High-Fidelity Operations

Production-Grade AWS Monitoring Stack

Simulating L2 Operations, Incident Management, and Infrastructure Observability

> A live production-like platform designed to move beyond theory and demonstrate full-stack operational control.

Moving From Theory to Live Production Operations

The Challenge

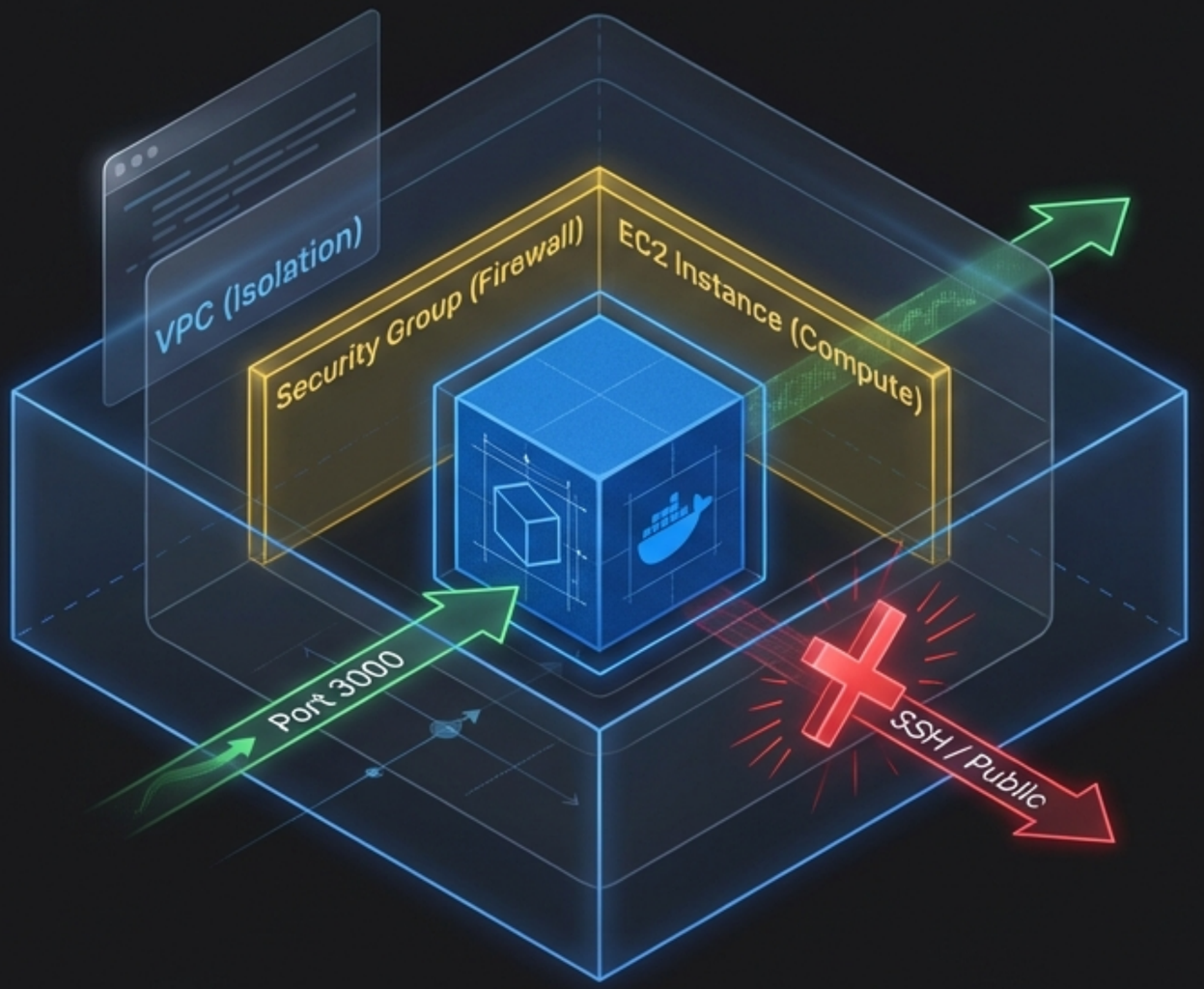
- Theoretical Knowledge
- Isolated Concepts
- Passive Learning

The Goal

- Real-time Health Monitoring
- P1 Incident Simulation
- ITIL-based Alerting

```
// This isn't a lab; it's a simulation of the enterprise environment I support daily.
```

Layer 1: The AWS Infrastructure Foundation



EC2 (Compute)

Used as the base compute layer to host the stack; industry standard for scalable environments.

VPC (Isolation)

Network segmentation isolating the monitoring server from public access.

Security Groups (Access Control)

Firewall rules enabling Grafana access while blocking unauthorized entry.

The Why: We treated this like a compliant production node: strict isolation, minimal attack surface.

Layer 2: The OS Battlefield

```
[root@production-node ~]# uname -a
Linux production-node 5.4.0-1045-aws #47-Ubuntu SMP Tue Apr 13 17:02:49 UTC
2023 x86_64

[root@production-node ~]# free -m
               total        used        free      shared  buff/cache   available
Mem:            3936         842        1254           2        1839        2814
[root@production-node ~]# df -h | grep /dev/root
/dev/root        29G         14G         15G         48% /
[root@production-node ~]# top -b -n 1 | head -n 5
%Cpu(s):  2.4 us,  1.2 sy,  0.0 ni, 96.2 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
```

Insight: Linux is the primary OS in 95% of enterprises. All monitoring tools integrate natively here. This is where we validate the 'truth' of the system.

Layer 3: The Platform & Containerization

Configuration: Docker & Docker Compose

Deployment: Single command spin-up

```
> docker-compose up -d
```

Strategic Rationale

- No Dependency Hell: The stack runs identically on any server.
- Industry Standard: Mirrors how fintechs and SaaS platforms deploy microservices.

The Monitoring Engine: Prometheus & Exporters

Node Exporter

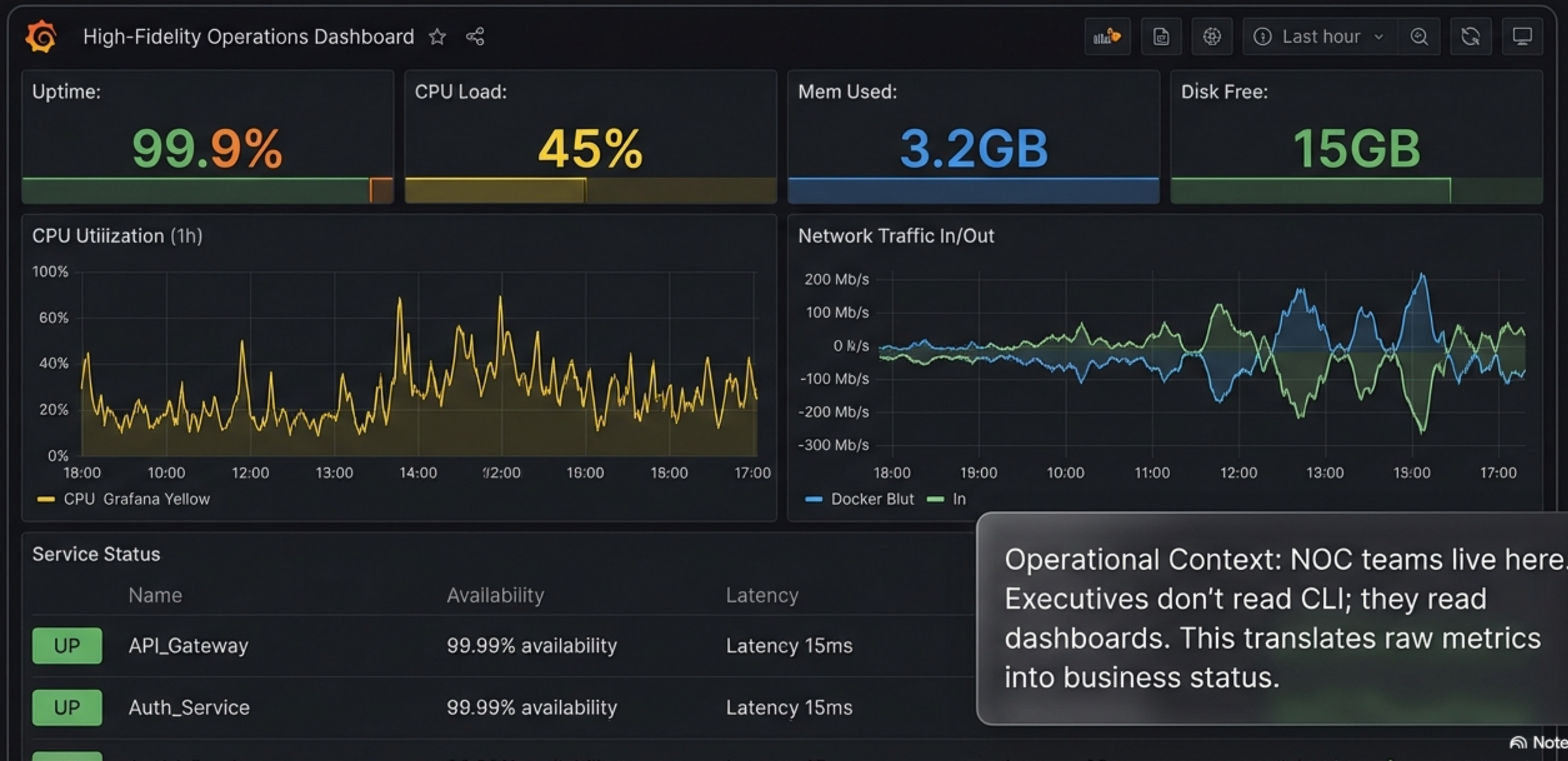
OS-Level Truth:
CPU, Memory, Disk

Blackbox Exporter

Service Health:
HTTP, API, DNS

Prometheus serves as the backend logic, similar to the internal workings of Datadog or Dynatrace, triggering alerts based on defined thresholds.

Visualization: The War Room Dashboard



Operational Context: NOC teams live here. Executives don't read CLI; they read dashboards. This translates raw metrics into business status.

Mapping Simulation to Enterprise Reality

Project Tool (Open Source)		Enterprise Equivalent
Prometheus	→	Datadog / Dynatrace (Metrics Backend)
Grafana	→	Splunk / Tableau (Monitoring UI)
Linux CLI	→	Root Cause Analysis (Universal)
ServiceNow (Simulated)	→	Incident Management

Takeaway: The specific tools change, but the topology and operational logic remain identical to real L2 support.

P1 Scenario: Managing CPU Saturation

Detection

Prometheus Alert Fires: CPU > 80%. Grafana dashboard spikes red.

Diagnosis

Transition to CLI. Commands used: 'top' and 'ps -ef --sort=-%cpu' to identify the runaway process.

Action

Resolution: Kill process, restart service, or scale EC2 instance.

A simulation of a daily L2 reality: Detecting the signal, isolating the process, and restoring service stability.

The ITIL Lifecycle Integration

We follow the full lifecycle. The job isn't done until the incident is documented and the root cause is analyzed.

Operational Expertise Over Theory

> “I designed this production-grade stack to simulate real operations. I don’t just study cloud concepts—I operate them.”

- [x] Infrastructure Metrics & Alerting
- [x] Service Health Checks
- [x] Incident Handling & RCA
- [x] 4 Years L2 Experience + Cloud Ops