Dungeons & Dragons Meets AI

From Hallucinations to Spec-Driven Systems on AWS

David Poindexter · Cloud Architect

CleanSlate Technology Group

AWS Community Day 2026

LLMs Are Fun Until They Need to Be Trusted

- LLMs generate confident, creative text
- But confidence ≠ correctness
- How do you trust output when the model can invent anything?

"What would it take for you to trust an AI response in production?"

D&D as a Trust Test

- D&D requires preserving: rules, entities, lore, scene order, player choices, continuity
- One wrong name = broken immersion
- One invented faction = broken trust

"D&D is not the gimmick. D&D is the test harness."

If an AI can't run a 10-minute D&D scene consistently, why trust it with customer data?

The Scene - Golgoron Arises

- City of Arches: a fantasy city with portal archways
- Garland Willowmane asks the party to welcome a new arrival peacefully
- She carries a welcome basket. The Golden Knights can't arrive in time.
- What happens: stirges → cultist → Golgoron (confused devil) → player choice
- The best outcome is social, not combat

14 named entities. 12 scene steps. Strict order.

Prompt-Only Failure

Player input: "We approach the archway."

"GOLGORON THE DESTROYER erupts from the portal -
wreathed in hellfire, wielding twin vorpal blades!"

"The Elder Dragon Mythraxis..."

"The old wizard Thaddeus Greymane steps forward
before being vaporized."

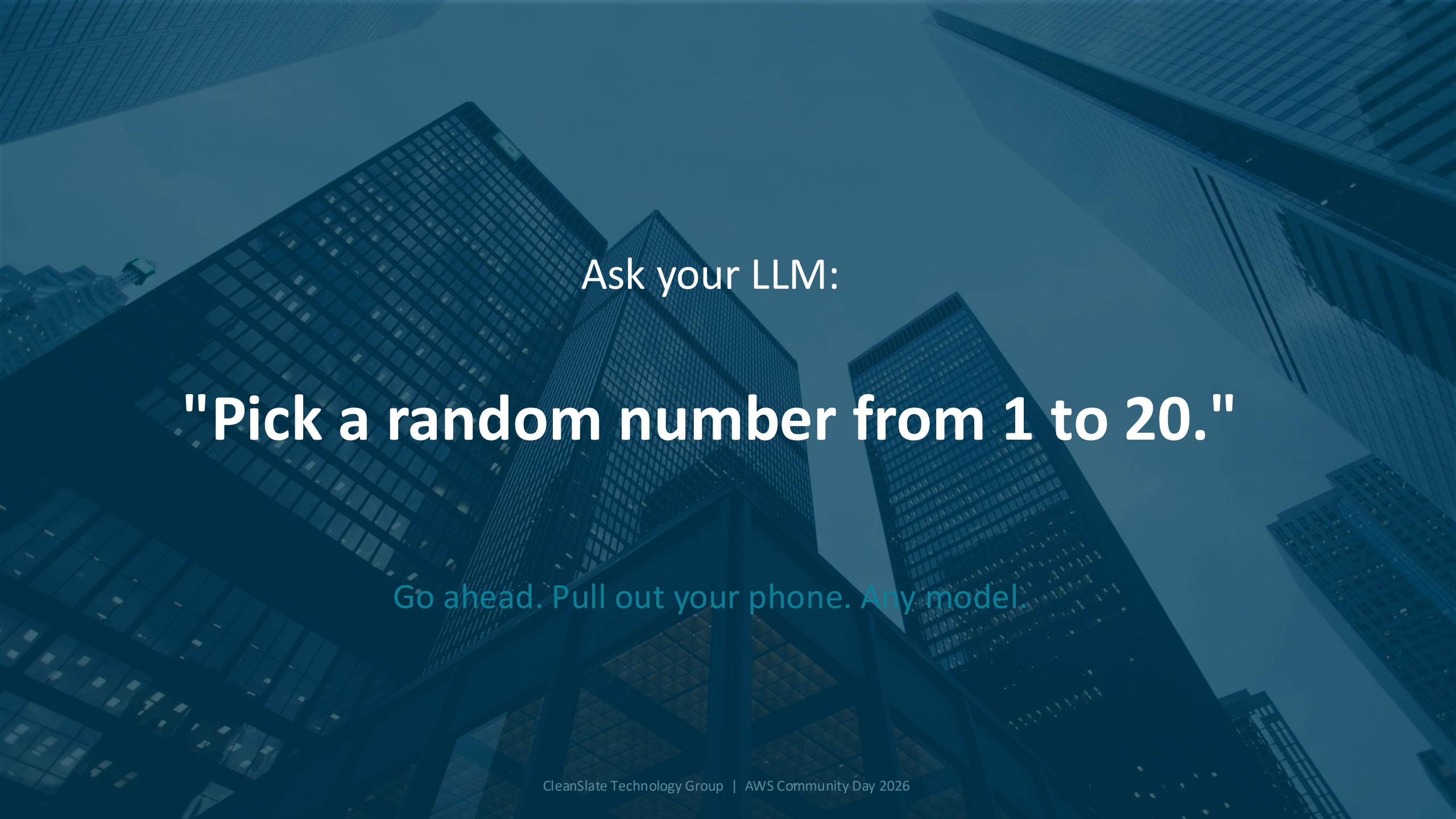
"This is a fight to the death. There is no escape."

11 errors • 5 validation rules violated • The model is creative, confident, and completely wrong.

What Went Wrong

1. No source material → model fills gaps with hallucinations
2. No structure → free-form prose, unverifiable
3. No validation → errors pass silently
4. No state tracking → scene order ignored

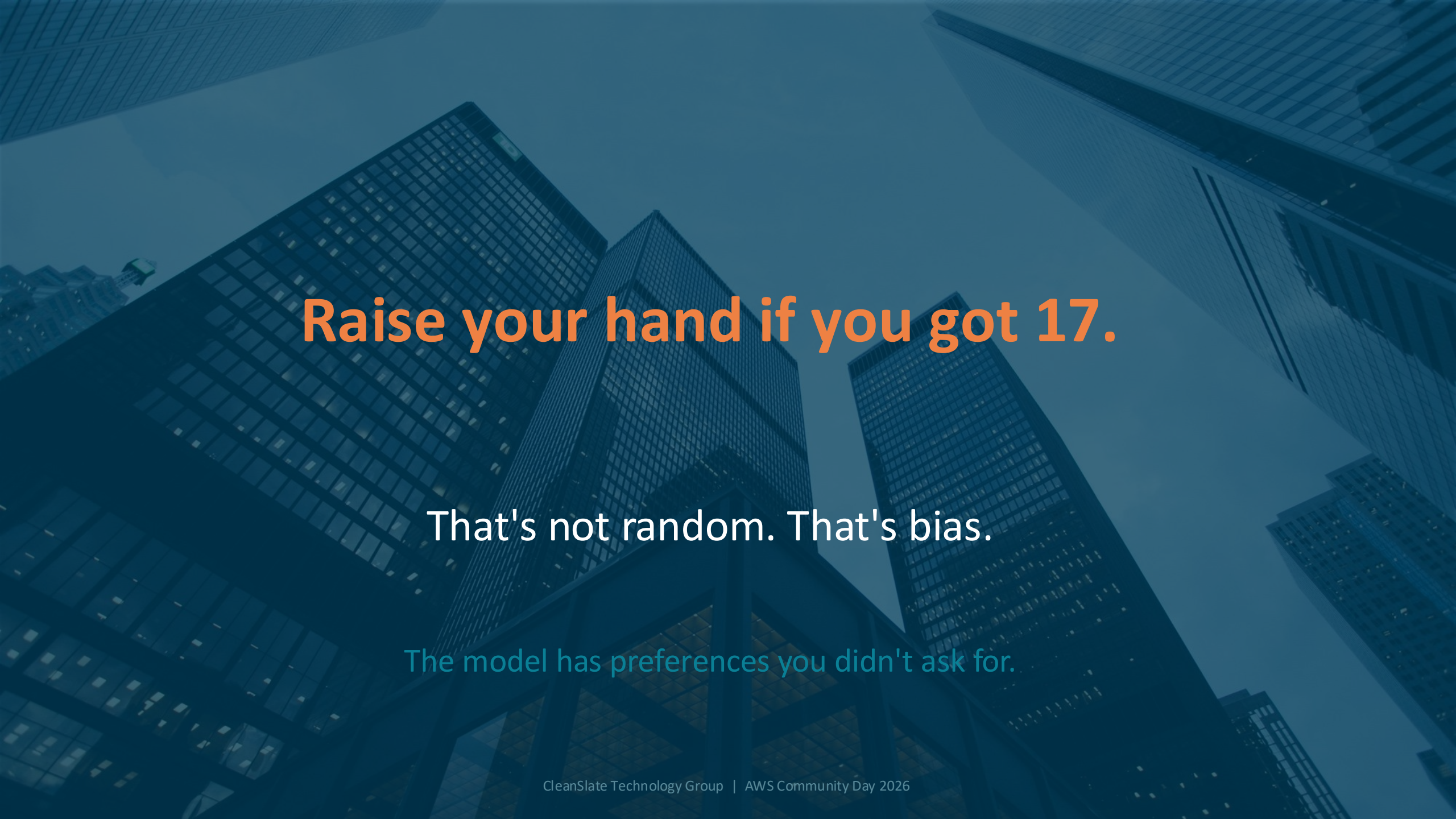
The fix is not "one better prompt" - it's a system boundary.



Ask your LLM:

"Pick a random number from 1 to 20."

Go ahead. Pull out your phone. Any model.


Raise your hand if you got 17.
That's not random. That's bias.
The model has preferences you didn't ask for.

The Spec-Driven Workflow

1. Source material - versioned, extracted, summarized (single source of truth)
2. Steering files - 6 rules defining what the AI can and cannot do
3. Feature spec - requirements, acceptance criteria, structured output schema
4. Structured output - JSON with required fields, not free-form prose
5. Validation - 8 programmatic rules, pass/fail in milliseconds

Source → Steering → Schema → Validation → Observable Output

Spec-Driven Response

```
{
  "scene_summary": "The party follows Garland Willowmane toward the archway...",
  "dm_response": "Garland leads you toward a rough stone archway...",
  "canonical_facts_used": [
    "Garland Willowmane leads the party to the archway",
    "Stirges emerge first from the archway and attack"
  ],
  "player_options": [
    "Fight the stirges", "Protect Garland",
    "Look for other threats", "Call out to take cover"
  ],
  "source_grounded": true
}
```

← traceable

← agency

← verifiable

Validation - Before and After

Spec-Driven Response

- ✅ Valid JSON
- ✅ source_grounded is true
- ✅ canonical_facts_used not empty
- ✅ Garland Willowmane not renamed
- ✅ Golgoron not automatically hostile
- ✅ Welcome basket preserved
- ✅ No unsupported entities
- ✅ Player options provided

8/8 rules passed

Prompt-Only Response

- ✅ Valid JSON
- ❌ source_grounded is true
- ❌ canonical_facts_used not empty
- ✅ Garland Willowmane not renamed
- ❌ Golgoron not automatically hostile
- ❌ Welcome basket preserved
- ❌ No unsupported entities
- ❌ Player options provided

2/8 rules passed

Same input. Same rules. Engineering vs vibes.

Why State Matters

Each d20 result introduces new state the system must track:

Roll	Event
1–5	The stirges scatter into the crowd
6–10	Garland drops the welcome basket
11–15	A player spots the cultist early
16–20	Golgoron pauses and listens

Prompt-only forgets it. Spec-driven preserves it.

Mapping the Pattern to AWS

Source material	→	**Amazon S3 (versioned)**
Orchestration	→	**AWS Lambda**
Model inference	→	**Amazon Bedrock**
Structured output	→	**JSON schema enforced**
Validation	→	**Lambda + rules**
Observability	→	**Amazon CloudWatch**

"The model is not the system. The model is one component inside a system."

Takeaways

1. LLMs need system boundaries to be trustworthy
2. Source-ground everything - if it can't cite the source, it's hallucinating
3. Structure your outputs - JSON schemas make validation possible
4. Validate programmatically - simple checks, milliseconds
5. Spec-driven workflows define boundaries before code
6. The pattern maps from local prototype to AWS production

One source document. One schema. One validation rule. Start there.

Thank you. • David Poindexter • CleanSlate Technology Group

CleanSlate TECHNOLOGY GROUP

Scan to connect

David Poindexter

Cloud Architect

CleanSlate Technology Group

Thank you - keep the d20.