Bullet-2 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/dahua-bullet2-gst.log &
    echo $! > /tmp/kvs_dahua_bullet2.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"
```

```
        if [[ -f /tmp/kvs_dahua_bullet2.pid ]] && kill -0 $(cat
/tmp/kvs_dahua_bullet2.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_dahua_bullet2.pid))..."
            kill $(cat /tmp/kvs_dahua_bullet2.pid)
            wait $(cat /tmp/kvs_dahua_bullet2.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_dahua_bullet2.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > dahua-bullet2-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x dahua-bullet2-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./dahua-bullet2-to-kvs.sh > kvs.log 2>&1 &   # start in
background
tail -f kvs.log                                    # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./dahua-bullet2-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```

Eufy C210

```
#!/usr/bin/env bash
set -euo pipefail
# ---------------------------------------------------------------------
# CONFIGURATION – edit only these 5 lines
# ---------------------------------------------------------------------
ROLE_NAME="LiveStreamRole"
```

```
STREAM_NAME="Eufy-C210-Stream"
HLS_URL="https://625238ff6b2b.eu-west-1.playback.live-video.net/api/vi
deo/v1/eu-west-1.766885402413.channel.zTdf3y3iuyU8.m3u8"
AWS_REGION="eu-west-1"
REFRESH_BEFORE_EXPIRY=300                  # 5 min early
# ---------------------------------------------------------------

METADATA_URL="http://169.254.169.254/latest"
TOKEN_TTL_SECONDS=21600                    # 6 h

get_token() {
    curl -sX PUT "$METADATA_URL/api/token" \
         -H "X-aws-ec2-metadata-token-ttl-seconds: $TOKEN_TTL_SECONDS"
}

get_credentials() {
    local token="$1"
    curl -s -H "X-aws-ec2-metadata-token: $token" \
         "$METADATA_URL/meta-data/iam/security-credentials/$ROLE_NAME"
}

export_credentials() {
    local json="$1"
    export AWS_ACCESS_KEY_ID=$(jq -r .AccessKeyId <<<"$json")
    export AWS_SECRET_ACCESS_KEY=$(jq -r .SecretAccessKey <<<"$json")
    export AWS_SESSION_TOKEN=$(jq -r .Token <<<"$json")
    export AWS_REGION="$AWS_REGION"
    export CRED_EXPIRY_EPOCH=$(date -d "$(jq -r .Expiration
<<<"$json")" +%s)
}

build_gst_cmd() {
    cat <<EOF
GST_DEBUG=kvssink:5 \
gst-launch-1.0 -v -e \
  playbin uri="$HLS_URL" \
    video-sink=" \
      videoconvert ! \
```

```
      x264enc tune=zerolatency bitrate=2000 speed-preset=superfast
key-int-max=30 threads=4 ! \
      video/x-h264,profile=baseline ! \
      kvssink stream-name='$STREAM_NAME' \
              storage-size=512 \
              aws-region='$AWS_REGION' \
              access-key=\"\${AWS_ACCESS_KEY_ID}\" \
              secret-key=\"\${AWS_SECRET_ACCESS_KEY}\" \
              session-token=\"\${AWS_SESSION_TOKEN}\" \
    "
EOF
}

start_pipeline() {
    echo "[$(date)] Starting Eufy C210 → KVS (expire $(date -d
"@$CRED_EXPIRY_EPOCH"))"
    bash -c "$(build_gst_cmd)" &>/var/log/eufy-c210-gst.log &
    echo $! > /tmp/kvs_eufy_c210.pid
}

main() {
    TOKEN=$(get_token)
    CREDS=$(get_credentials "$TOKEN")
    export_credentials "$CREDS"
    start_pipeline

    while true; do
        NOW=$(date +%s)
        SLEEP=$(( CRED_EXPIRY_EPOCH - REFRESH_BEFORE_EXPIRY - NOW ))
        (( SLEEP > 0 )) && {
            echo "[$(date)] Sleeping ${SLEEP}s until refresh..."
            sleep "$SLEEP"
        }

        TOKEN=$(get_token)
        CREDS=$(get_credentials "$TOKEN")
        export_credentials "$CREDS"
```

```
        if [[ -f /tmp/kvs_eufy_c210.pid ]] && kill -0 $(cat
/tmp/kvs_eufy_c210.pid) 2>/dev/null; then
            echo "[$(date)] Graceful restart (PID $(cat
/tmp/kvs_eufy_c210.pid))..."
            kill $(cat /tmp/kvs_eufy_c210.pid)
            wait $(cat /tmp/kvs_eufy_c210.pid) 2>/dev/null || true
        fi
        start_pipeline
    done
}

trap 'echo "[$(date)] Shutdown → killing pipeline"; kill $(cat
/tmp/kvs_eufy_c210.pid) 2>/dev/null || true; exit' SIGINT SIGTERM
main
```

How to Use It

```
# 1. Save the script
cat > eufy-to-kvs.sh << 'EOF'
# [PASTE FULL SCRIPT HERE]
EOF
```

then

```
chmod +x eufy-to-kvs.sh
# 2. Run it (as the same user that has access to the IMDSv2 endpoint)
nohup ./eufy-to-kvs.sh > kvs.log 2>&1 &   # start in background
tail -f kvs.log                                  # show the log live
```

One-Liner (Start + Live Logs)

```
nohup ./eufy-to-kvs.sh > kvs.log 2>&1 & tail -f kvs.log
```