

Production Setup: Explained Simply

n8n Performance Architecture — Burger Shop Analogy (200 VUs / 1 Hour Example)

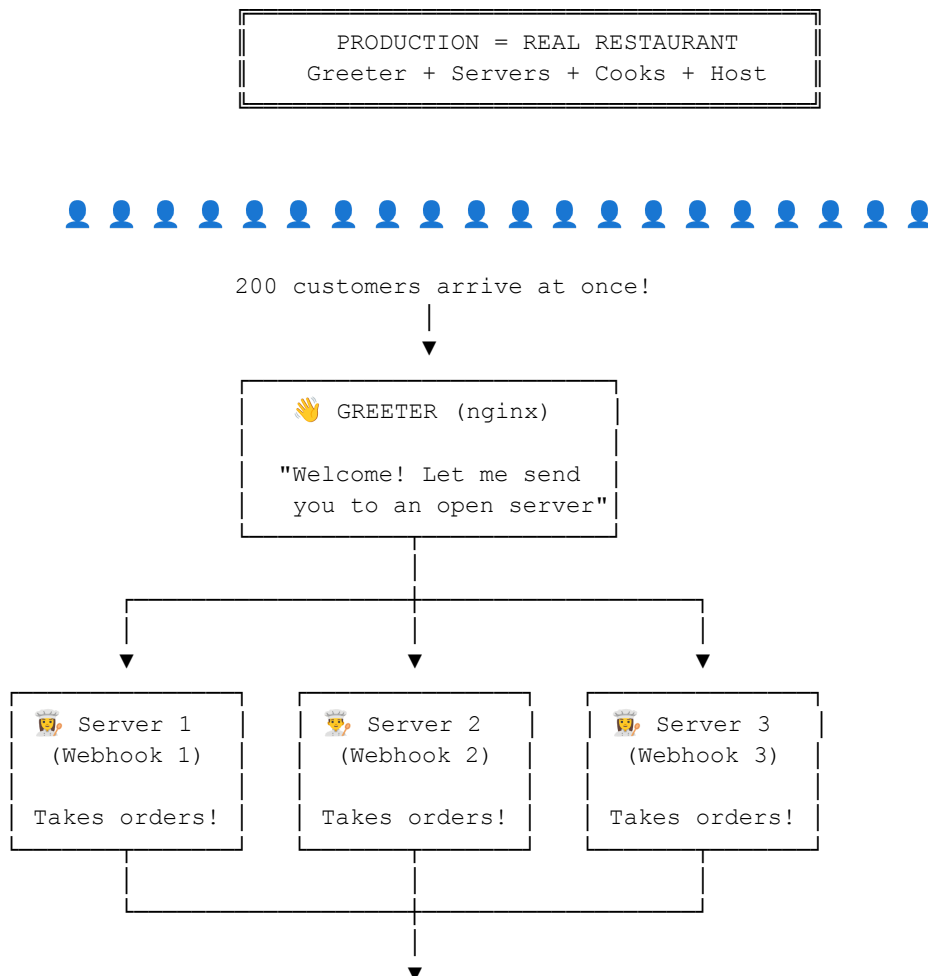
1) Big Idea (In One Sentence)

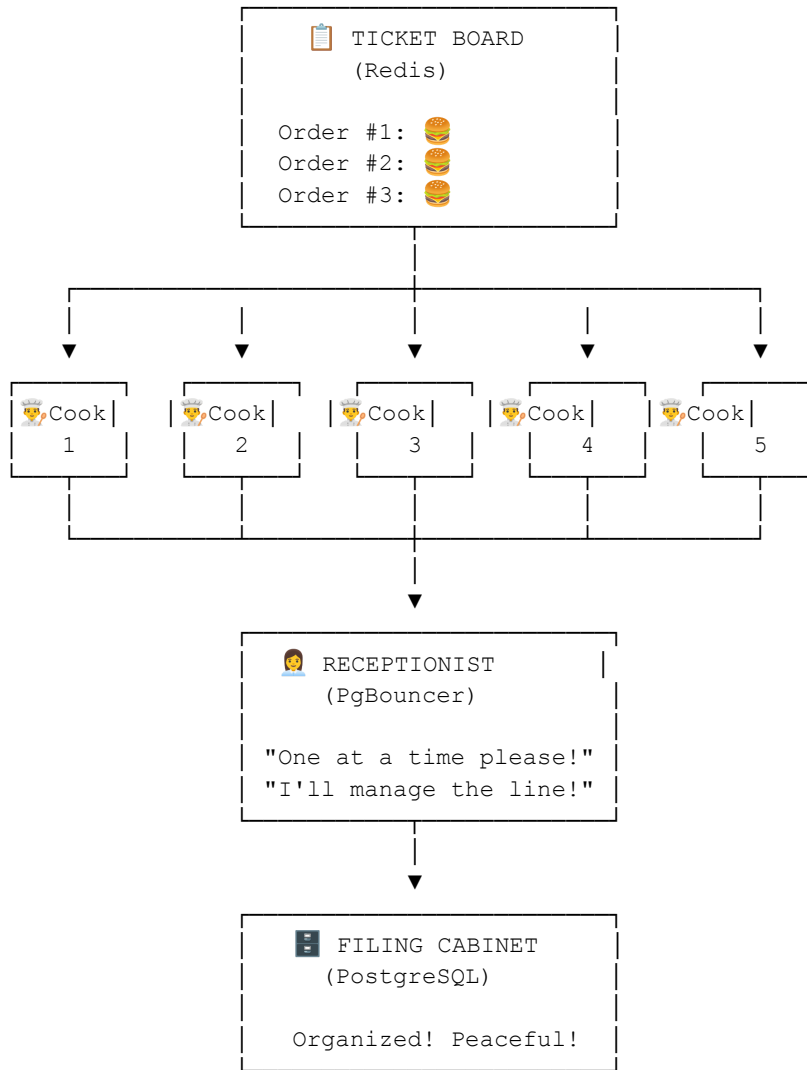
Production is like a real restaurant: a greeter (nginx) spreads customers across multiple servers, a ticket board (Redis) organizes work, cooks (workers) handle heavy execution, and a receptionist (PgBouncer) prevents database connection chaos—so everything stays calm under load.

2) Architecture Diagram (ASCII)

🔴 Production Setup: Explained Simply

🍔 Imagine a REAL Professional Restaurant!





3) Everyone Has One Job (Why This Scales)

🎯 Everyone Has ONE Job!

THE TEAM:

👋 GREETER (nginx)
"I welcome customers and send them to open servers"

👨🍳👨🍳👨🍳 SERVERS x3 (Webhook Processors)
"We ONLY take orders. That's it!"

📁 TICKET BOARD (Redis)
"I keep all orders organized"

👨🍳👨🍳👨🍳👨🍳👨🍳 COOKS x5 (Workers)

"We ONLY cook burgers. That's it!"



RECEPTIONIST (PgBouncer)

"I manage ALL traffic to the filing cabinet"



FILING CABINET (PostgreSQL)

"I store everything safely"

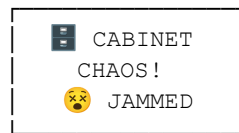
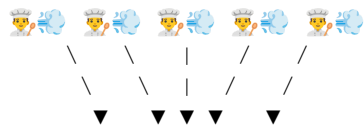
EVERYONE DOES ONE THING REALLY WELL!

4) Why PgBouncer Matters (No More "Database not ready")



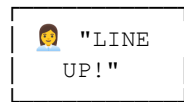
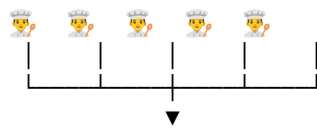
The Magic of the Receptionist (PgBouncer)

WITHOUT RECEPTIONIST (Queue+PG):



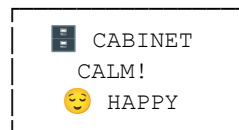
← Everyone crashes in!
"Database not ready!"

WITH RECEPTIONIST (Production):



← Receptionist manages traffic

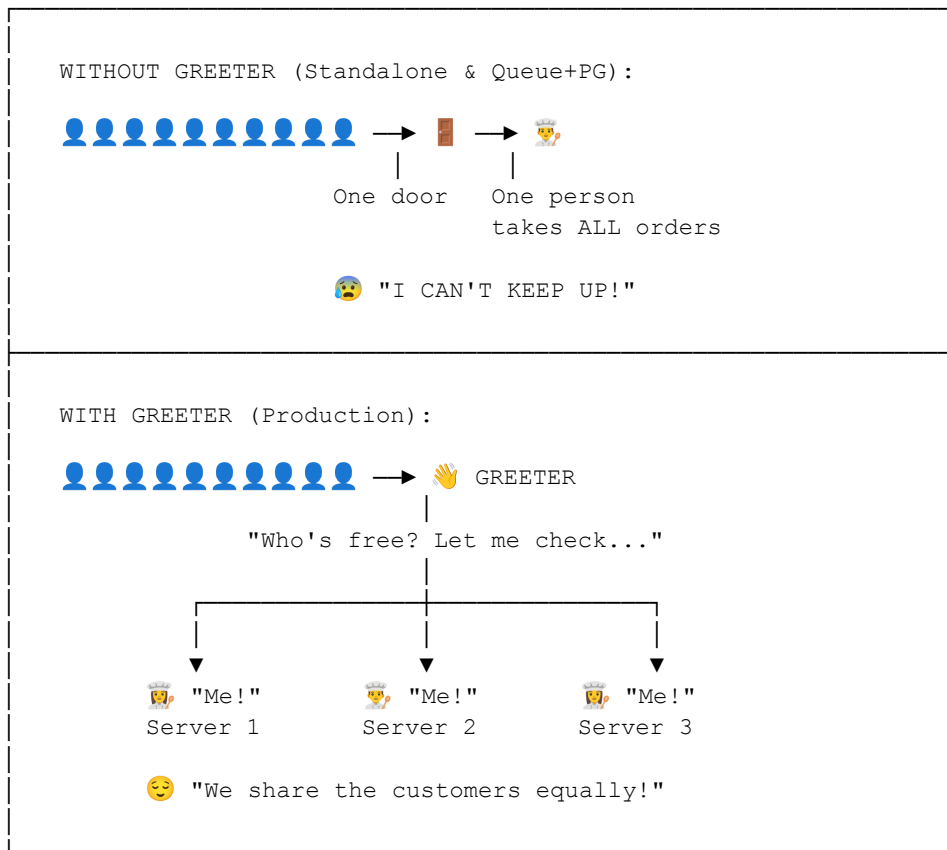
One at a time



← Organized access!
No crashes!

5) Why nginx Matters (No Single "One Door" Bottleneck)

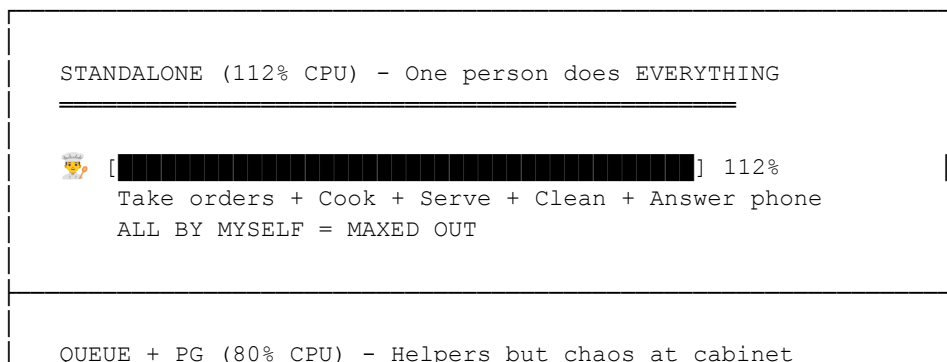
👋 The Magic of the Greeter (nginx)



6) Performance Insights (Why CPU Drops to ~6%)

In Production, work stops piling up in one place. Requests are distributed across multiple webhook processors, heavy workflow execution runs in separate workers, and PgBouncer prevents connection storms to PostgreSQL. That combination cuts wasted CPU dramatically.

📊 Why CPU is Only 6%!



8) What Happens at 200 Customers for 1 Hour (Result Summary)

🔥 What Happens at 200 Customers for 1 HOUR

STANDALONE (After 2 minutes)

👤😵 "I QUIT!" (crashed)
❌ 65% of customers left angry
✅ 35% got served

QUEUE + PG (After 2 minutes)

🗄️😵 "Cabinet jammed!" (database overwhelmed)
❌ 33% of customers left angry
✅ 67% got served

PRODUCTION (After 1 HOUR!)

✅ 720,448 customers served
✅ 100% success rate
❌ 0 angry customers
Still running strong after 60 MINUTES!

9) Final Comparison Table (All 3 Setups)

📊 The Final Comparison

🍔 BURGER SHOP COMPARISON 🍔

	STANDALONE	QUEUE+PG	PRODUCTION
CPU	🔴 112%	🟡 80%	🟢 6%
Memory	🔴 1.17 GB	🟡 980 MB	🟢 251 MB
Success	🔴 35%	🟡 67%	🟢 100%
Duration	🔴 2 min	🔴 2 min	🟢 1 HOUR+

10) Bottom Line (One-Liner)

Production wins because it removes bottlenecks and removes waste: nginx balances incoming load, workers execute separately, and PgBouncer prevents database connection storms—so performance stays stable even under sustained traffic.