What "The Best" Looks Like

> *Talent hits a target no one else can hit. Genius hits a target no one else can see.*
>
> *—Arthur Schopenhauer*

The second that the next round of funding hits the bank, every new CTO starts obsessing over the same thing: *Who the hell do I hire next?* The answer is surprisingly non-obvious. You're told that you always want—scratch that, *need*—the best of the best, your startup's future depends on it. You're told your company is unique, special, and it requires the most hardcore among researchers, designers, engineers, and product managers. You never skimp on who builds the golden goose. You can't succeed with any less than that.

But, is that *actually* true?

Every startup-hustle YouTube video, VC podcast and celeb founder interview regurgitates that the key to success is to hire the very best people, no matter what it takes. And yet you live in the real world, with real-world constraints. You have only so much money in the bank. Only so much time and bandwidth to hire. Only so much attention in a given day. And you're not alone in your hunt. You're competing with hundreds of other players in your geography, industry, and problem space looking for "the best." Many of them have a more famous brand, more cash, more promising equity, more charming founders, and maybe even a high production value promo video showcasing happy employees, rare wood office counters and a shoes-off policy.

Will you actually hire the best of the best against those odds? Many years ago I found myself in this pickle and I had to learn all the relevant lessons the hard way. I share these lessons here, so that you don't have to struggle through that same maze yourself. My pain is your gain.

My first time around the startup world in 2012, I hardly knew what I was doing and relied mostly on luck—and unfortunately, firing—to end up with a team I could be proud of. I had no real point of reference for greatness, for what "the best" in our area could look like, and building that model required lots of experimentation, with high highs and many low lows.

The story of David, our brilliant infrastructure ops engineer at Freckle, has stayed with me ever since. David applied to the company back when we were only hiring for an ops role. We were growing slowly, so there was zero room for anybody who wasn't absolutely essential on the team. We had no open-ended extra seat for a smart person who just happened to be on the market, unlike some companies these days.

David was 100% what we were *not* looking for. He had never done any ops. In fact, he had never done anything related to web dev or product engineering. I'm not sure he even knew what AWS was at that point. He was a sharp physics guy working with a professor on simulations in an academic context. He didn't let that deter him. He wanted to work at Freckle, thanks to our reputation as one of the few software startups in the world using Haskell in production at scale. We were an odd outlier in a sea of buggy and laborious Rails apps, a shelter for people who didn't want yet another web slop gig. And Haskell was oh-so-hot in the Hacker News programming language theory space at the time, a technology attracting software nerds obsessed with correctness and new, better ways of developing bug-free apps.

I immediately told David that he was not a fit; he had none of what we were looking for. And yet he persisted, emailing me that he would do whatever test project we threw at him, and if he bombed it, no problem, he would go away. But if he nailed it, we would have proof that he was qualified, in spite of what his CV indicated. Fine. I sent him a meaty cloud ops take-home project, expecting to never hear from him again. Importantly, this was in the days before you could have Claude Code slap that together for you in two prompts.

A day or two later, he returned the project to us, and it was pretty much flawless, doing even more than we had asked for. That was not expected. I got curious about what else he could do. We weren't drowning in applicants anyway, so I figured I didn't have much to lose. We took him through the usual interview process. He was humble, optimistic, well-spoken, actively communicating and taking feedback well, eager to get to work. He was pumped about everything he could learn on the job, about the doors that would eventually open if he nailed it. He didn't have much experience, as someone who had never written commercial software before, but he was really quick to absorb everything we threw at him.

We gave him a chance. As predicted, he was stellar, and we had a really good run with him until he moved on to a much more illustrious employer. A few years later he became a senior principal engineer at Stripe, going from a physics lab, then a starry-eyed K-12 software startup, to being a big deal at one of Silicon Valley's finest firms. An unsurprising path for one of "the best."

While building the different companies I worked at, I've run several times into "Davids" who ended up with meteoric career trajectories, sometimes already pedigreed in all the right ways, sometimes completely invisible to everyone but the trained eye of a CTO looking for gold. What is special about someone like that? And how can you, with your humble hiring budget, identify them before their value is obvious to everybody else in the market?

With David, it was obvious that I had lucked my way into finding him. He had to badger me into seeing what he was capable of, at a time when I was only looking for obvious signs of success. Later on at my game studio, thanks to a brutally skewed job market, I had the total pick of the litter during

Hiring Summer and could select one or two stellar engineers from hundreds of perfectly reasonable applicants. There was hardly any competition, with money drying up all around the industry. Again, I found a diamond in the rough. But this time I had the right knowledge and strategies in place to end up with the kind of team I wanted with far less reliance on luck. Even as I was drowning in hundreds of resumes that all started to look the same, “the best” candidate was still in there, like a needle in a haystack. This time, I felt like I was actually skilled enough to find that hidden gem, not simply crossing my fingers and relying on luck alone. That’s a feeling of empowerment I would like you to experience as well.

What do we mean by “the best”?

Of course, when every company says they have the best people, the math doesn’t work out. And of course, if they regularly have to fire staff, something doesn’t add up again. One day you end up landing a job at one of these companies and realize that the braggadocio was hardly reflective of reality, but then again, it’s all part of the startup founder LARP that requires you play the part. “Our team is okay, you will probably like working here, we sometimes know what we’re doing” would be a much more accurate depiction, but you’ll never find that quote on the careers page.

Regardless, you want to be a good Boy Scout CTO and live up to the lofty expectations set for you by the Silicon Valley elders, and at least try to hire “the best of the best” as they say in all of the fireside chats.

But then, a question naturally emerges: what exactly do we mean by *the best*? The accolades? Pedigrees? Github stars? Hacker News front pages? Job title? Celebrity employers? Number of former YCombinator companies worked at? Hackathon wins? Typing speed? Clout among other Rust developers wearing fuzzy animal costumes? That all sounds good, *I guess*, but is that all relevant to your company?

It turns out that “the best” is mostly subjective and specific to your situation. The culture, the vibe, the industry, the processes of your company, and the technical choices will all influence who will be a phenomenal addition to your tribe. That person might not be the same individual bringing in a million or two a year at Meta. In fact, that highly pedigreed, highly connected individual might be a net negative for your company, even though they might, at least on paper, seem the most high-end among candidates. They may blow the mind of a recruiter at Netflix, but they may not want to schlep through all of the messiness and chaos of an early stage company still trying to define itself. They’re not the best *for you*, and they’re likely not the best for startups, but plenty of other people are. Your job is to determine what those traits are and where to find people who have them.

Universally Best Startup Hires

Your company, timing, industry, problem space and founding team personalities are unique, which is why a generic blog post or a book could never tell you exactly what hires are optimal for you.

However, experience shows that there are many universal winning commonalities between great hires that will be applicable regardless of your early stage company's unique DNA. I identify 11 of them below.

The 11 traits:

- Non-obviousness
- Not-too-junior
- Not-too-senior
- Hunger
- Humility
- High EQ
- Team-centricity
- Competence
- High-agency problem-solvers
- Cross-disciplinary empathy
- Tolerance of uncertainty

As a side note, I found Patrick Lencioni's framework from *The Ideal Team Player* to survive first contact with both pre-seed and Series A+ realities. It's one of those dry-as-dust HBR book club management self-help guides that get an eyeroll by your median 20-year-old founder going through YCombinator, but in my experience it confirmed and put a simple model around something I had personally seen emerge again and again in the field.

Lencioni identifies three essential traits: *hunger*, *humility* and *smarts*, the latter of which should have been called EQ all along. You'll see them referenced below. The three criteria may seem obvious at a glance, but having a simple framework to work with as you're making hundreds of decisions makes a big difference. Simple things done right turn out to be pretty darn powerful.

Let's take a closer look at the 11 commonalities among great startup hires I've identified. Some of them are simply statistically likely to occur due to the nature of startup hiring. Others are ones you should straight up filter for when you're doing your search.

Non-obviousness

The best hires in the early stages are usually non-obviously good to the untrained eye. They don't look as appealing to employers with infinite resources who otherwise would have already hired them. If they were obviously incredible, it would be unlikely you—at your startup—would ever talk to them.

One of the most important skills of a startup CTO trying to hire an amazing team is the ability to uncover awesome talent that others have overlooked. These candidates get skipped often due to being too off-the-beaten-path and under-pedigreed compared to the obvious picks that every other company is making. Often these candidates are bad at marketing themselves, don't know where to look, and are off-putting in some way to the naive search. You have an opportunity to take advantage of that oversight.

Realistically, someone who looks like the perfect FAANG candidate who has won all of the math olympiads, had all of the stellar internships, and went to the top CS schools will either:

a. get a big cash dump from YC and start their own thing, or
b. go work at one of the career-building brand names in the Valley and make a monstrous amount of money.

The karmic wheel is just about cycling between those two until that person either gets into VC or becomes an exec at a prestigious firm years later. If that's what the universe has in store for a candidate, taking a chance on a no-name team with 12 months in the bank, poor development practices and sloppy management is a tough proposition.

That leaves a pretty large pool of "everybody else" who didn't pattern-match. How you go about sifting through them is a big topic I will leave for another chapter, but the key is to look for signals that are less obvious than a Stanford degree and an OpenAI internship. Often that looks like a large volume of work, expertise in unsexy niche areas of technology, an intense work ethic driven by curiosity, and many others.

A Stanford CS degree is no guarantee someone will be a phenomenal contributor *at your company*, but it's a reasonable proxy of future potential for large employers with long time horizons that allow them to invest in coaching and nurturing their junior staff. That's not in the cards for a pre-revenue startup that will run out of money next year.

Not-too-junior

The best people for your startup will most likely be senior, or at least mid-stage contributors with several solid years of experience. Too senior isn't great either, you don't need the large scale cross-team mature product jousting chops. There's likely no staff-level work for them to do, and staff-level engineers aren't just faster-typing seniors.

As an early stage startup, until you have enough lead developers, senior product managers, senior designers etc. to set the standard, hiring fresh-out-of-school contributors is a gamble. Junior staff are still developing their taste, judgment, ability to work in a team, and understanding of how to follow the process and when to deviate from it. Without an adult in the room you're in cat-herding territory. This isn't to say that you won't get work done this way, but you could have had someone else in that seat who required less supervision. And you're at a stage where every seat is mission-critical and the opportunity cost is significant.

While it might not matter too much as you throw prototype spaghetti at the wall in the early days, as soon as you have something worth maintaining and complexity rises, you're now in a race against time. Someone must actively garden the complexity of the work and be ultimately responsible for it, and that someone will tend to be not fresh-out-of-school. Sure, a green dev *can* care about complexity. But most haven't yet lived through a soul-crushing, multi-week refactor fueled by years of tech debt, the kind of preventable trauma that earns you "character-building scars." If they're learning those lessons on your watch, they're doing it at the expense of your delivery timeline and your sanity.

I'm fond of mid-stage software developers who demonstrate terrific chops, hunger to learn and prove themselves. Yes, they require more hand-holding, but usually the volume of work they put out, and their openness to re-do it, if needed, is worth betting on. You should be able to manage a couple of them yourself as the CTO in the early days, and soon enough you'll have other senior people on the team who will pick up the mentorship torch.

Not-too-senior

This also means that the "best people" to hire at this stage will tend to be earlier in their careers. The longer great people are on the market, the more likely they are to be identified by the market as being great. Your job as a startup CTO is to find them before others do. Once the market has found them and has actively started rewarding them, they will be out of your price range.

Now, it's worth mentioning that in specific well-capitalized niches of the startup market, companies are now able to pay a decent chunk of change thanks to larger VC rounds and quicker time-to-revenue. Thus, the old school notion that BigCo always pays best may not hold true as often. Here's a quick sample of many Work at a Startup roles in December 2025. Not quite at FAANG-level, but far from starvation.

Category	Count	Mean	Median	Min (Midpoint)	Max (Midpoint)
Regular SWE	45	**$157.4K**	$150.0K	$85.0K	$275.0K
Senior SWE	32	**$209.8K**	$202.5K	$140.0K	$350.0K
Founding SWE	33	**$189.3K**	$185.0K	$125.0K	$255.0K

That also means that, unless your company grows fast or shows tremendous potential for the yet-unvested equity, you won't get a ton of time with your great early career hires. They will quickly accumulate valuable experience and prove themselves to be stellar, and move on to a prestigious employer with longer time horizons, a great brand name, and a stupendous paycheck that you will not be able to match. Being an L5-7 at BigCo pays a pretty penny, with none of the unpredictability of startup equity and saner hours, but none of the adventure and camaraderie of a pirate ship. For employees later in their careers, that's not a bad tradeoff. Once they have vested with your company, it makes sense for them to diversify their equities, as they already own one lottery ticket.

Being on the receiving end of a reference call for one of your star employees, while they're still working at your company, is always bittersweet: on one hand you want what's best for them, to level up and grow in their career. On the other, they're *your* discovery—dang it—and you know you won't be able to counter-offer the type of employer they can now attract. Or they decide to start their own company, emboldened and informed by the experience at yours.

In the end, that’s okay. Sometimes a great hire’s growth trajectory is faster than a startup’s, and you can be grateful for having given them a chance to prove themselves and to find a new path that they wouldn’t have otherwise. That kind of hire will have likely made a major difference to your company and having trusted alumni out there in the world doing great things should be a source of pride and good industry karma for any CTO who discovered them.

Seniority here also unfortunately often correlates with age. The best startup employee will usually be someone early in their career who doesn’t have as many responsibilities or as much need for consistency due to having more dependents. They may have fewer immediate cash flow constraints, fewer “adult responsibilities.” Kids need braces and karate classes, and if Mom is doing 996 at a ten-person company paying her peanuts, offering a crappy health care plan, promising an epic payout ten years from now, that’s a real mismatch. Startups are an extreme sport, and generally inadvisable for anybody who’s not in a safe position to speculate on their career for several years.

Hunger

The best hires are self-driven, stoked by the ability to learn, to gain mastery of the craft, and by being able to show off their accomplishments to the rest of the team they want to impress. When you look at their track record, even when it’s not filled with household-name accolades and pedigrees, you will typically see breadcrumbs of exploration, toy projects and experimentation fueled by curiosity. It’s that classic desire to hack things and understand them.

You don’t need to ask them to work a little harder, they know this is a big growth spurt for them and they want to take advantage of it. The company is giving them a valuable practice canvas for their skills and they want to make the most of this opportunity, which unfortunately isn’t that common. They love a challenge, they want to prove themselves, they love the work and they are excited by being around others who will push them to be their best. They would have been hacking on something interesting either way, but now you’re actually paying them to do it.

They’re curious about how everything works: your company, the startup world, the industry they’re diving into, other disciplines, the tech they’re using. They’re learning, absorbing. Maybe they want to start their own thing one day, or maybe they’re set on using this gig as a ramp to something else they dream about. Maybe they just want the new optionality granted by having your company on their resume. They’re sponges and will gladly take on the challenges you send their way.

I remember throwing a big Redux experiment at David in what must have been his first month at work. Again, no web dev experience prior to this. Nobody on the team knew the technology and he, hired as an infrastructure engineer with a background in physics and zero product engineering experience, had something up and running for us within days. He was *pumped* he got to do something so radically new for the team as one of his first assignments. Ultimately we decided the tech wasn't a fit for our existing codebase, we scrapped the experiment, and he ran off to the next challenge with no loss of enthusiasm.

At Double Dusk we could not figure out why players would regularly de-sync their character positions on the server when using a custom—and wildly complicated, due to needing to replicate Half Life 2-style movement to the smallest detail inside of Unreal Engine—character movement component in combination with our own Unreal Engine networking tweaks, making character movement authoritative on client instances of the game. A big deviation from the defaults.

Our newly-hired star engineer, us being *his first ever employer*, had to dive into the pit of madness. He almost gave up several times before finally identifying all of the spots in the physics sub-stepping logic that was causing the drift across the network. I'm confident I myself would have gone completely bananas trying to debug that one. He was proud of the trust we had put in him, he was eager to learn, he didn't want to let the team down, and I knew enough about him to make the bet that the challenge would have been just barely within his abilities.

Hunger is a massive force-multiplier, and unlike many other traits, it doesn't seem to be teachable. Failures will instill humility. Rejection will hone your EQ. But you're either driven to compete, learn and prove yourself, or you're not. No manager's pep talk will kickstart that drive in you, at least not sustainably. The hubris has to come from somewhere much deeper.

Humility

Startups are all about rapid experimentation, trying countless ideas that don't work, and bootstrapping the skills and processes of the company as it figures out just what exactly people will pay for. Trust-building is critical in any team endeavor, and members who are unable to admit fault, who take up all of the space, and/or who need to be right at all costs will be a real problem as the company matures.

The best hires are humble and will happily talk about both their victories and their biggest implosions, about what they learned from their misadventures, about talented past coworkers and how their efforts were part of a team. They rarely claim to have single-handedly carried everything on their backs.

As you interview candidates searching for the best, many non-obvious great hires will be bad at behavioral interviews—or interviews in general, for that matter. They didn't get extensive training on answering questions using the STAR method, didn't drill stock responses to predictable and tired interview questions. They're early-ish stage hands-on technology people, not smooth-talking execs jumping from one boardroom to the next. It is your job to fish for key nuggets of insight buried in the rough presentation. After you interview enough people with a similar set of standard questions, you'll be able to spot the outliers in the bell curve of responses.

An underrated aspect of humility in startups is detachment from one's work. Companies in the process of inventing themselves need to actively cycle through many ideas, most of which will ultimately not stand the test of time. *Design for the trash can* and *kill your darlings* are important mindsets in creative work, and they extend far beyond music, film and games. Quantity leads to quality, and a great hire will accept regularly needing to apply a flamethrower to their work and try again with the new learnings from the latest experiment. They will not be precious about it, nor will they feel diminished by needing to try again. The Davids do not keep score; they keep trying and do the work knowing that there's a good chance it will not go anywhere.

That is also highly correlated with their ability to take feedback without taking it personally. They're not afraid of criticism and quick feedback loops. In fact, they seek disconfirmation sooner rather than later. They have excellent feedback metabolism.

High EQ

Building technology companies is a team sport. Everything you do is with the help of other people, for the benefit of other people. As much as the startup world fetishizes the cracked ninja jedi 10x code-poet, if that person makes others never want to talk to them again, cracked they are not. Someone who is able to work with others' quirks, understand how they operate and what they respond to, and act in a way that brings the best out of their teammates is worth their weight in gold. Brilliant jerks, as great as they are for producing riveting drama in TV shows, quickly become a net negative in the real world.

We're in the squishy human feeling territory here (aka "soft skills") and you can't LeetCode your way into knowing if someone will be an ass. There's no FizzBuzz for empathy. You have to ask them standard questions, shoot the breeze, interact during the various testing phases of the interview, and outside, and ultimately make a call based on the limited data you have. You will never have all of the evidence, but ultimately you have to make a go or no-go call. Sometimes you get lucky and the candidate discloses a consistent pattern from their past that is obviously disqualifying, but often you don't.

How they interact with you and your team throughout the interview is usually a good data point, and that, of course, goes both ways. Regardless of how you investigate this side of them, it's non-negotiable, and the best hires you'll make will all have a high EQ.

Team-centricity

At the same time, someone highly empathetic, but who hides in a cave and is unable to coordinate their work with the rest of the software orchestra, is not the best.

Great team players anticipate others' questions and concerns, and are proactive about reaching out and communicating both their status and the progress of their work to those around them. They make their work visible instead of requiring constant polling for the team to figure out what exactly is going on with them.

Remote environments in particular benefit from strong proactive communicators, as you can’t get away with always knowing what’s going on with everybody due to sitting in the same room for most of the week. One great Loom is worth a thousand words, but a thousand words is still much better than having to pull status out of people.

Early stage startups move at a pace that generates tremendous entropy, and someone who’s able to coordinate with the rest of a team in a way that feels natural and effortless will allow the small teams to scale without requiring project management and complex processes meant for lower-performing contributors.

Competence

Perhaps counterintuitively, I don’t fret too much about competence. To me, that’s a given. If your interview funnel is set up well, with a great take-home project, an onsite, or even a work trial, determining if the person you’re interviewing is competent should emerge naturally. After all, you and your team are technical experts, craftspeople who can look at another craftsperson’s work and quickly judge if it is any good.

Yes, AI is getting pretty good these days and it’s becoming easy for candidates of all disciplines to hide behind prompts, but that’s only an extra reason to allow them to bring those tools into the interview itself and showcase their use rather than shamefully hide their existence. How they use the agents, the back-and-forth, the planning, the tweaking, the types of searches they do as part of the exercise are all valuable data points for determining competence, and you should allow candidates to surface them.

In-person interviews and work trials are for now non-gameable, so if you’re extra paranoid, they are a great option. Non-obvious candidates will be in the hiring race with fewer other companies, possibly none, depending on your industry. That gives both you and them more flexibility and time to try working together for a few days before making it official. With a sufficiently relevant set of tasks for the work trial, you can rest assured that they’re competent, and learn a lot more about their other traits in the process.

High-agency problem-solvers

The best start-up hires are problem-solvers, not ticket resolvers. They want to understand what the business is trying to accomplish, what challenge is standing in the way of it, and how to make that problem go away. They’re autonomous in the best of ways. They don’t sit around waiting for someone to tell them what to do; they proactively find challenges and opportunities for improvement, iterate on the feedback, and go out of their way to help their teammates. They figure things out, and they do it without needing their hands held.

Someone like David didn’t take no for an answer; he assumed there had to be a narrow path through the rejections he kept getting from me, that it was only a matter of his resourcefulness before he found a crack through which to get what he wanted.

The communication, coordination and cohesion overheads that emerge with the addition of more and more staff to the roster are probably the toughest parts of scaling any business. A leader’s job is to streamline and remove these emergent dependencies between people and teams as much as possible.

Besides keeping the number of hires low, the other powerful lever that a start-up CTO has is to hire people who will be self-sufficient and require little to no support to get their jobs done. They will ask for help when necessary, but they will take pride in figuring things out on their own, checking in with you when the time is right. They will require little supervision overall, and only direction when it comes to their work. If they’re falling behind, being great communicators, they will let the rest of the team know. If they’re early, they’ll gladly pick the next interesting opportunity to be useful.

If they’re struggling, their pride in their work will make them double-down and overcome the challenge at all costs. In that case your job is to help them pump the brakes when necessary, or their pace can become unsustainable.

Cross-disciplinary empathy

A great startup hire doesn’t only think about their respective lane, but cares about the other disciplines around them that their work is impacting. They understand their angle, their priorities, and their expectations. For example:

- A back-end engineer anticipates the needs of the front-end and of the infrastructure developers as they make their changes rather than waiting for their work to be pushed back once it doesn’t meet the standard.
- A gameplay engineer doesn’t wait for the game designer to tell them that the feel of what they implemented won’t fly with the player. They put themselves in the shoes of the other discipline and think like them.
- A front-end engineer doesn’t just roll out the interface they’re working on, they think through the UX and the UI and the usability and the product management side.

They think about where and how their work will land and anticipate those objections, addressing them in advance. They proactively reach out to those disciplines to avoid yet another coordination chokepoint that will only gobble up time. As more and more skills are compacting into one single individual, we see AI engineers, product engineers, and product managers becoming more technical and moving downstream into programming. This sort of cross-disciplinary mindset buys companies a ton of leverage and ability to iterate faster.

Of course, this becomes progressively easier with experience and seniority. But that’s also why startups moving fast benefit from more mature contributors as time goes on. The reduced communication costs multiplied across more and more team members really add up over time.

Unless you have very specific technical needs, having a generalist bias in the early days is a great idea. In fact we're seeing more and more disciplines get compressed into one, with software developers covering UI, UX, PM and engineering, effectively becoming a cross-disciplinary team of one. Even in later stages, half-a-pizza teams powered by AI can move mountains on timelines that felt impossible just a few years ago.

Tolerance of uncertainty

Early-stage startups are notoriously chaotic and unpredictable. You might be working on one feature one day, it might get cancelled the next day, and maybe now you're fielding customer support calls for something you shipped that accidentally blew up. The next day you're taking a trip to a customer site to chat with potential users.

At no point do you know if any of this will make you money, how long until you kill that feature, and how many more of these iterations you'll do before you either strike gold or the company runs out of money. There are no guarantees in the startup world, except for the fact that your runway will eventually reach zero if you don't find something worth selling.

As a CTO, it is your responsibility to shield your team from the messy everyday financial reality of where your company is headed. At the same time, I prefer to keep as much as possible in the open so that the team knows how much longer the music will play and when it's time to start refreshing their LinkedIn profiles. I never want anything to be a surprise to the team when I could have been candid about it far in advance. It's a fine balance between hiding the daily volatility—mostly of the founders' moods and their confidence in the company making it—and exposing the long-term trends.

This type of universe requires people who are okay with fewer guarantees.

A startup will never have the time to run comprehensive studies, to build extensive plans, to gain all of the information necessary to make the right decision. Seeking perfection at the cost of an action is generally unacceptable, and you learn much more by failing in a valiant attempt than in delaying a perfect attempt that you might have validated months earlier by just doing the thing.

Great hires respect that process and are willing to operate without all of the information, accepting that the team will figure it out as it moves forward, often deviating from the original objective. Deleting existing artifacts or completely pivoting to a new strategy on a dime is the ultimate startup superpower.

Anybody who needs well-spelled-out plans and continuity in whatever they're doing will ultimately struggle in that kind of environment.

Wrap-up

Hiring is a gamble, but you can tilt the odds in your favor if you stop playing the same game as everyone else. The "best" hire isn't the person with the most GitHub stars or the flashiest resume; it's the person who makes your team better by existing within it. It’s the David who breaks down doors and ends up shaping the direction of the company through their relentless contributions.

Developing an eye for non-obvious talent is the only way to build a high-leverage team on a startup budget. Yes, it takes more work. It requires you to actually pay attention. It’s one of the hardest feats to pull off in startup team building. It means you must trust your own judgment over a candidate’s credentials. But if you do it right, you don’t just get a “nice” team, you get a team capable of doing the impossible. Now stop reading and go find “the best”.

28 Dec, 2025 | #cto #startups #leadership #management #hiring #recruiting #talent-acquisition #company-building

---