GitLab

Building (and Keeping) GenAI Teams

You'll need your best people, but that's the easy part

Michelle Gill
Head of DevOps Engineering, GitLab

What it's like to lead teams from the bleeding edge

Like **conducting an orchestra** where every musician is a maestro who wants to play their own composition. You have to balance between **brilliant individual contributions** and forcing **consensus on a direction**, all while the tempo of the industry **accelerates relentlessly.**

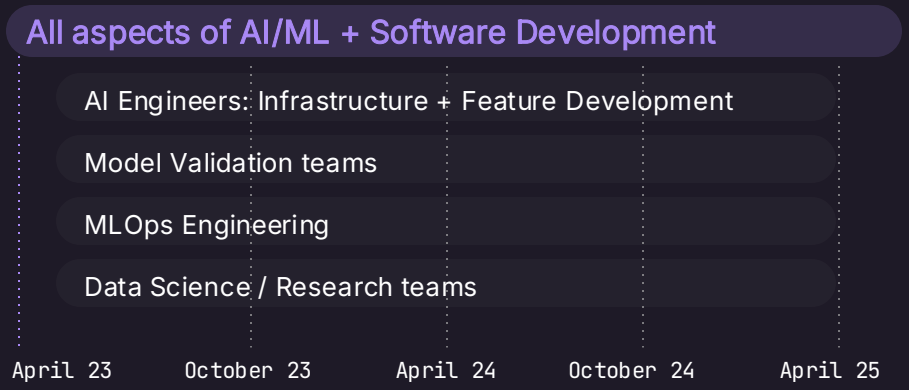
Who am I to say?

Michelle Gill

Experience managing Data Science (ML, AI) and Platform Engineering teams

Why me?

History of forming, scaling, and leading platform teams. As a leader, I lean into a challenge and create cultures of adaptability.



This applies to you if...

You have or will have an "AI Platform" or internal tooling team

This looks like a team that centrally manages models for your other teams in some way (API Gateway, model knowledge, provider partnerships).

You employ 1 or more of the AI/ML roles

If your projects have reached a level that benefits from a dedicated "AI Engineer" or "ML Engineer" role being on the team.

What do you do first?

Step 1

Identify the talent you need

Put your best people on the case!

The ideal team player for what you need has one or a combination of these traits:

- Natural curiosity (to continue learning)
- Grit (to keep going)
- Technical versatility across ML, AI, and/or software engineering

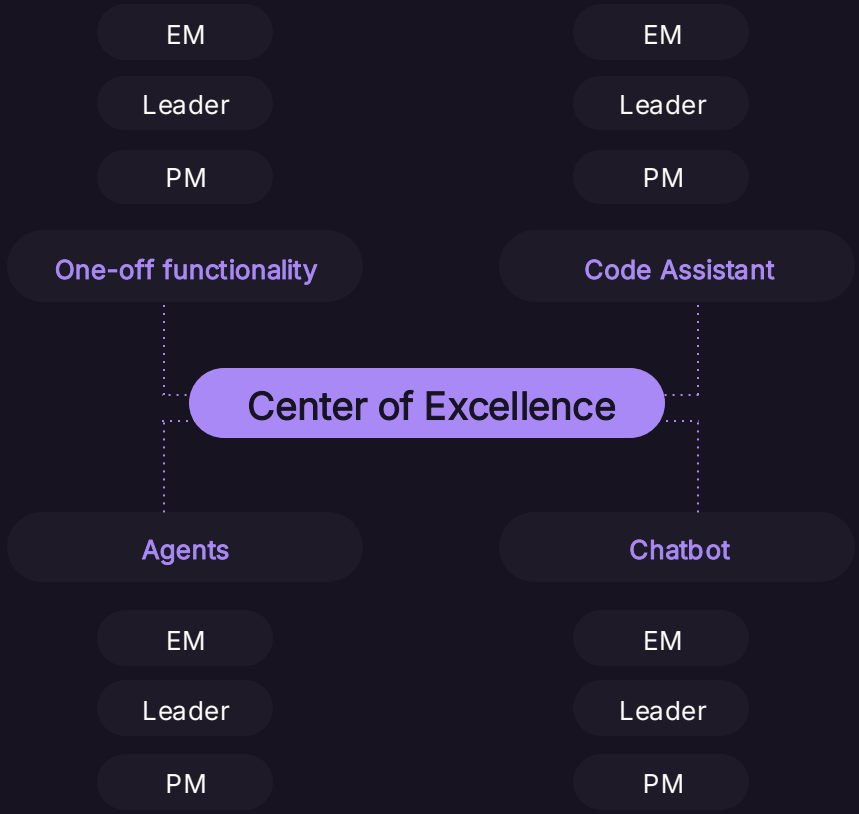
Step 2

Form your center of excellence

- Central team to provide programmatic access to models
- Subject matter experts who understand LLM nuance
- Up to date on all latest trends and best practices

This framework powers other teams in building...

- Code assistants
- Chatbots
- One-Off feature functionality
- Agents
- Guidance to others



Finally the team is ready

Assuming things went "to plan", you might now be in problematic territory...

Problems achieving consensus?

There's an increase in competing solutions and political reasons (e.g. promo, self-taught, etc).

Need to accelerate a project?

Pulling from your center of excellence could leave your platform unstable over time.

Architecture proposals come in weekly for the same subjects as technologies are experimented with broadly.

Supporting roles like PM, EM are unclear how to support their team within an evolving landscape.

Several problems, but primarily 2 root causes...

The Brilliant Mind Dilemma

More expertise = more correct solutions = more discussions.
Ten brilliant people, ten different frameworks for "good enough."

The ultimate irony is that a team of average engineers with clear leadership often ships faster than a room full of brilliant minds struggling to agree on the optimal approach. **These are the people you should have picked to be on your AI team.**

Pro	Con
Depth of expertise and innovative thinking, curiosity to challenge.. a lot	Strong opinions, disagreement loops, and parallel solutions

The impact of a Brilliant Mind

Week 1
Support team delivers working prototype with embeddings

Week 2
ML team proposes RAG architecture instead

Week 3
Knowledge Graph discussion begins (more use cases!)

Week 5
Someone told Infrastructure

Week 8
Still debating complexity while competitor ships

Tension between trusting your experts and the unforgiving pace of innovation

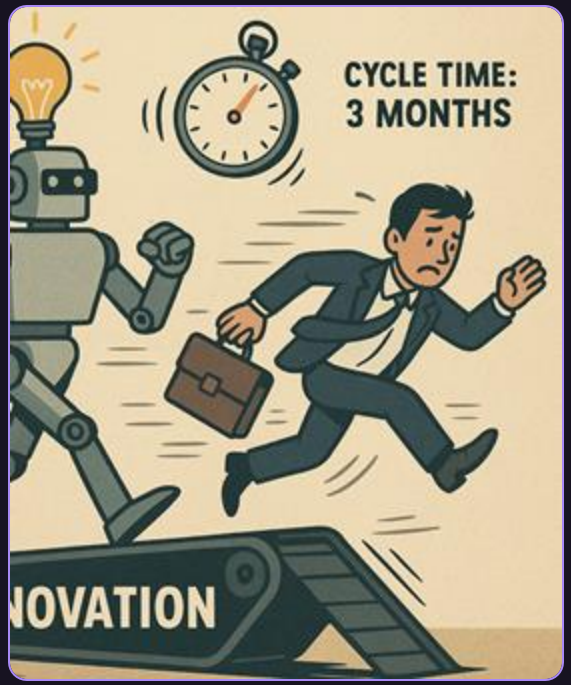
Unforgiving Pace of Innovation

2-month delay = competitive disadvantage

Blueprints, rearchitectures, rewrites, new features: These are not worth doing if they can be replaced in 2 months with an advancement.

Sometimes good people must "move out of the way"

No time to performance manage underperformers, those who can't keep pace, stay updated, aligned.



Goodbye, respectfully

The problem: An underperforming team is responsible for guidance critical to delivery

Jonathan's traditional approach: Set expectations, give feedback, invest in his struggling team

The result: They improved steadily. But the AI industry moved exponentially. Engineers waited for decisions. Features stalled. Competitors shipped.

Despite good intentions, his team's need for coaching was actively causing delays with delivery.

The brutal reality: AI doesn't care about fault or effort—only results and timing.

It doesn't have to be this way, there are tactics that work

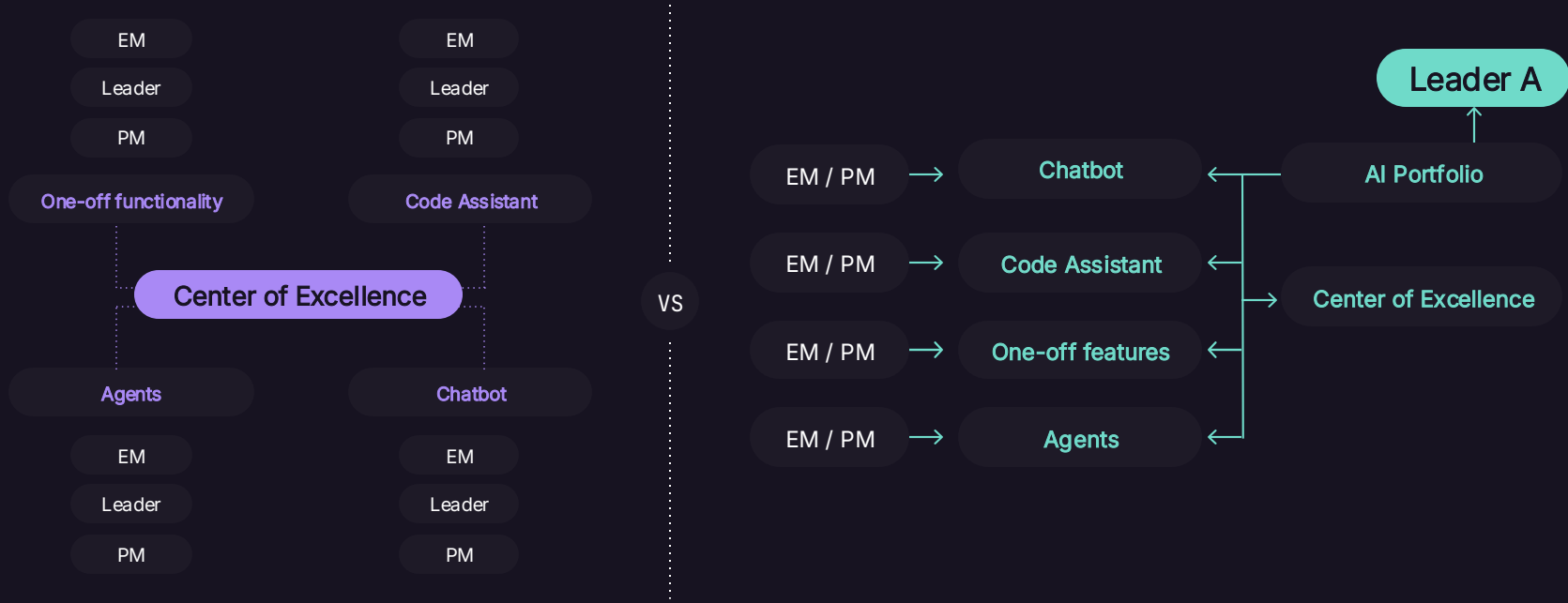
Flatten organizational structures for faster decision-making which will eliminate some of the bureaucracy teams might face.

Timelines shorten alongside the pace of innovation: Use this to balance "when to uplevel talent" vs. "when to move on" as the impact can be felt in numerous ways when the decision is delayed.

Set unapologetically high standards for these teams who will need to deliver on time with the pace of innovation and adapt to the ever-evolving rewrite opportunities.

Manage multi-disciplinary experts with strong opinions with tools like decision frameworks, data, and **support them** in the ability to achieve consensus amongst peers.

Flattened organization



A few (specific) ways we're working smarter

Promote faster decision making

Designate decision owners who have final say after input is gathered - align duplicate ideas through this person, timebox discussions, and have success criteria for evaluating ("better than it was").

e.g. 1 SME fielding 16 proposals on context

Use only Evidence to pivot (vs 'newness')

- Test competing theories by rapidly prototyping
- Set clear metrics for success before experiments begin ("better than it was")
- Be happy to fail fast and move on

Separate Ideas from Execution

- Commit to 1 direction for a fixed period after making a decision
- Questioning the approach is temporarily suspended or should be happening in parallel aligned to success criteria
- In reality, Theory competes with Theory - use time wisely

Meet experts where they're at

- Ask ML experts to quantify their concerns (e.g., "How many percentage points of accuracy will we lose?")
- Ask engineers to provide concrete examples rather than theoretical edge cases
- Request product experts to prioritize user stories that would be directly impacted

Finally, retain them

Keep them engaged

Barriers like a slower velocity (lack of vision, debates), unappealing product ideas, or a lack of competitiveness on pay and benefits could cause attrition.

Career progression and pay changes

Higher compensation due to increased complexity and AI expertise

Continuous learning

Don’t let them lose the edge that landed them on the team in the first place… your team needs dedicated time to focus on upcoming opportunities and advancements in the space.

Here's the help I'm looking for

If you've been successful in integrating GenAI teams as traditional SWE teams...

OR

If you have strong opinions about how DevOps will evolve in the industry...

I want to talk about it!