

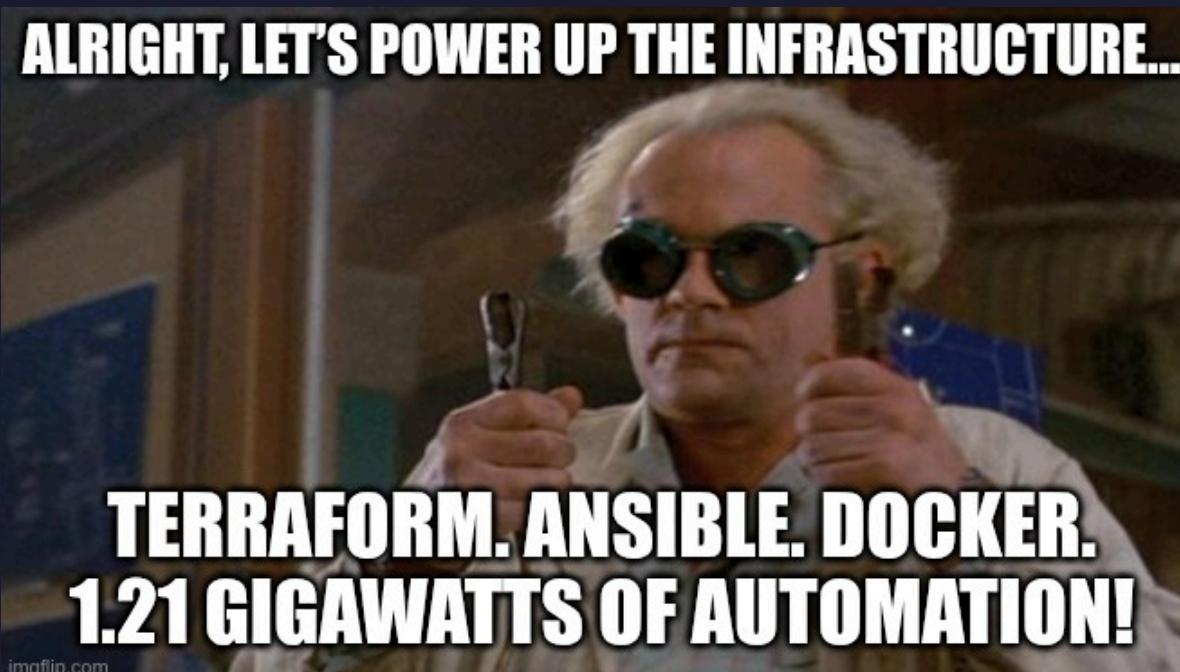
# ⚡ Multi-Stack DevOps Infrastructure

---

## Automated Cloud Deployment with IaC

Martin Kaiser

AWS | Terraform | Ansible | Docker



# Project Overview

**Objective:** Deploy polyglot microservices voting application on AWS with full automation

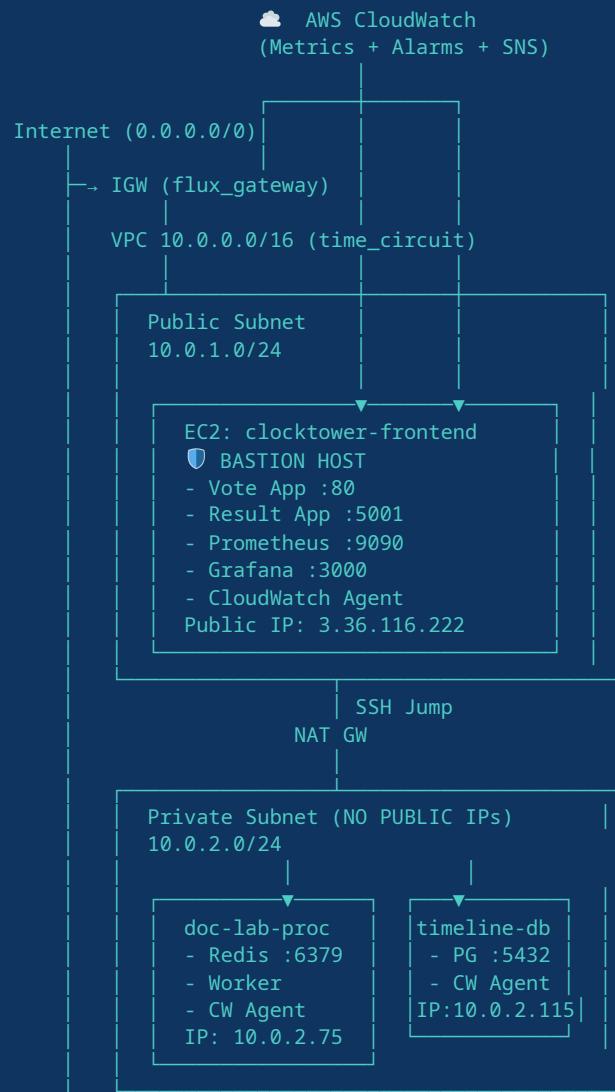
**The Stack:**

- **Vote App** (Python Flask)
- **Redis** (Message Queue)
- **Worker** (.NET Core)
- **PostgreSQL** (Database)
- **Result App** (Node.js)

**Infrastructure:** Terraform + Ansible + Docker

**Monitoring:** Prometheus/Grafana + CloudWatch

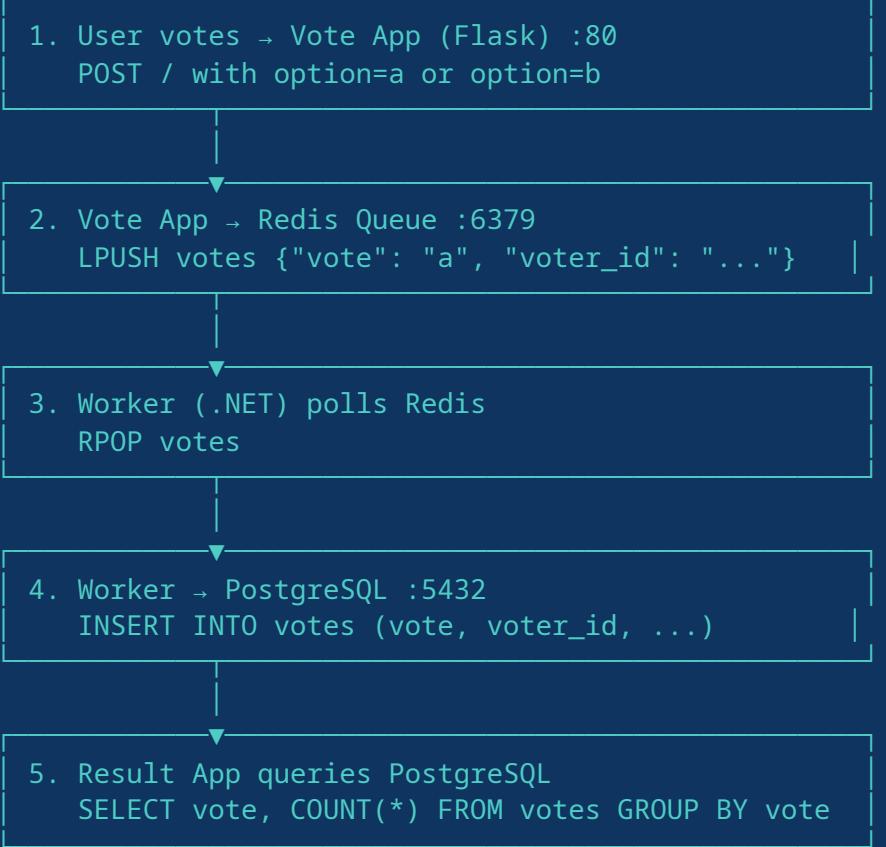
# AWS Architecture Overview



# Security Group Architecture

Security Group	Ingress Port	Source	Purpose
<b>sg_frontend</b>	22	Your IP	SSH access
	80	0.0.0.0/0	Vote App HTTP
	5001	0.0.0.0/0	Result App
	3000	0.0.0.0/0	Grafana Dashboard
	9090	0.0.0.0/0	Prometheus
	22	sg_frontend	SSH via bastion
<b>sg_backend</b>	6379	sg_frontend	Redis from Vote App
	9100	sg_frontend	Node Exporter metrics
	22	sg_frontend	SSH via bastion
<b>sg_database</b>	5432	sg_backend	PostgreSQL from Worker
	5432	sg_frontend	PostgreSQL from Result
	9100	sg_frontend	Node Exporter metrics
<b>All instances</b>	Egress 443	0.0.0.0/0	CloudWatch Agent → AWS

# Application Data Flow



# Problem #1: SSH & Private Network

**The Situation:** Can't SSH to private instances (10.0.2.x)

**What Went Wrong:**

- Security groups need **source SG references**, not CIDRs
- Terraform IPs → Ansible automation difficult
- SSH fingerprint auto-confirmation failed

**The Fix:**

```
ingress {  
  security_groups = [aws_security_group.sg_frontend.id]  # ✓  
}
```

**Key Learning:** Stay at proven practices. Don't overautomate or you will regret it.

# Problem #2: Docker SDK - 3 Hours of Hell

**The Goal:** Use Ansible `docker_container` module (Python SDK)

**The Journey:**

Phase	Issue	Result
1	Group membership not propagated	✓ Fixed
2	Docker SDK 7.x → 6.1.3	✗ Still fails
3	System urllib3 vs pip conflicts	✗ Still fails
4	urllib3 2.x dropped <code>http+docker://</code>	✗ Still fails
5	SDK fundamentally broken	✗ Total fail

**Error:** `URLSchemeUnknown: Not supported URL scheme http+docker`

# Problem #2: The Reality Check

## What Works: Docker CLI

```
docker ps  # ✓ Works perfectly
```

## What Doesn't: Docker SDK for Python

```
import docker
client = docker.from_env()  # ✗ Fails with URL scheme error
```

## The Solution - Use Docker CLI:

```
- name: Deploy containers
  ansible.builtin.command:
    cmd: docker run -d --name worker --restart always worker:latest
```

## Industry Reality:

- Netflix, Spotify, Google → Use **Docker CLI** in production

# Problem #3: Phantom Votes

**The Mystery:** Send 1000 votes → Get 987-1003 in database

**Root Causes:**

1. **Latency: Redis → Worker → PostgreSQL** migration takes time
2. **Be patient** - Worker processing slower than vote arrival
3. No end-to-end measurement

**Bottleneck:** Message queue retention, not CPU (40-80%)

# Problem #3: Measuring Throughput

## 3-Level Verification:

Level	What It Proves
1. API (HTTP 200)	Request accepted
2. Queue (Redis)	Queued
3. Database (PostgreSQL)	✓ Persisted

## Measurement:

```
psql -c "SELECT COUNT(*) FROM votes;" # Before  
./quick-stress.sh 1000 40 # Test  
psql -c "SELECT COUNT(*) FROM votes;" # After
```

**Key Learning:** HTTP 200 ≠ data in database. Measure **end-to-end**.

# Monitoring Architecture

## Why Prometheus + Grafana?

Curious about these tools - heard so often, high importance for applications

System	Purpose	Retention
Prometheus + Grafana	Real-time infrastructure metrics	15 days
CloudWatch + SNS	Production alarms, email notifications	15 months

**Infrastructure Metrics:** CPU, Memory, Network, Disk per instance

**Application Metrics:** Measured via stress tests (41.66 votes/sec)

# Technology Stack

## Infrastructure as Code:

- **Terraform** - 800+ lines, 20+ AWS resources
- **Ansible** - 11 playbooks:
  - Deployment: `deploy-vote`, `deploy-worker`, `deploy-redis`, `deploy-database`,  
`deploy-result`
  - Setup: `install-docker`, `deploy-monitoring`, `setup-cloudwatch`
  - **Testing**: `test-connectivity`, `check-logs`, `stop-all`

## Monitoring:

- Prometheus + Grafana, CloudWatch + SNS

# Key Learnings

## 1. Security & Automation:

- Security groups need **source SG references**, not CIDRs
- Automation is **difficult** - IP migration Terraform → Ansible, SSH fingerprints

## 2. Tool Selection:

- "Correct" tool (Docker SDK) ≠ "right" tool (Docker CLI)
- Going away from best practices → **total failure**

## 3. Validation & Measurement:

- Measure end-to-end throughput under load
- HTTP 200 ≠ data persisted (latency in Worker)

**The Slim Path:** Stay at proven practices. Don't overautomate or you will regret it.

# Live Demo

## 1. Vote & Result Apps

- Vote: <http://3.36.116.222>
- Result: <http://3.36.116.222:5001>

## 2. Monitoring

- Grafana: <http://3.36.116.222:3000>

## 3. Stress Test

```
./quick-stress.sh 500 20
```

## 4. Verify

```
SELECT vote, COUNT(*) FROM votes GROUP BY vote;
```

# Thank You

Martin Kaiser

AWS Solution Architecture | Infrastructure as Code

Access:

-  Vote: <http://3.36.116.222>
-  Grafana: <http://3.36.116.222:3000>
-  GitHub: <https://github.com/kaiser-data/aws-terraform-devops-infrastructure>

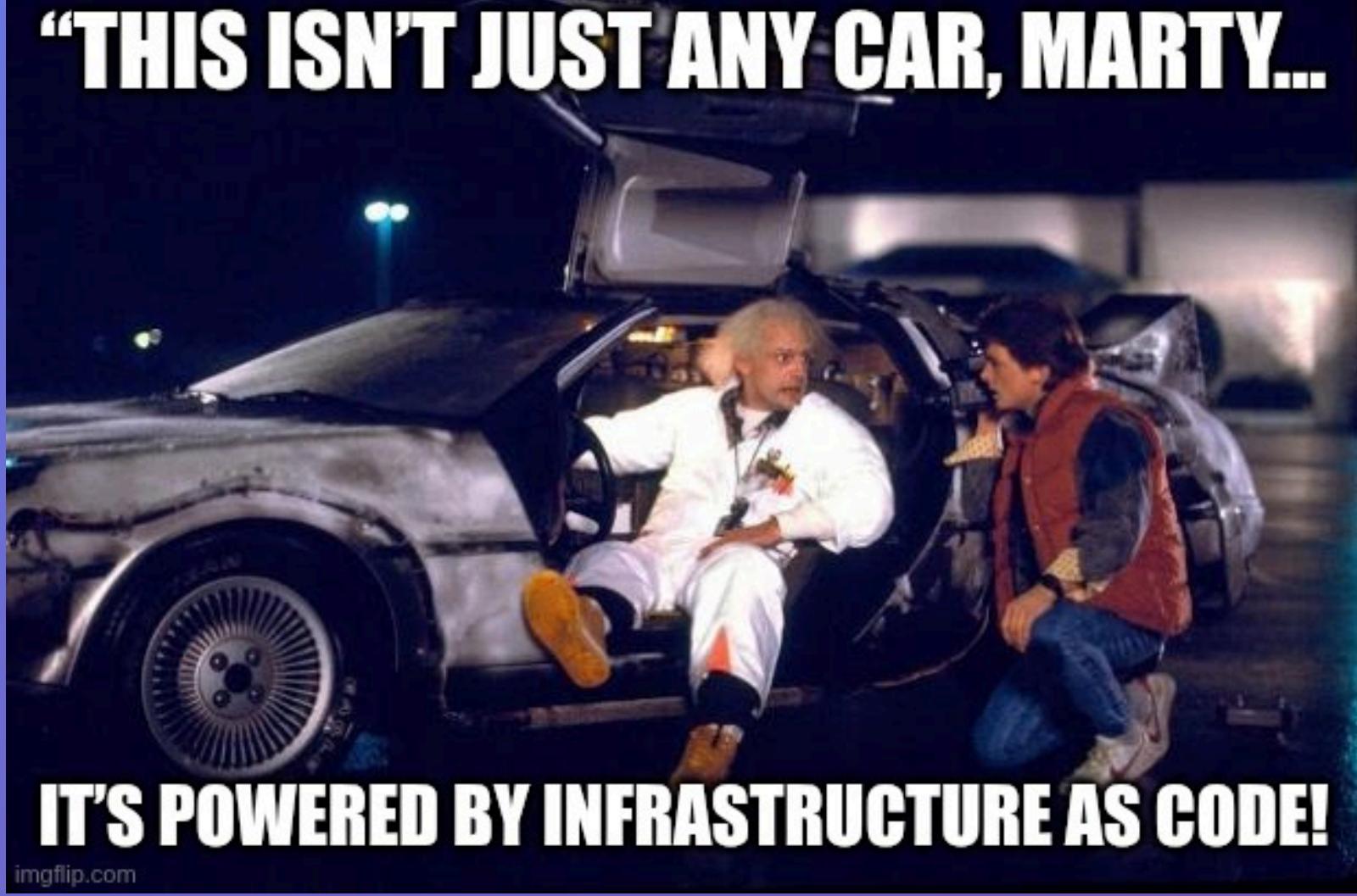
Key Takeaways:

1. Security groups and SSH need careful adjustment
2. Docker CLI reliable, Docker SDK totally failed
3. Measure throughput end-to-end to verify reliability

Built with: Terraform, Ansible, Docker, AWS

AI assistance: Primarily Claude (Anthropic)

**“THIS ISN’T JUST ANY CAR, MARTY...**



**IT'S POWERED BY INFRASTRUCTURE AS CODE!**

imgflip.com

