Dev Environments

Sneaking into an Uber parking lot to get your Development Environment up and running

Nader Khalil

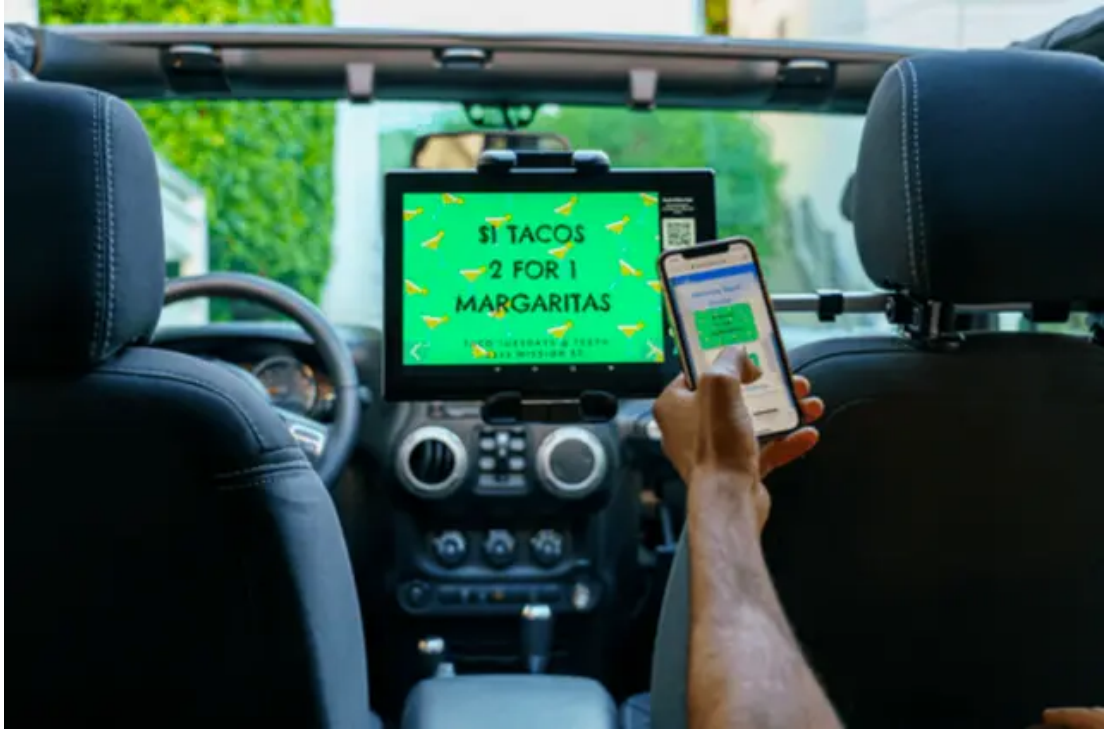
May 11, 2022 · 5 read

Paneau (YC W20) to Brev.dev: Why Configuring development environments were a painful challenge.

We got into YCombinator for Paneau: an advertising platform for local businesses.

We spoke with bar owners who complained digital ads felt like snake oil: they had thousands of clicks yet empty bars. We wanted to build a digital ad that actually brought people into the business.

So we put tablets in Ubers/Lyfts and let local businesses advertise only when the car was nearby. The tablet had QR codes linked to your Uber/Lyft app to automatically re-route your ride. Say you were going out with friends for drinks, once your Uber was within walking distance of Teeth in the Mission neighborhood, you would see an ad for “Buy 1 Get 1 Free Margaritas”. If you re-routed your Uber/Lyft, the business owner knows his ad worked, you got a free drink, and the driver got a tip. Everyone won!



Launching a startup is hard, and launching one with physical operations is even harder. There's just so much surface area for things to break, and unlike scaling a server, things cost money before you have users. We onboarded all the drivers from my living room, so they knew my address and sometimes knocked on our door at 3 am because the tablet didn't turn on. At the time, it was exhilarating to have something working, but I do enjoy sleeping at night now.

Finding drivers was the trickiest part of the business until we discovered the most valuable ride for drivers was the airport! Drivers would literally wait outside the airport for the 3-5x fare. As a result, Uber and Lyft both made parking lots about a half-mile from SFO for drivers to wait, FIFO, for the valuable rides. This was a jackpot. I parked outside of the lot and tried to walk in, only to be stopped immediately by security. I find myself back in the car frustrated, like a kid in a candy store without an allowance. I can see 300+ drivers. Meanwhile, we only have 10 signed up. I went to a nearby gas station and bought my first pack of cigarettes. I'm Lebanese, and given the stereotype of Arab drivers, I thought if I lit up a cigarette and walked onto the lot, security wouldn't bat an eye. And...it worked!! We had 300 drivers on the waitlist that night (+30x!!!). When we really started to scale during YC, their security caught on.

Forced to adapt once again, we ditched the cigarette-stroll-in approach. It was likely bizarre being the only person walking onto a parking lot, so we gave the much easier approach of driving directly into the parking lot a try, and ta-dah, we were back in action. Now able to bring a trunk full of tablets with us to the driver lot, we began

onboarding drivers on the spot, which was exponentially more efficient for drivers and us.

(Extremely supportive girlfriend getting tablets ready for me in the living room while I'd onboard drivers in the street ❤)

(Happy Driver)

(Another Happy Driver :))

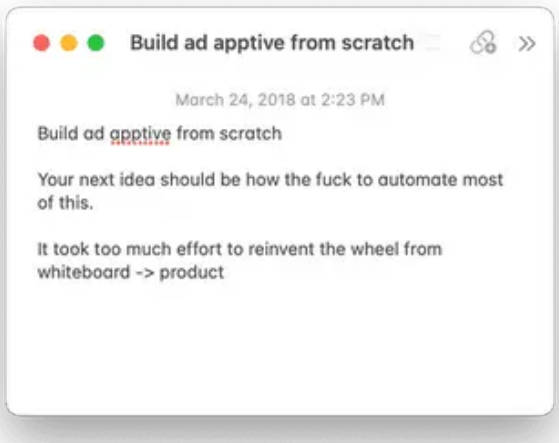
Hardware costs can be lethal. With software, our AWS bill goes up when we get users. With hardware, you have heavy costs regardless. Initially, we launched by having drivers use their hotspot for data which was miserable. If they took a smoke break or filled up gas, the hotspot would disconnect, we wouldn't get their driving metrics, they wouldn't get paid, and they'd churn. We needed SIM cards, but at this point, we hadn't gotten into YC yet and were bootstrapping. We called AT&T and got a quote. We called Verizon to beat it, then called T-Mobile to beat Verizon's, then called Sprint to beat Verizon's. Finally, we circled back to AT&T and they offered free devices if we paid $10/mo for data. PHEW.

Financials were looking fine, and we had $25K in the bank (so we thought). I wake up on a Saturday in June 2019 to a suspicious email, check our bank balance, and lo and behold we're at -$200. In a total crapshoot, I called Fidelity and told the service rep to drain my 401K so we can keep the business going. The service rep mentioned this was a lousy retirement decision. Retirement?? I couldn't see further than three weeks in the future. This was a great startup lesson: people don't pay invoices on time. We went negative because we had $20K in unpaid invoices. My 401K kept us going, we chased after the unpaid invoices, then GOT INTO YC. That $150K coming into the bank was such a sigh of relief.

We always found a solution for physical things, and I wouldn't even say it slowed us down. Whether it was using a stereotype to my advantage, purging my 401k to buy two more months of runway, or hot-swapping tablets for drivers on their trips, we always figured it out. The only time I felt truly defeated was when something with our development got messed up. I'm not originally a business guy! I was enamored by development at 14, to the point that the local university had me recruiting new freshmen into Computer Engineering, the field I would eventually graduate with a major in 8 years later.

(14 yo Nader getting new freshmen excited about robotics)

When we first had to deploy Paneau in March 2018, it took the entire month. This was my first experience deploying, and I was so frustrated, because "it worked locally". I could launch ads onto my tablet when running on localhost, why could I not send my folder to AWS, and have it just work? Frustrated, I wrote this in my notes:



Even after that, it was never a “one and done”. Experts reading this are probably laughing at my naivete. When we were scaling, we spent an entire month finding a bug making the advertiser’s app infuriatingly slow. We couldn’t replicate it locally, so we had to push changes blindly, sit back, and hope. This, too, seemed like a solved issue! I worked at Workday prior to Paneau, building a cloud-based dev environment for the internal app developers. It standardized the dev environments so they’re just like prod environments which dramatically reduced the amount of time developers spend solving self-inflicted bugs. Similar to Replit and Codespaces, this solution had developers code in a web app or get locked into a certain set of tools.

So Paneau was doing extremely well, growing advertisers and drivers by some 40% monthly, until the pandemic sent it crashing down. It was March 2020. We were the last physical YC batch. Our YC partners had a call with us where they said “Demo Day is no longer in 2 weeks. It’s in 6 days. There’s no pitching, we’re just sending your decks to investors. You’ll never raise again the world is ending. Grab as much money as you can.” We went from a fleet of 400 to 7 that week. We didn’t raise a dime.

The world really felt like it was ending, or as some put it at the time, “nature is healing”. We called our tablet distributors and data plans to pause everything since we didn’t bill any customers for March, and our $240K ARR turned to $0 overnight. Unfortunately, the service reps had been furloughed because the world was ending, so nothing could be paused. My co-founder and I left our SF apartment and moved into my parent’s garage in San Diego (thanks mom + extremely patient significant others) and worked on pivots until we realized the company was going to bleed dry. (Side note: want to carry on the baton? Get in touch. I’ll give you our code and tell you what to avoid — but heads up, this journey isn’t for the faint of heart!)

We sit in this garage and we thought. The world was in a dire place. Even the beaches in San Diego had locked down (if you're not from SD, this was a huge deal). Life's too short to build just anything, so if we could build whatever we wanted, what would it be? We kept coming back to our pain points in the years prior, and what we would have given to not get stuck solving machine problems having nothing to do with the domain/product of what we were building.

A lot of venture dollars and founder energy are focused on tools that make deploying code easier. There wasn't nearly as much attention given to how painful developer environments could be, with most advice being to learn Docker and forcing a tool meant for deploying production code to also be used for developer environments. But this is a poor developer experience. We wanted something lightweight that allowed you to forget anything other than the programming task at hand - where a naive approach to development is anything but. Remember when we thought that 'development mostly involves sitting in front of a text editor and compiler programming', rather than endless dashboards, tooling, and distractions masquerading as abstractions. When we didn't need to know everything to do anything.

Brev.dev provides local-first developer environments so you can keep using the tools you love on your machine, instantly onboard new engineers, and rid yourself of installation README's and painful software updates forever.

We should know - we exclusively use Brev for our own development, and we think it's pretty great - we would love for you to check it out.

Previous

← Harness Multi-stage builds to create optimal images

Next

Free your mac from docker →