Building a Full SQL Server Telemetry & Alerting Pipeline From the Ground Up

It has been a challenging and incredibly rewarding journey going from helpdesk to sysadmin and now into DBA work in a data-heavy environment. Before we had commercial tools, when I began learning the ropes as a Jr. DBA we had at the time 60+ SQL Servers where we relied primarily on internally built email alerts — which worked well for years before I joined.

As the environment grew, so did the volume and complexity of operational signals. The email alerting functioned reliably, but the sheer scale of new activity meant that issues were increasingly difficult to triage quickly. Important alerts were scattered across dozens of emails or required manually running ad-hoc queries. New emails weren't needed — centralized visibility was.

This became the foundation for a centralized telemetry, monitoring, and alerting project that helped surface key SQL Server metrics in a more visual and actionable way.

The Solution: SQL Server Big Brother

I built a centralized telemetry + monitoring system ("SQL Big Brother") powered by ~30 custom PowerShell and Python scripts. These scripts collect metrics across our DEV/TEST/PROD SQL Servers every five minutes and write them into a centralized SQL monitoring database.

That data powers:

1. A unified Power BI dashboard

- Hosted on our on-prem Power BI report server
- Auto-refreshes every 5 minutes
- Provides an "at-a-glance" operational snapshot across the entire SQL estate
- Includes 15+ granular drill-down pages (blocking, TempDB, failed jobs, backups, login anomalies, disk space, business specific monitoring, etc.) Complete with sliders and other drill down tools.

2. Automated alert ticketing

A companion set of scripts leverages the Freshworks API to automatically generate tickets when conditions are met — such as blocking thresholds, job failures, disk pressure, missing backups, and more.

This has closed a long-standing gap in documenting operational incidents and ensures DBAs can respond quickly with clear audit trails.

A Real Example of Preventing Production Issues

One notable case occurred earlier this year:
On a major production SQL Server, hidden shadow copy files had begun consuming disk space. These weren't visible through normal workflows, and no traditional SQL alerting surfaced them.

But on our **disk space heat map**, which shows disk utilization trends across the estate, one of the production drives began drifting from "green" into "yellow." That visual deviation was subtle — but it immediately stood out.

When we investigated, we discovered that a component of our Commvault backup infrastructure was unexpectedly generating large shadow copies. Had it continued, the production system would have run out of disk space — causing a critical outage.

The centralized dashboard provided **early detection**, long before the issue became severe. That single catch prevented:

- A major production impact
- Potential data loss
- Emergency after-hours remediation
- Service disruption to our business

This was one of several early warning wins that reinforced the helpfulness of the project.

What the System Monitors

The dashboard and automated alerts allow the DBAs to monitor:

- SQL Agent Job Failures
- Down Servers
- Last Server Restart
- Noteworthy Login Failures
- Lower Environment Refresh Status
- Heavy Sessions (Blocking, Waits, Memory, etc.)
- Azure Backup Status
- Logshipping Health
- Missing/Failed SQL Backups
- Noteworthy Database States
- Terminated Users With SQL Logins
- SQL Server Encryption Status
- CPU Usage/History
- TempDB Condition
- Disk Space Monitoring & Low Disk Alerting
- Memory Dump Ticket Generation

Outcome

This project transformed our team's ability to observe, respond to, and understand the health of our SQL Server environment. Instead of relying solely on scattered emails or manual queries, the team now has:

- **Centralized telemetry**
- **Visual situational awareness**
- **Automated issue detection**
- **Consistent ticket documentation**
- **Faster triage and response cycles**
- **Actionable insights that helped prevent real incidents**

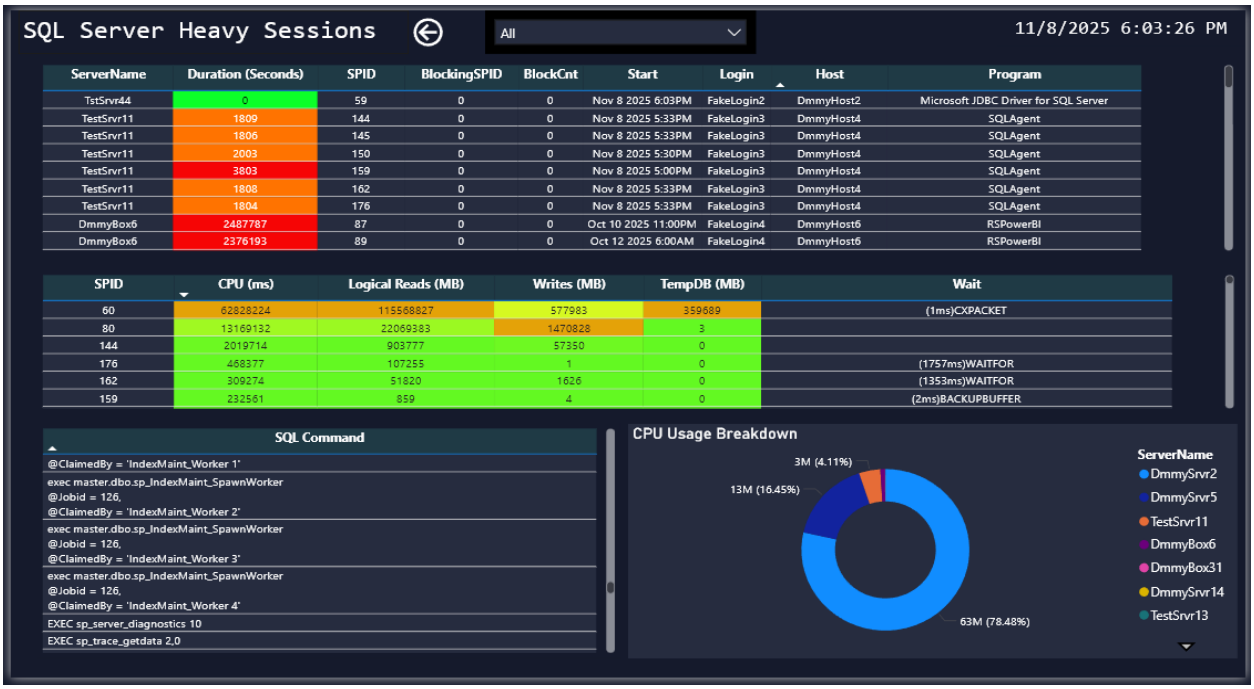
What began as a personal learning initiative quickly became a full monitoring ecosystem for our SQL Server environment — one that improved visibility, reduced triage time, prevented outages, and elevated the DBA team's operational awareness.

This project challenged me to deepen my scripting, automation, data engineering, and monitoring skills, and it remains one of the most meaningful engineering efforts I've contributed to.

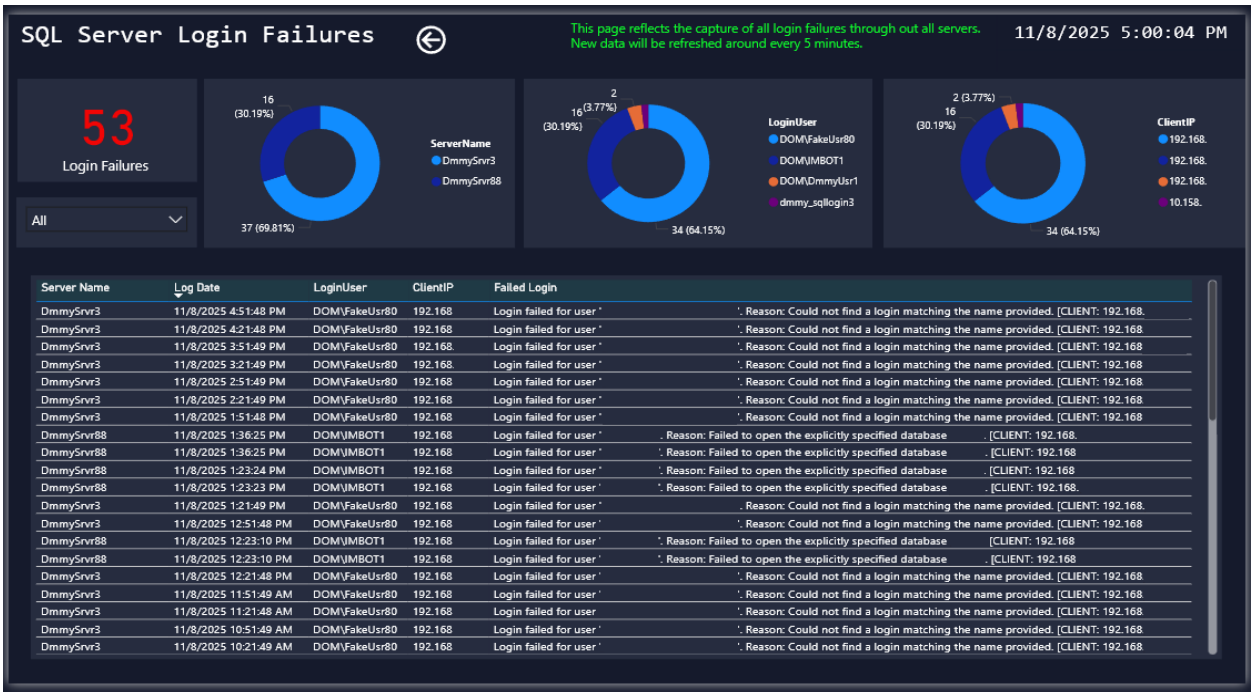
Screenshots:

Main page. Cards above utilize conditional formatting as tuned for our needs. All clickable cards

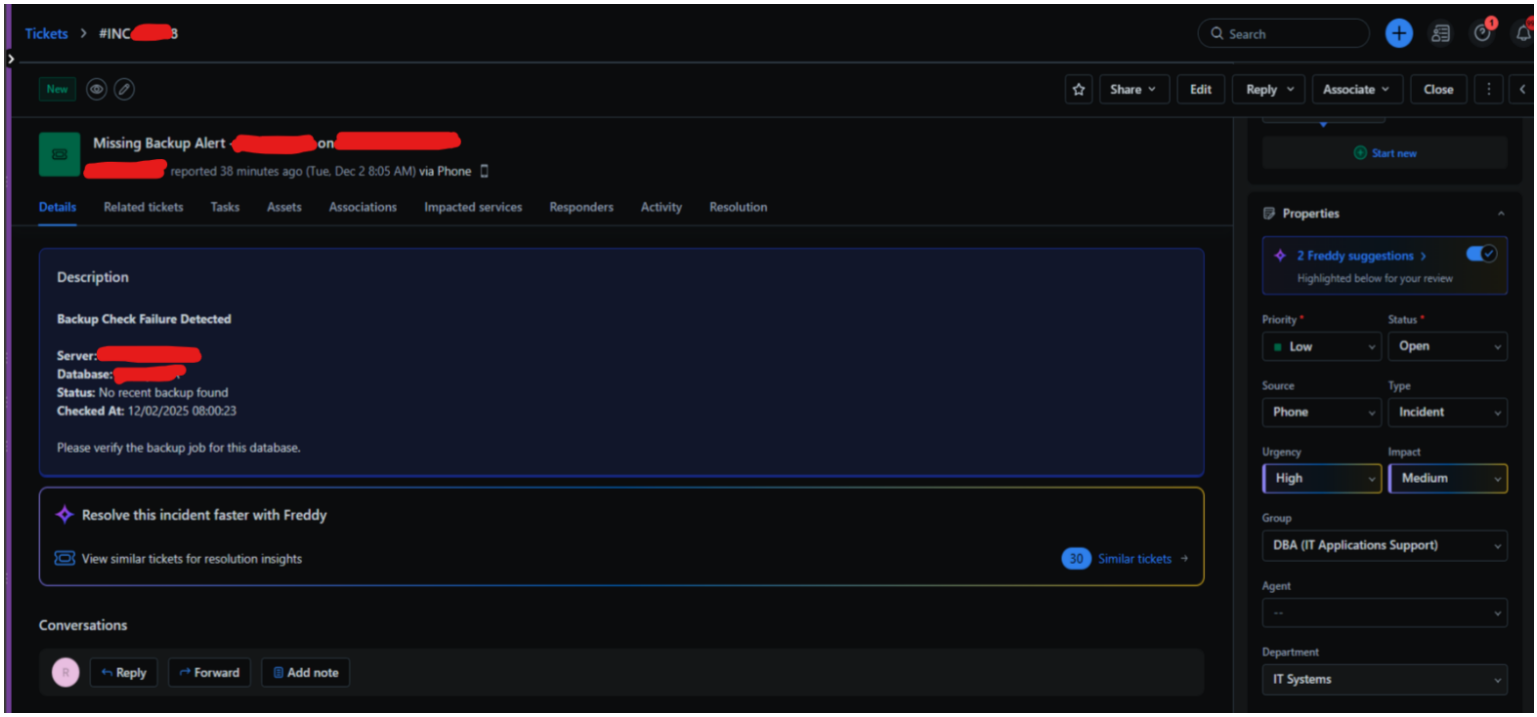
Heavy Sessions page allow for a quick look at all noteworthy sessions in our whole environment, allows for easy drill-down and filtering.

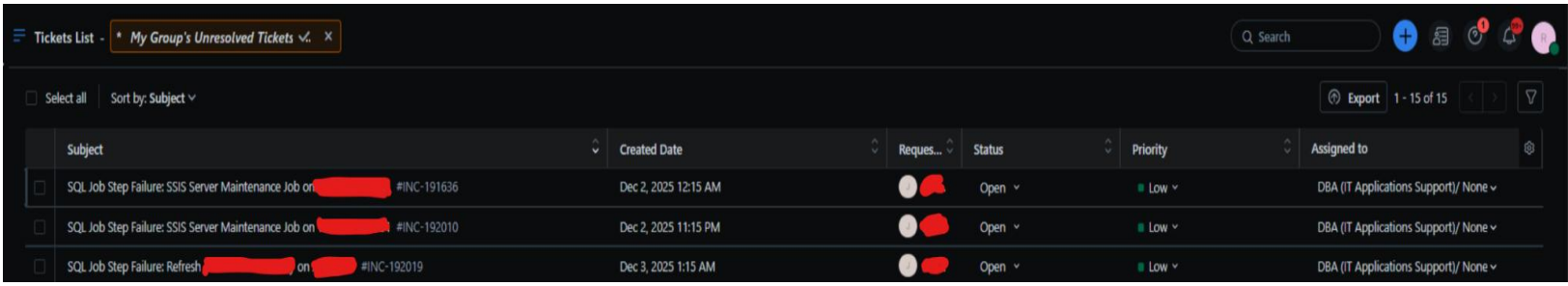


Login Failures page offers the team quick insight into all failed attempts at connecting. We can see who, where, and when they occurred along with visual aids to make finding possible trends and issues easier.



Example of an automated alert ticket utilizing the data gathered from this project. Tickets in this example are for a missing backup and for SQL Agent job failures.





Quick overhead visual of how the whole thing works.