



In Praise of “Normal Engineers”



# Charity Majors

CTO | Honeycomb

 [linkedin.com/in/charity-majors](https://www.linkedin.com/in/charity-majors)

 [@charity.wtf](mailto:@charity.wtf)



# The curious durability of the (rather ridiculous) “10x engineer” meme

 **Shekhar Kirani** @skirani · Jul 11, 2019

1. 10x engineers hate meetings. They think it is a waste of time and obvious things are being discussed. They attend meetings because the manager has called for a "Staff meeting" to discuss the features and status.

40 177 646

 **Shekhar Kirani** @skirani · Jul 11, 2019


3. 10x engineers laptop screen background color is typically black (they always change defaults). Their keyboard keys such as i, f, x are usually worn out than of a, s, and e (email senders).

115 405 494

 **Shekhar Kirani** @skirani · Jul 11, 2019

5. Most of the 10x engineers are full-stack engineers. For them code is code, they don't care whether it is front-end, back-end, API, database, serverless, etc. I have rarely seen them doing UI work.

86 246 794

 **Shekhar Kirani** @skirani · Jul 11, 2019

7. 10x engineers rarely look at help documentation of classes or methods. They know it in memory and can recall from memory. They write code at the same ease as writing English. No breaks, no pause, just type.

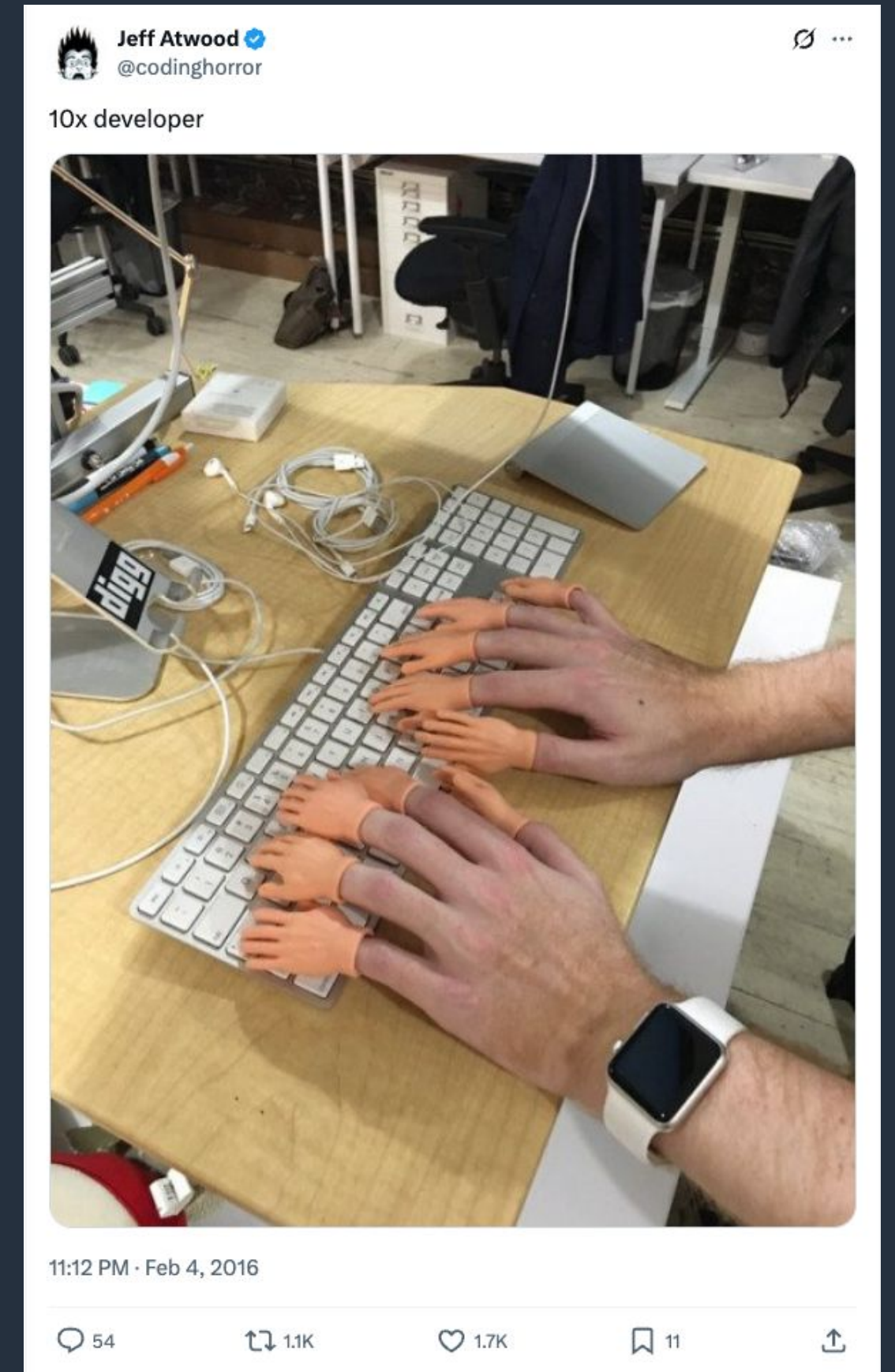
84 278 405

 **Shekhar Kirani** @skirani · Jul 11, 2019

9. 10x engineers are poor mentors as they can't teach others on what to do OR parcel the work. They always think "It takes too long to teach or discuss with others, I would rather do it myself." They are also poor interviewers.

84 305 603

If you've ever met anyone  
who **self-identifies** as a  
"**10x engineer**"...  
they were probably an  
**asshole.**

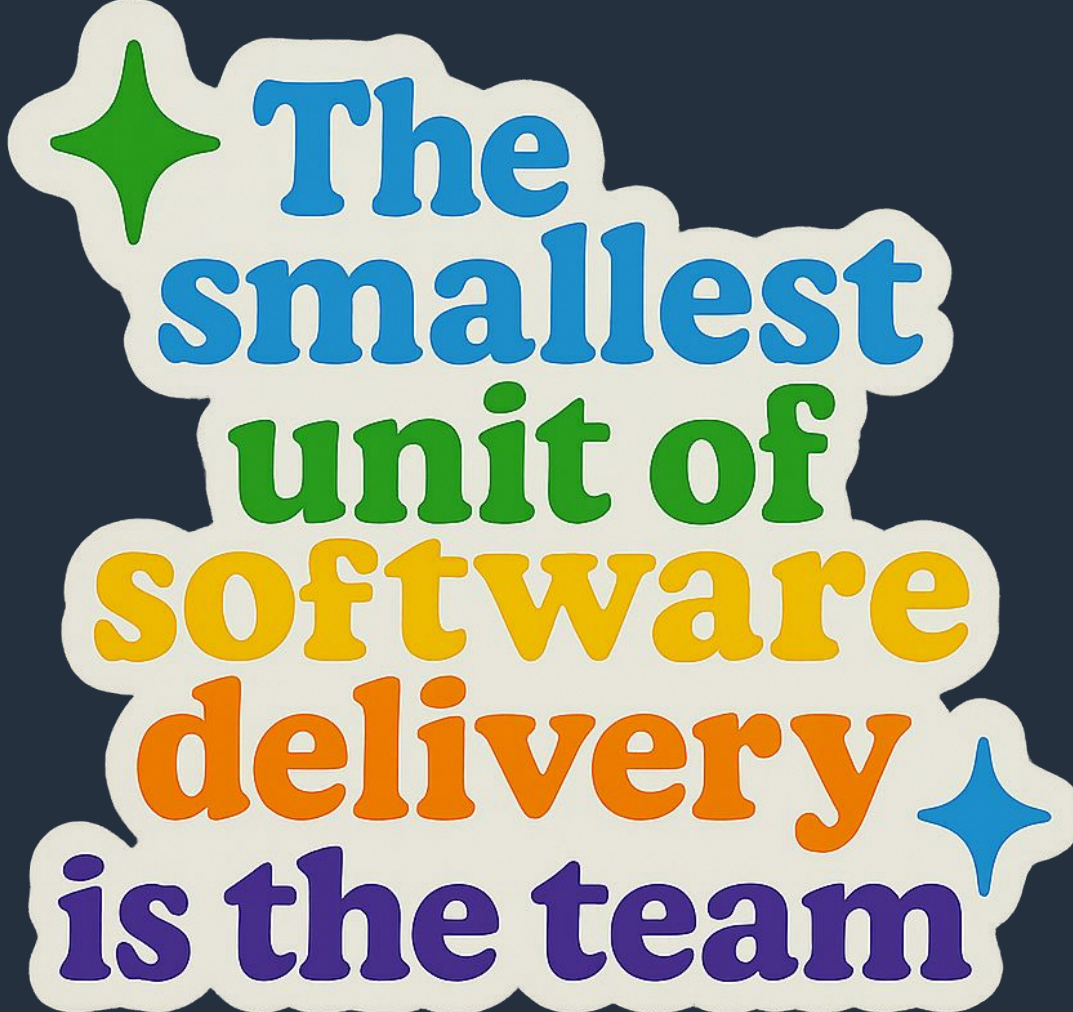




# Engineers don't own software, engineering teams own software.

If it takes the slowest engineer five hours to ship a single line of code, it will take the **fastest** engineer five hours to ship a single line of code.

Generating code is, and has always been, the easiest part of the software development lifecycle.



✦ The  
smallest  
unit of  
software  
delivery ✦  
is the team

Actually, we *do* have a term for individual engineers who own services or surface area:



Software engineering is about **solving business problems using technology**, not writing tons of code.

The only measure of productivity that matters is **business impact**.

When people talk about world class engineering orgs, they tend to obsess over levels and pedigrees.

This is **exactly backwards**.





# The best engineering orgs are the ones where **normal engineers** can do **great work**.

A truly great engineering org is one where normal, workaday software engineers can consistently **move fast**, **ship code**, **respond to users**, **understand the systems they've built**, and **move the business forward**, a little bit more, every single day.

# It is a huge competitive advantage if you can draw from a broader talent pool

So many folks out there bragging about how they hire the top 10% (or top .1%!!!) of talent.

If that's your strategy, I hope you're prepared to offer them something in the top 10% (or .1%) of salary bands.

December 16, 2024

## How Coinbase hires the top 0.1% talent

Most companies say they only hire the best. Coinbase actually means it, and they've structured their talent strategy to live up to it.

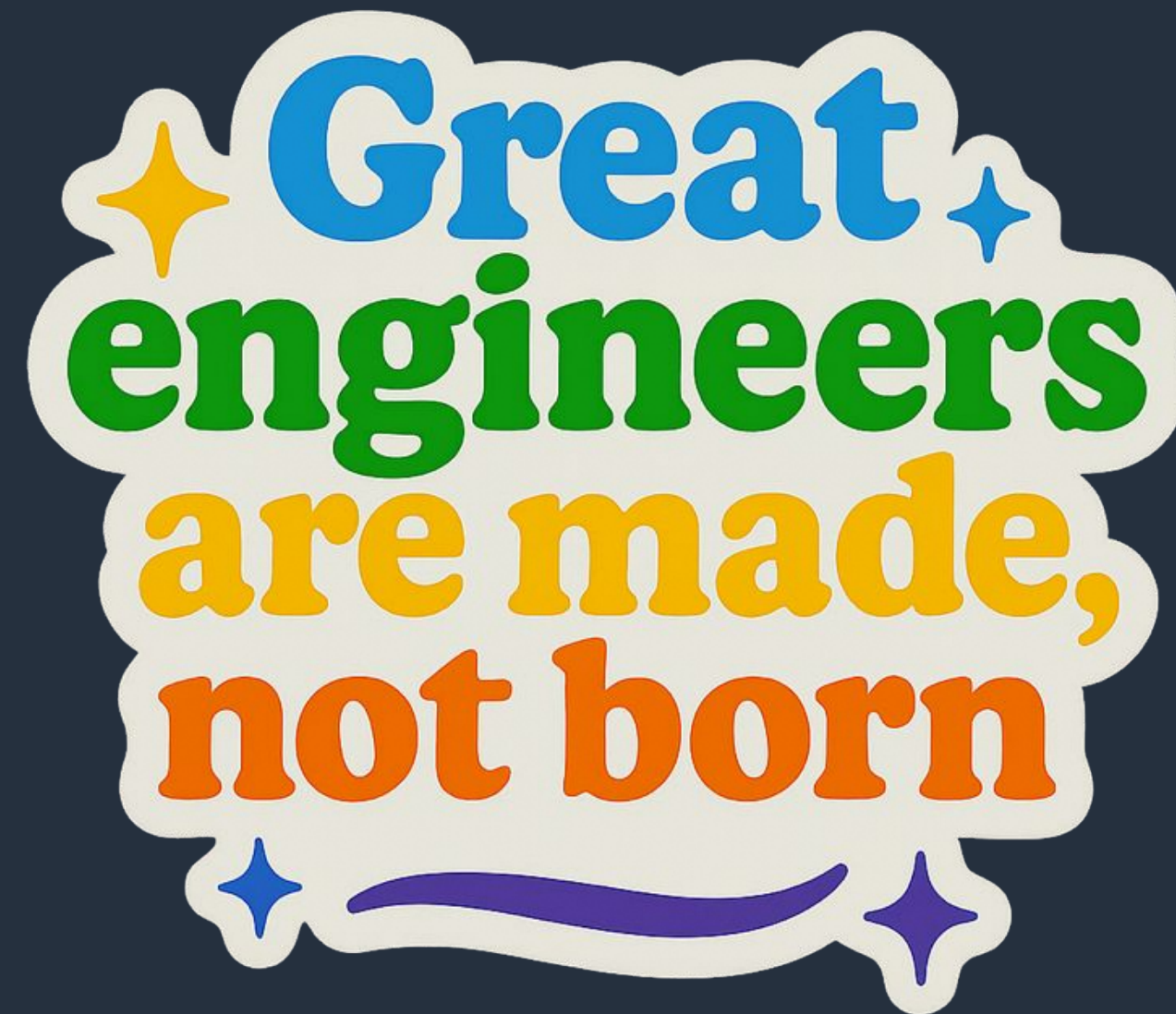
# Let's talk about “normal” for a moment.

It can be humbling to think of ourselves as normal people, but most of us are in fact pretty normal people, with many years of highly specialized practice and experience.

“Normal” encompasses a wide spectrum of neurodiversity and neurospiciness. Even those of us who are certifiable geniuses on some criteria are likely quite normal in other ways — kinesthetic, emotional, spatial, musical, linguistic, etc.

We are ALL more normal than we are not. 💜

Nobody is **born** a great software engineer;  
**great engineers** are **forged** over  
years and years of practice.





# Build your sociotechnical systems with “normal people” in mind

Normal people have cognitive biases — confirmation bias, recency bias, hindsight bias. We care, we try, we work hard; we also develop habits and get fatigued. Our eyes are drawn to the color red.

Our emotional state can affect the quality of our work. Our relationships can affect our ability to get shit done.

When your systems are designed to be used by normal engineers, all their excess brilliance can get poured into the product, instead of wasting it on navigating the system itself.

How do you turn **normal** **engineers** into  
**10x engineering** **teams**?



# 10x engineering teams

The shorter the interval, the lower the cognitive carrying costs.

A long, laggy feedback loop can turn into a “**software engineering death spiral**”, where lots of changes get batched up and rolled out infrequently, and deploys are scary and treacherous.

Deploy cycle time is the feedback loop at the ❤️ of every sociotechnical system.

1. Keep the deploy interval short and sweet

# 10x engineering teams

Make it easy to do the right thing, hard to do the wrong thing.

The **fastest** way to ship a single line of code should also be the **easiest** way to ship a single line of code.

Treat your platform like a product. Engineers are people too, so wrap designers and design thinking into all the touch points engineers have with production systems.

1. Keep the deploy interval short and sweet
2. **Easy to do the right thing**



# 10x engineering teams

Every engineer should be able to deploy their own code, figure out if it is working as intended or not, and if not, roll forward or back, swiftly and easily.

(Better yet, use **feature flags** to toggle code on and off without needing to deploy.)

Keep in mind that people will be engaging with production late at night or when they are stressed, tired, and possibly freaking out. Make it simple and foolproof.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. **Fast and simple rollbacks**

# 10x engineering teams

Tests are great, staging can be helpful, but **only production is production.**

Good, friendly sociotechnical systems **invest heavily** in instrumentation, observability, and tools for sense-making.

Being able to visualize your work is what makes engineering abstractions accessible to actual engineers. You shouldn't have to be a world-class engineer just to debug your own damn code.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. Fast and simple rollbacks
4. **Invest in observability**

# 10x engineering teams

If fast, safe, automated deploys with guard rails, instrumentation, and fast, efficient test suites are “everybody’s job”, they will end up **nobody’s job**.

Engineering productivity isn’t something you can outsource.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. Fast and simple rollbacks
4. Invest in observability
5. **Invest in internal tooling and enablement**

# 10x engineering teams

Learning is the norm, growth is the baseline.

People do their best work when they feel a sense of belonging.

An inclusive culture is one where **everyone** feels safe to ask questions, explore, and make mistakes; where **everyone** is held to the same high standard, and **everyone** is given the support and encouragement they need to achieve their goals.

This isn't about being "squishy". Manage underperformers out decisively.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. Fast and simple rollbacks
4. Invest in observability
5. Invest in internal tooling and enablement
6. **Build an inclusive culture**



# 10x engineering teams

Yes, a team of senior engineers with a shared background can move incredibly fast. But a **monoculture is fragile**.

Someone gets sick, someone gets pregnant, you start to grow and you need to integrate people from other backgrounds and the whole team can get derailed — fast.

When your teams are used to operating with a mix of genders, racial backgrounds, identities, age ranges, family statuses, geographical locations, skill sets, etc — when this is just table stakes, standard operating procedure — you're better equipped to roll with life when it happens.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. Fast and simple rollbacks
4. Invest in observability
5. Invest in internal tooling and enablement
6. Build an inclusive culture
7. **Diverse teams are resilient**

# 10x engineering teams

The best engineering teams aren't the ones stuffed with the most staff engineers and principal engineers.

The best engineering teams are ones where nobody is running on autopilot, banging out a login page for the 300th time; everyone is working on something that challenges them and pushes their boundaries.

Everyone is **learning**, everyone is **teaching**, everyone is pushing their own boundaries and **growing**. All the time.

1. Keep the deploy interval short and sweet
2. Easy to do the right thing
3. Fast and simple rollbacks
4. Invest in observability
5. Invest in internal tooling and enablement
6. Build an inclusive culture
7. Diverse teams are resilient
8. **Assemble a range of levels**

# Great engineering orgs with 10x engineering teams actually mint world-class engineers

Places where engineers can ship fast, get shit done, and have a lot of impact are a magnet for top performers.



# If you already have world-class engineers in your org, good for you!

Your role as a leader is to leverage their brilliance for the good of your customers and your other engineers, without coming to *depend* on their brilliance.

After all, these people *don't belong to you*. They may walk out the door at any moment, and that has to be okay.

Tech companies obsess over finding, hiring and keeping world class engineers. But they over-index on finding these people AFTER they've already developed into world class engineers. They overlook *the majority of the talent out there*.

Talent may be evenly distributed, but opportunity is not.



I am NOT saying that **excellence** doesn't matter,

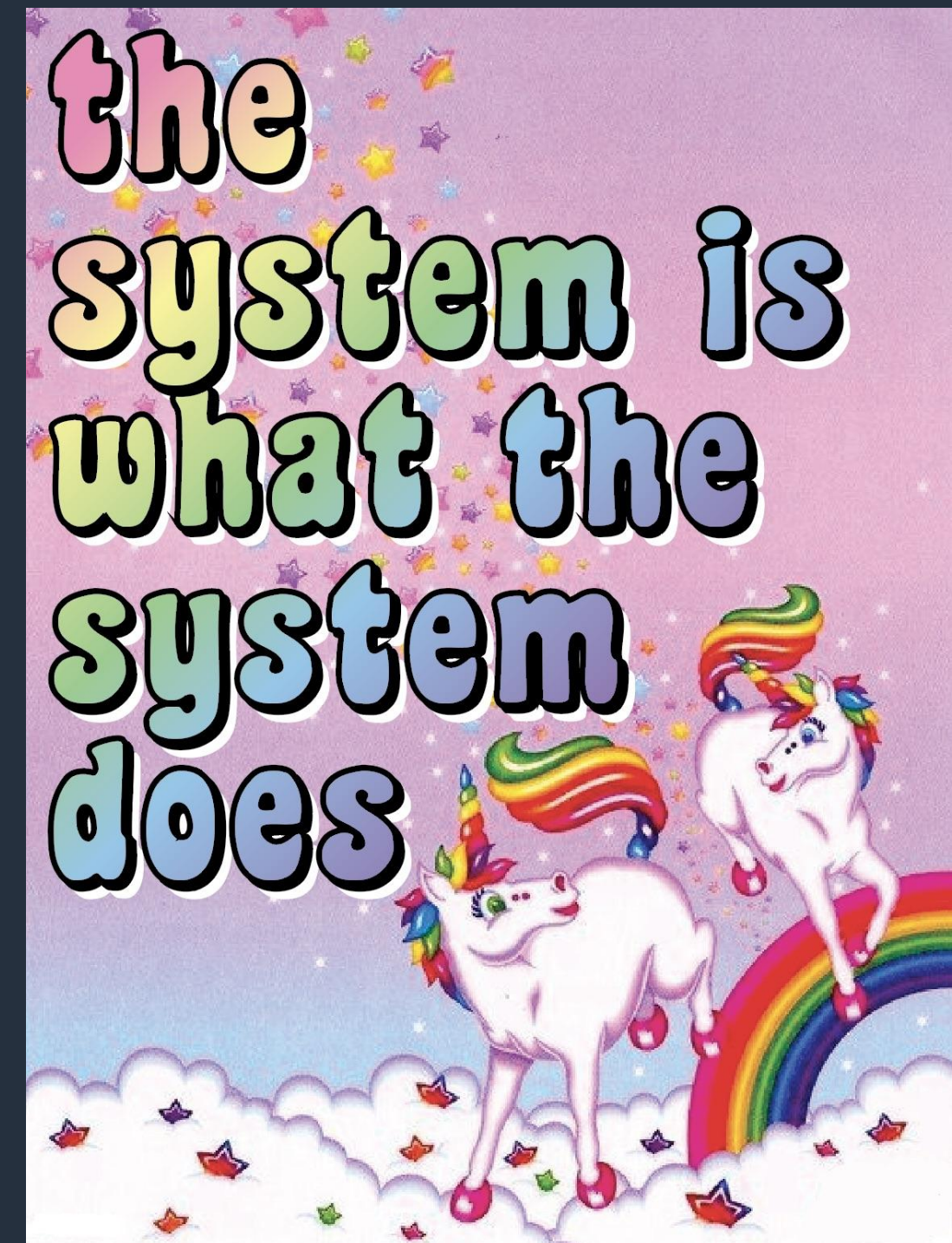
or that you should manage to the “**lowest common denominator**,”

or that you should never manage people out,  
or that you should not have **high standards**.

# I am saying that your company is a

Who will people become in the time they spend on your team?

Any work you put into building a system where engineers are in the driver's seat and can self-serve deploys, instrumentation, and fast feedback loops; where experimentation and learning is baseline, where feedback loops are fast and curiosity is rewarded.. will pay off over and over again.



# What kind of engineer does your system forge?





I am saying that **talent** is as **common** as **dirt**;  
the potential for excellence is everywhere.

Be an organization that values continuous  
learning and development, curiosity and  
experimentation, and invests in its talent.

Don't hire the “best” people.  
Hire the right people.





Be the kind of place that builds and sustains  
**high performing teams**, and the next  
generation of **world class engineers**.



Be the kind of company they don't want to  
leave. 



See everything. Solve **anything.**

honeycomb.io

