

# GROK CONTRIBUTION TRACKING FIX - November 20, 2025

## 🎯 Problem Diagnosed

### Observation from Screenshot:

-  Badge shows: “ Verified by Grok (98% confidence)” - **Grok IS working!**
-  Committee panel shows: “Grok-4.10/12(0%)” with “30 offline” - **Not tracking properly**

### Root Cause:

Grok verifier was successfully calling the xAI API and returning verified results, BUT the `VerifierAgent` wasn't recording contributions to the `LLMCommitteeService`. This created a visibility mismatch:

- The verification badge showed success (from actual Grok API responses)
- The committee panel showed offline (no contribution records in database)

## ✓ Fixes Implemented

### 1. Grok Pricing Correction (Commit: `61c84d1`)

**Issue:** Grok costs were 10x too high

```
// BEFORE (WRONG)
'grok-4': {
  inputPer1k: 0.002,    // $2.00 per 1M tokens ✗
  outputPer1k: 0.010,   // $10.00 per 1M tokens ✗
}

// AFTER (CORRECT - per xAI Nov 2025 pricing)
'grok-4': {
  inputPer1k: 0.0002,  // $0.20 per 1M tokens ✓
  outputPer1k: 0.0005, // $0.50 per 1M tokens ✓
}
```

**Impact:** Users were being charged 10x more for Grok verifications than actual cost!

### 2. LLM Committee Contribution Tracking (Commit: `b6773e7`)

**Changes to `Verifier.agent.ts`:**

#### 1. Injected Committee Service

```
typescript
constructor(
  private llmService: LLMService,
  private truthMycelium: TruthMyceliumService,
  @Optional() private committeeService: LLMCommitteeService | null, // NEW
) {}
```

#### 2. Record Successful Verifications

```
typescript
// After successful verification in verify() method
if (this.committeeService) {
```

```

        await this.committeeService.recordContribution({
            session_id: messages[0]?.conversation_id || 'unknown',
            model_name: verifiedData.model || 'grok-4',
            agent_name: 'verifier',
            contributed: true,
            offline: false,
            tokens_used: verification.tokensUsed?.total || 0,
            cost_usd: verifiedData.cost || 0,
            latency_ms: verifiedData.latency || 0,
        });
        this.logger.log(`📝 Committee: Grok-${verifiedData.model} contribution recorded`);
    }
}

```

### 3. Record Failed Attempts

```

typescript
// In catch block
if (this.committeeService && messages[0]) {
    await this.committeeService.recordContribution({
        session_id: messages[0].conversation_id || 'unknown',
        model_name: 'grok-4',
        agent_name: 'verifier',
        contributed: false,
        offline: true,
        error_type: error.message,
    });
}

```

### 4. Added Post-Synthesis Tracking

- Same contribution logging for `postSynthesisCheck()` method
- Agent name: `'verifier-post'` to distinguish from main verification

## 🔍 How Tracking Works Now

---

### Data Flow:

1. **User asks question** → Pipeline starts
2. **Verifier calls Grok API** → `llmService.verifyWithGrok()`
3. **Grok responds** → Parse verification data
4. **Record contribution** → `committeeService.recordContribution()`
5. **Database stores** → `LLMContribution` entity saved
6. **Committee endpoint** → `/api/committee/session/:id` aggregates stats
7. **Frontend displays** → Committee panel shows correct percentages

### What Gets Tracked:

- Model name (`grok-4`, `grok-4-1-fast-reasoning`, etc.)
- Agent name (`verifier`, `verifier-post`)
- Success/failure (`contributed: true/false`)
- Online/offline status (`offline: true/false`)
- Tokens used (input + output)
- Cost in USD (actual xAI pricing)

- Latency in milliseconds
- Error messages (if failed)

## Expected Behavior After Deployment

### Before (Current):

```
LLM Committee:
- Grok-4.10/12(0%) [30 offline]
- GPT-5.29/12(42%) [1 offline]
```

### After (Fixed):

```
LLM Committee:
- GPT-5.29/12(42%) [1 offline]
- Grok-4/12(17%) [0 offline] ← NOW VISIBLE!
- Direct-Claude/12(58%) [1 offline]
```

### Logs Will Show:

```
✓ Verifier complete - confidence: 0.98, facts: 3, cost: $0.0012, latency: 1250ms,
model: grok-4
👉 Committee: Grok-grok-4 contribution recorded
🍄 Mycelium grew by 3 verified facts
```

## Deployment Instructions

### Step 1: Verify Code on GitHub

- Branch: `main`
- Latest commit: `b6773e7` - “Fix: Add Grok contribution tracking to LLM Committee”
- Files changed:
  - `src/config/llm.config.ts` (pricing fix)
  - `src/agents/verifier.agent.ts` (contribution tracking)

### Step 2: Deploy to Render

1. Go to: <https://dashboard.render.com/>
2. Click: `vctt-agi-backend` service
3. Click: “**Manual Deploy**” (top right)
4. Select: “**Clear build cache & deploy**”
5. Click: “**Deploy**”
6. Wait: ~5-10 minutes for build

### Step 3: Verify Environment Variables

While deploying, check:

1. **Environment** tab on Render
2. Confirm `XAI_API_KEY` is set
3. If missing → Paste your xAI API key → Save → Redeploy

## Step 4: Test After Deployment

1. Go to: <https://vcttagi-kernal13-peters-projects-3a28ae0e.vercel.app>
2. Ask: "What is VCTT-AGI?" (triggers verification)
3. Check Committee panel:
  - Should show Grok-4/12 (~17%)
  - Offline count should be 0
4. Open browser DevTools → Console
5. Look for logs:

 Committee: Grok-grok-4 contribution recorded

## Local Testing (Optional)

### Test xAI API Key:

```
curl https://api.x.ai/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_XAI_API_KEY" \
-d '{
  "model": "grok-4",
  "messages": [{"role": "user", "content": "Say hello in 3 words"}],
  "max_tokens": 20
}'
```

### Expected Response:

```
{
  "id": "chatmpl-....",
  "object": "chat.completion",
  "created": 1732147200,
  "model": "grok-4",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello there friend!"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 12,
    "completion_tokens": 5,
    "total_tokens": 17
  }
}
```

### If you get 401/403:

1. Go to: <https://console.x.ai/>
2. Regenerate API key
3. Update Render environment variable
4. Redeploy



## Valid Grok Model Names (Nov 2025)

Per xAI documentation:

- 'grok-4' - Stable flagship (alias)
- 'grok-4-0709' - Stable flagship (dated version)
- 'grok-4-1-fast-reasoning' - Grok 4.1 Thinking mode
- 'grok-4-1-fast-non-reasoning' - Grok 4.1 Fast mode
- 'grok-4-fast-reasoning' - Grok 4 Thinking
- 'grok-4-fast-non-reasoning' - Grok 4 Fast

**Current config:** 'grok-4' (valid and recommended)



## What This Fixes

1.  **Badge + Committee Sync** - Both show Grok is online and working
2.  **Accurate Cost Tracking** - 10x cheaper (\$0.20/\$0.50 per M tokens)
3.  **Transparency** - See exact Grok contribution % in committee panel
4.  **Failure Visibility** - If Grok goes offline, it's logged properly
5.  **Performance Metrics** - Track latency, tokens, cost per verification
6.  **Truth Mycelium Growth** - Verified facts stored and attributed correctly



## Future Upgrades

When you want to upgrade to **Grok 4.1** (requires SuperGrok subscription):

```
// In llm.config.ts, change:  
verification: 'grok-4-1-fast-reasoning', // Grok 4.1 Thinking mode  
  
// Or for faster responses:  
verification: 'grok-4-1-fast-non-reasoning', // Grok 4.1 Fast mode
```

**Pricing stays the same:** \$0.20/\$0.50 per M tokens



## Troubleshooting

### Symptom: “Grok still showing offline”

#### Check:

1. Render deployment logs for "Grok API error"
2. Environment variable `XAI_API_KEY` is set
3. Test API key with curl command above
4. Check xAI account has credits/access

### Symptom: “Badge works but committee still shows 0%”

#### Fix:

1. Clear browser cache (Ctrl+Shift+R)
2. Check database has `LLMContribution` records:

```
sql  
SELECT * FROM llm_contributions WHERE model_name LIKE 'grok%' ORDER BY timestamp DESC LIMIT 10;
```

3. Verify committee endpoint returns data:

bash

```
curl https://your-backend.onrender.com/api/committee/session/YOUR_SESSION_ID
```

## Symptom: “Costs too high/low”

### Verify pricing in config:

- Grok: \$0.0002 input, \$0.0005 output (per 1k tokens)
- GPT-4o: \$0.0025 input, \$0.010 output (per 1k tokens)



## Status: Ready to Deploy

---

- Pricing fixed (10x error corrected)
- Contribution tracking added
- Error logging implemented
- Post-synthesis tracking included
- Code pushed to GitHub (main branch)
- Waiting for Render deployment
- Waiting for user testing

**Next:** Deploy to Render → Test Vishnu query → Verify committee panel shows Grok!