

# How to Improve Your Service by Roasting It

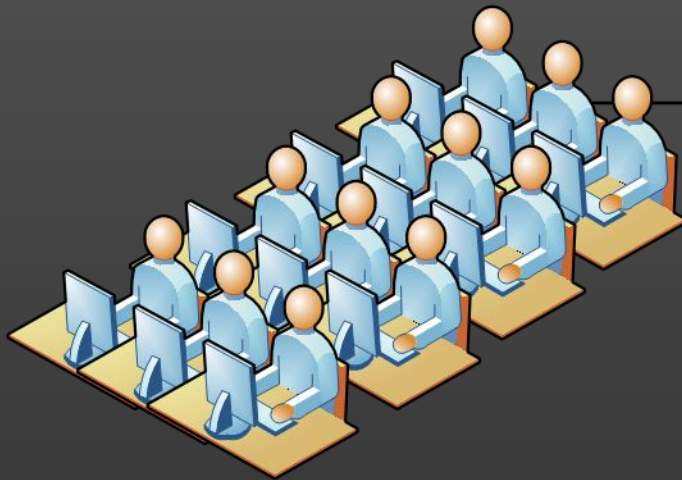
Jake Welch  
jawelch@microsoft.com  
@jaketwelch / #AzureSRE



Web Server



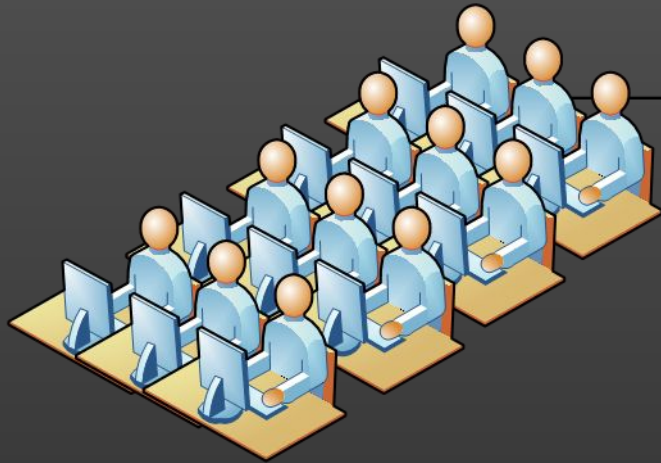
Developer



Web Server



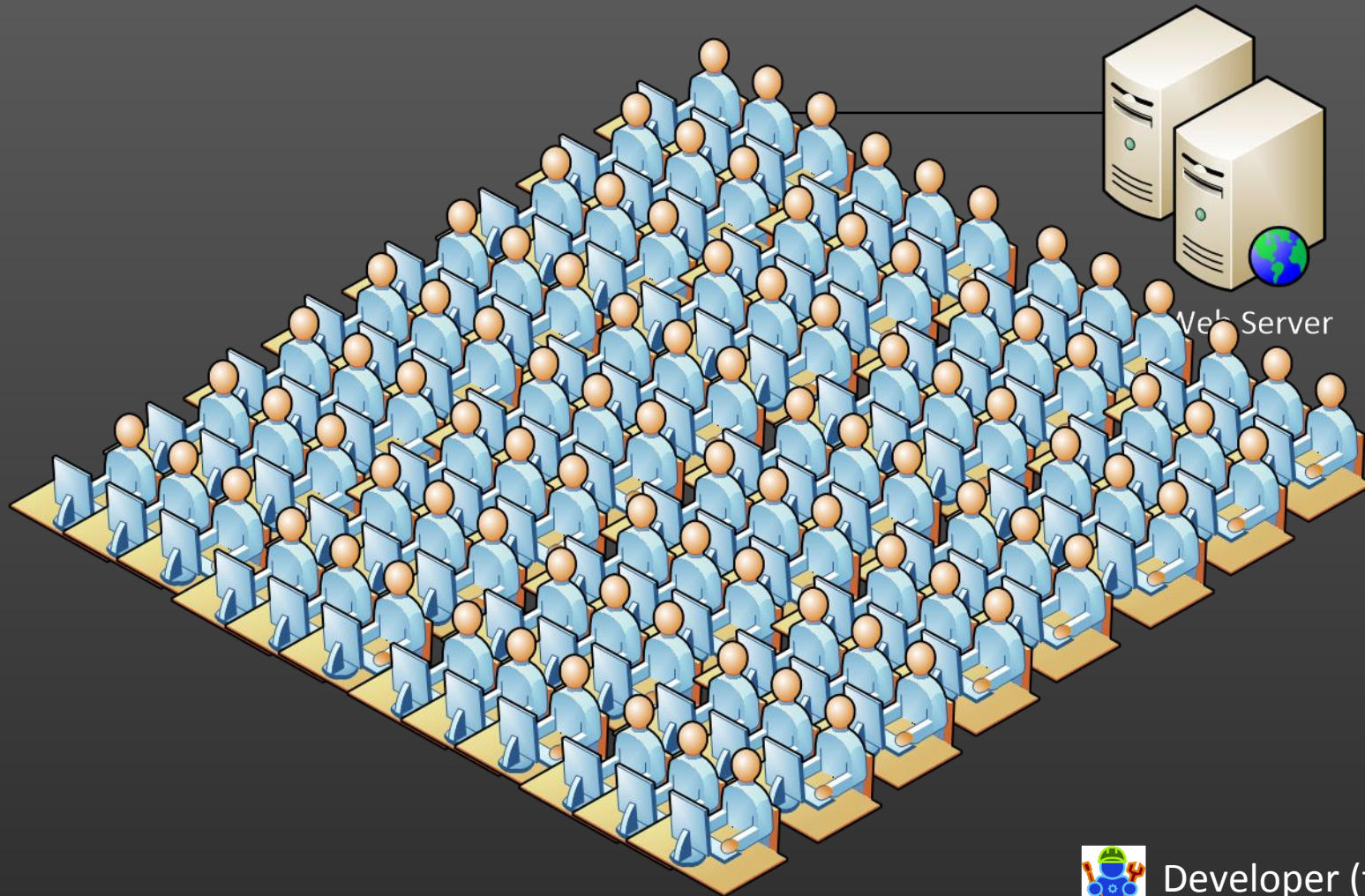
Developer



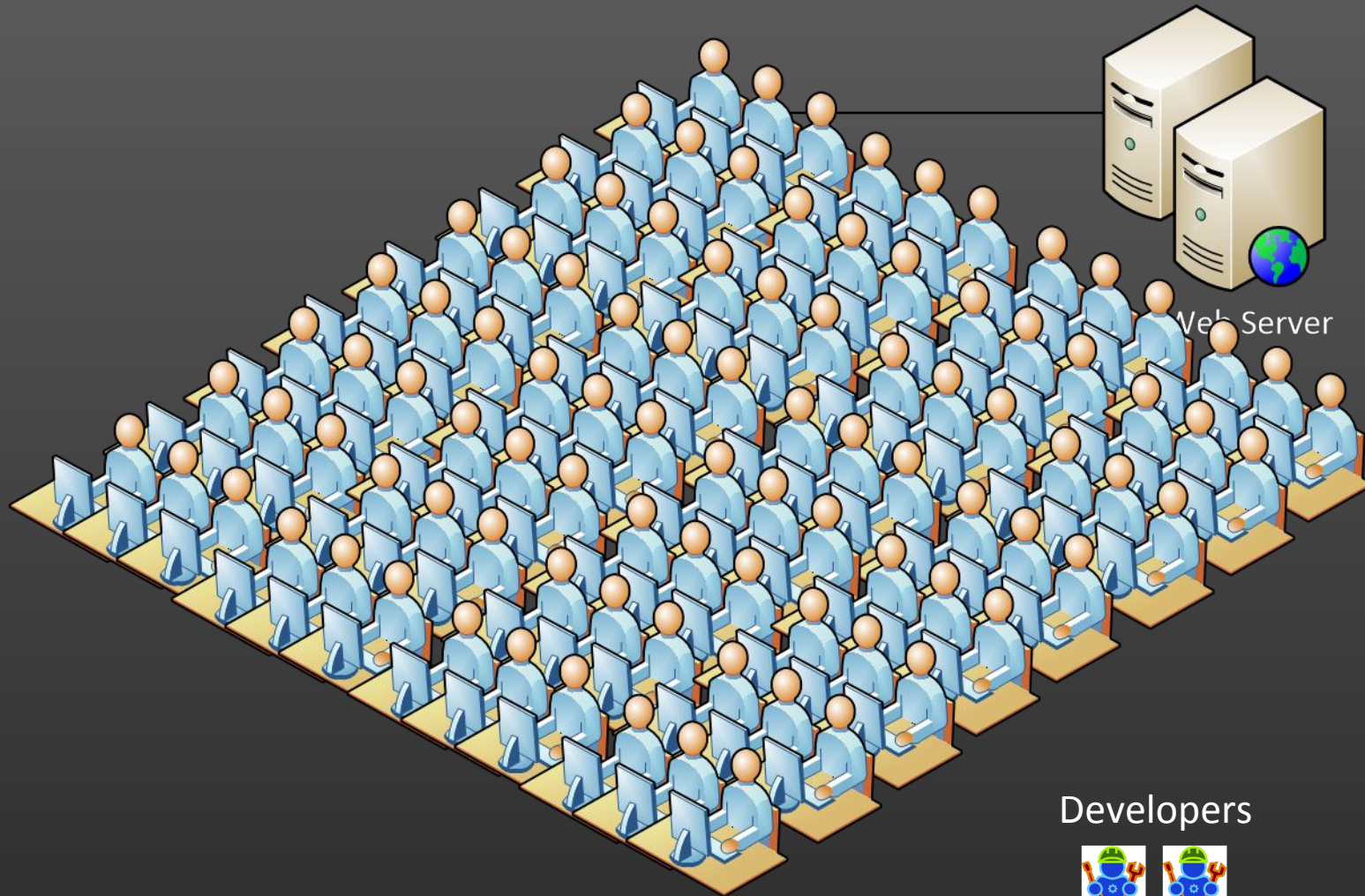
Web Server



Developer

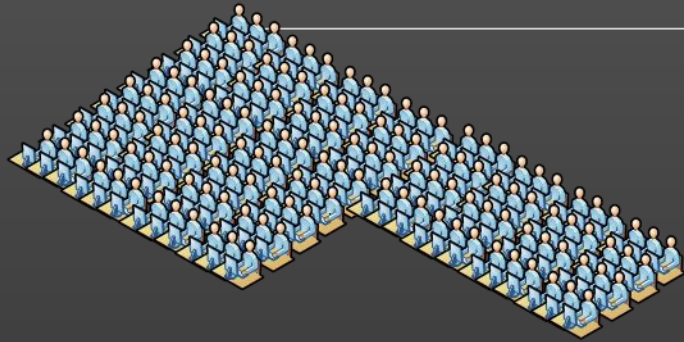


Developer (furiously optimizing)



Developers





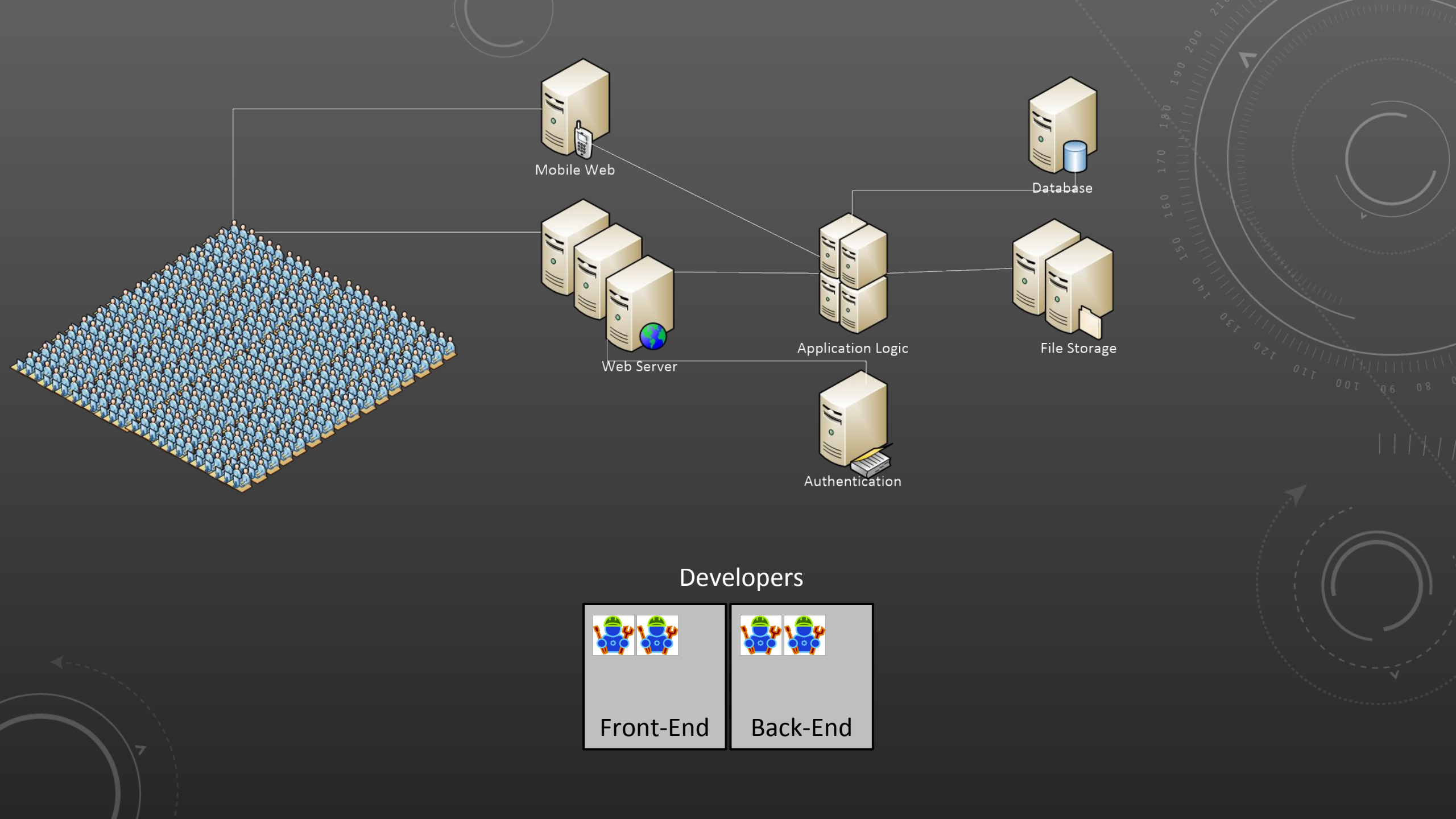
Application Logic

File Storage

Web Server

Developers





## Developers

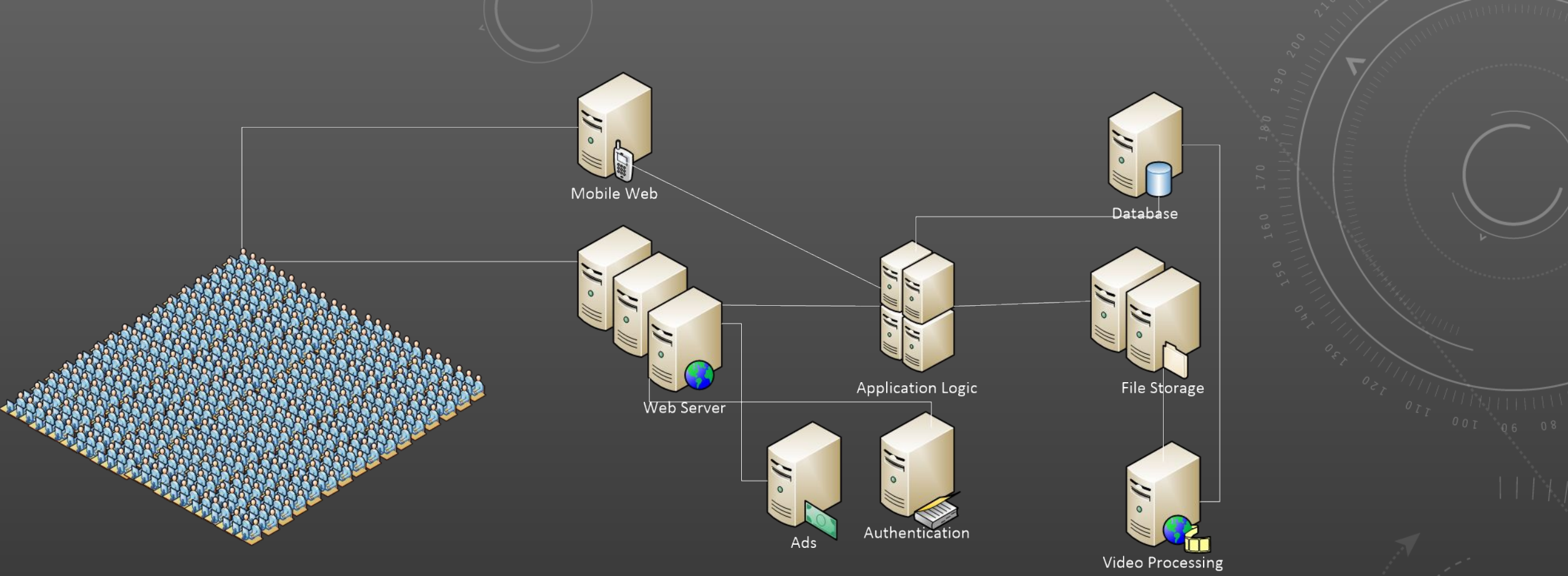


Front-End

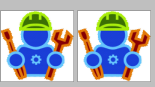

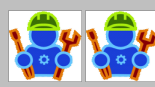
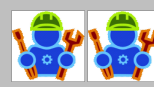


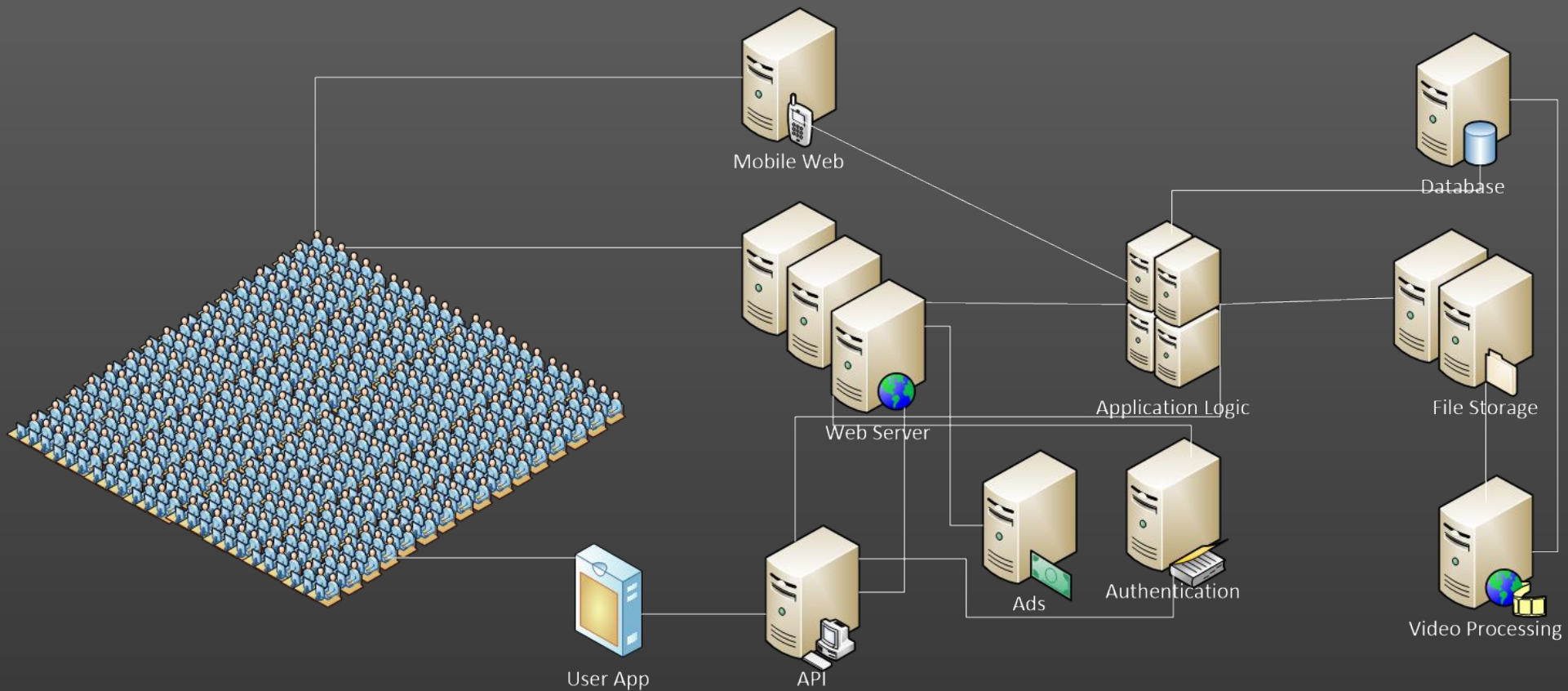
Back-End



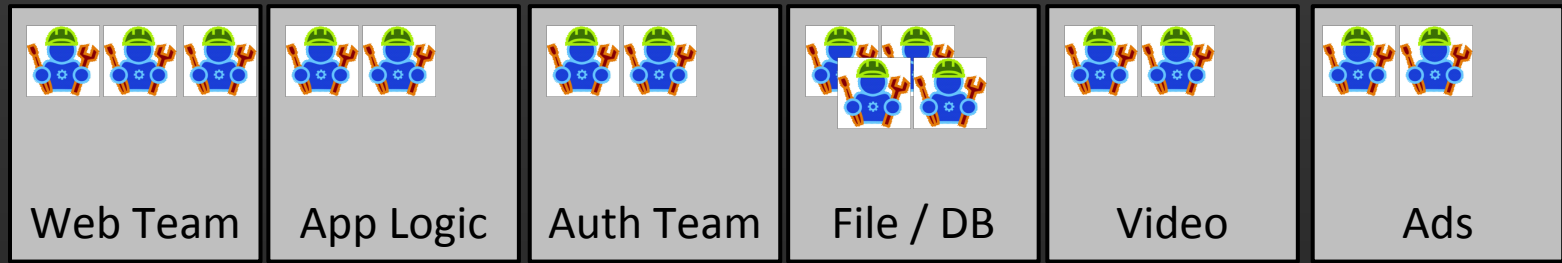


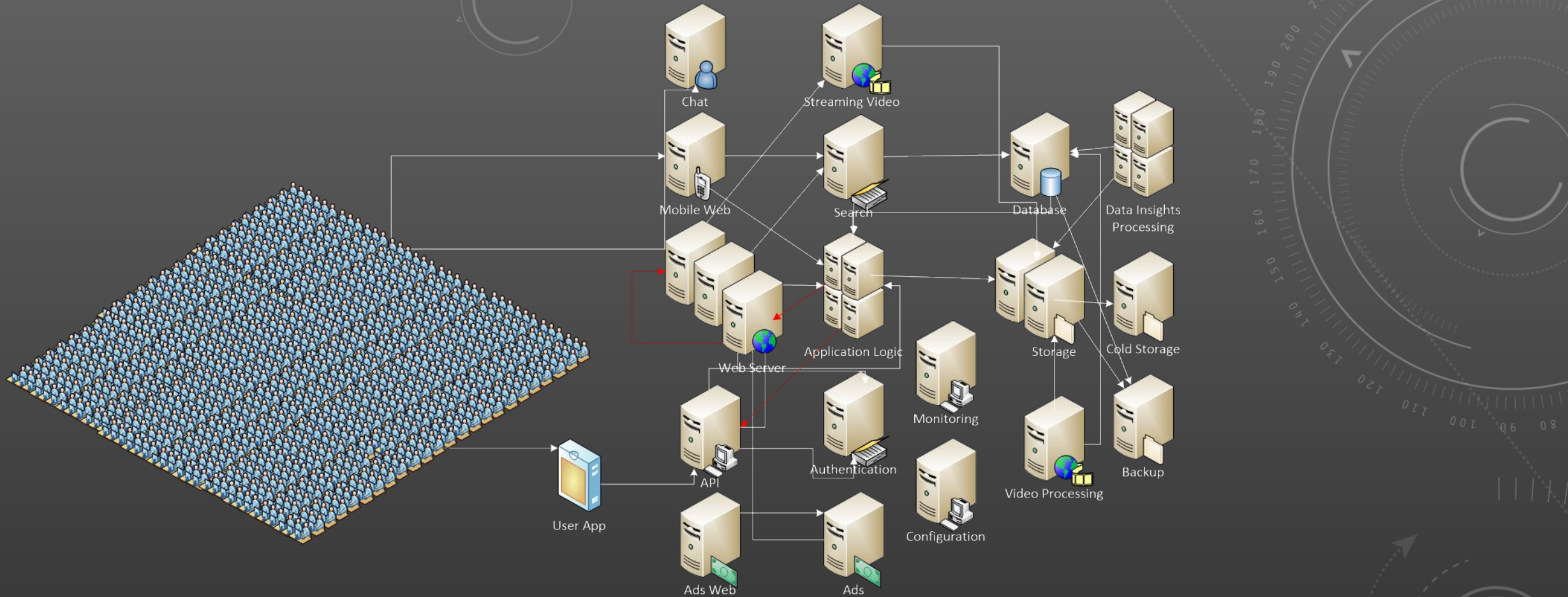
Developers

			
Web Team	App Logic	Auth Team	File / DB

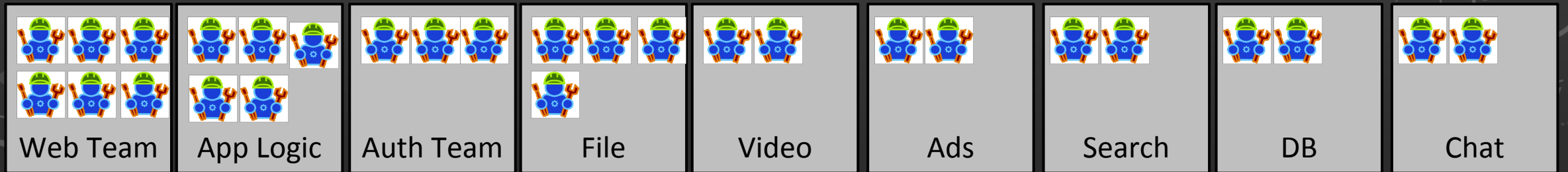


### Developers





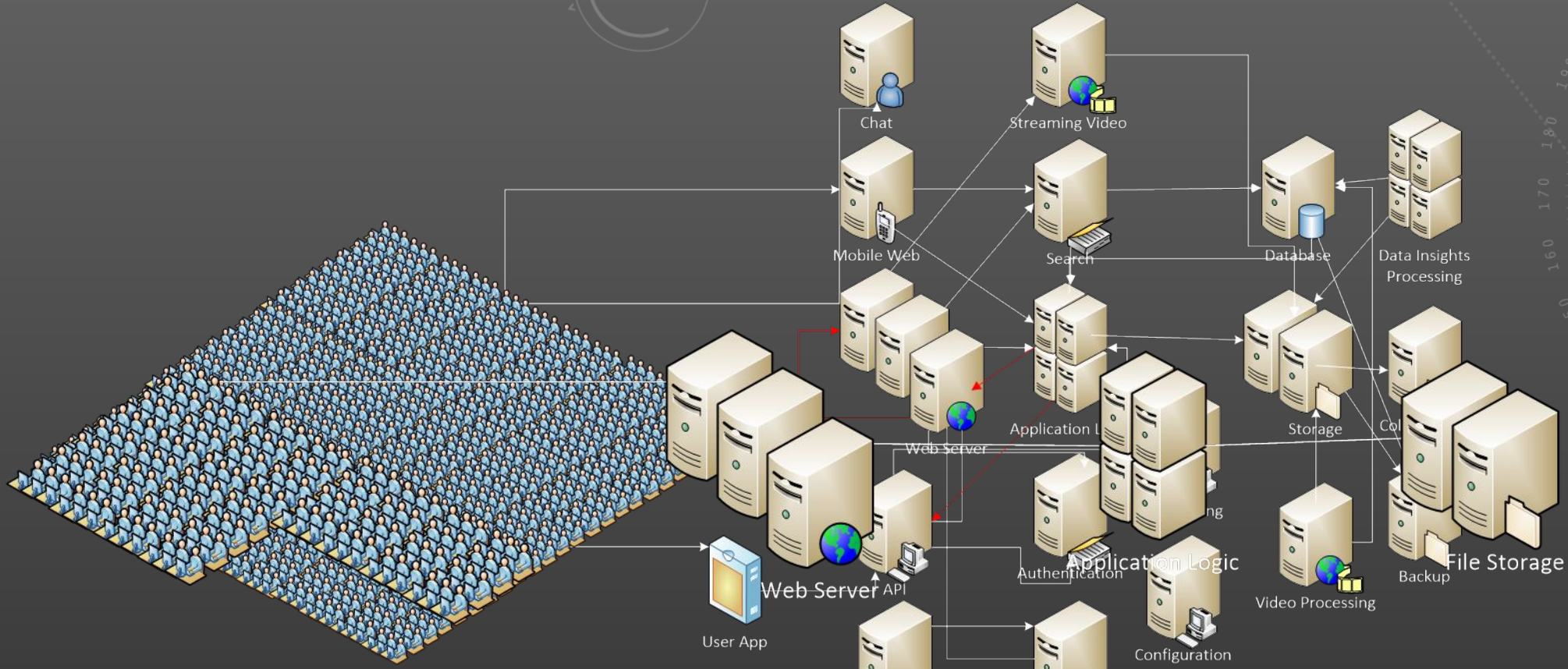
### Developers



Teams will organically implement the  
service lifecycle to fit their needs

From source control and deployment to capacity planning

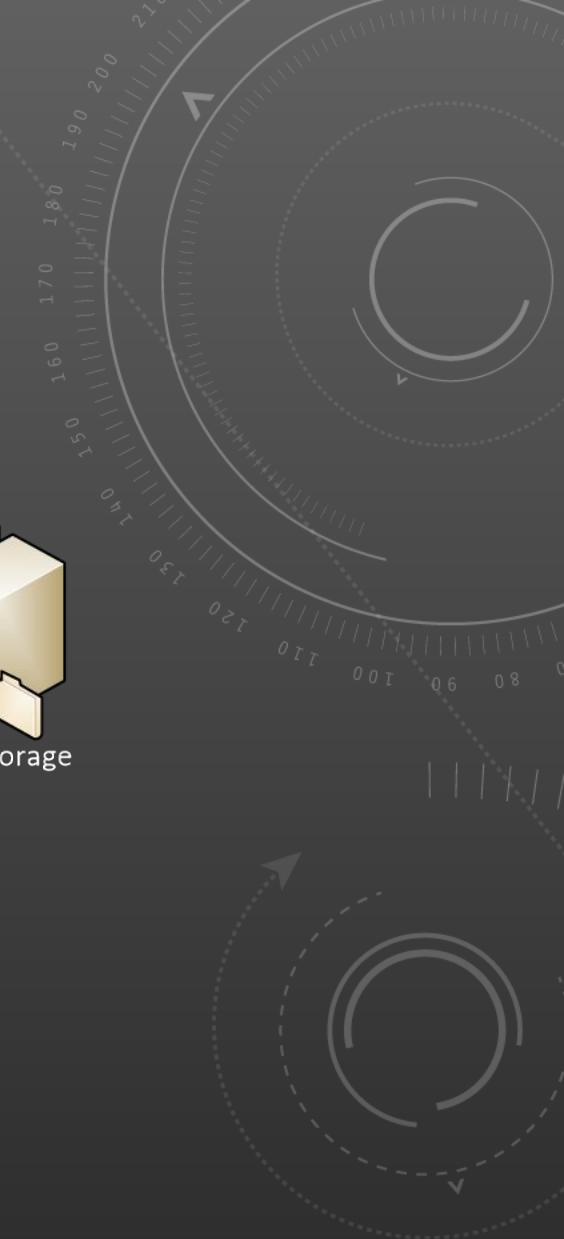




Web Team



SRE



# Welcome to The Pit of Opportunity



# Engaging with Product Teams

- Only they know where debt lies, what it looks like, where their service fails
- How do you get a product team to open up and work with you?
- Is there a common understanding of SRE, agreement on goals?

*We can't help you if you won't tell us where it hurts*

# Service roast

Pronunciation: \ 'sər-vəs \ \ 'rōst \  
*n.*

A series of meetings at which a service is subjected to good-natured but frank discussions to uncover design/process flaws, scale limits or other shortcomings



# What is a Service Roast?

- Goal: Expose and understand the warts, wrinkles, design flaws, shortcomings and problems everyone knows a service has but doesn't want to talk about
- Covers the entire service lifecycle from Development to Disaster Recovery
- Outcome: Understanding and a shared backlog of opportunities for improvement

*You can and should do this for SRE-built services*

# Why Do This?

- Builds relationships and trust between the teams
- Speeds up 'newbie to expert' process
- Exposes details that otherwise would be difficult (or painful) to learn of
- Creates a *shared* backlog of improvements

# Guidelines: Working Together

- Requires investment from SRE and product teams
- Get real contributors in the room (go away managers)
- End to end requires ~10 hours over several weeks
- 45 minute meetings avoid emotional fatigue

# Guidelines: Tone

- Clarity of purpose and tone are key
- Not an attack on the service or past choices
- 'Why' questions are judgmental

# Example Questions

✓ How does  $\${feature}$  work?

✓ When do these two pieces communicate?

✓ What part of the system handles  $\${feature}$ ?

✓ Where are user requests routed?

✗ Why did/didn't you... ?

✗ Why don't you instead... ?

✗ Why can't you just... ?

✗ Why aren't you simply... ?

# Roles

<b>Service Owners</b>	SME experts on service providing insights
<b>Roast Participants</b>	Ask questions, gain clarity on service (typically SRE)
<b>Scribe</b>	Keeps track of interesting tidbits, actions, learnings
<b>Roast Master</b>	Impartial moderator not otherwise involved in the engagement

# The Roast Master

- Impartial moderator with conflict resolution experience
- Focuses on language, tone and body language of participants
- De-escalates conversations as necessary
- Decides when to call the meeting off

*Strongly recommend implementing this role*

# Meeting Agenda

- Choose a single area or subsystem to drill into
- Moderator provides overview of guidelines and sets tone
- SME provides an overview using whiteboards, diagrams as needed
- Sessions are interactive: ask questions, clarify, dispel misinformation
- Moderator keeps conversation on topic
- Scribe tracks off-shoots for future meeting topics



# Service Roast Sample Topics

Service Overview	What is it, who uses it, where does it fit in overall
Technical Architecture	Overview, upstream dependencies, sub-components
Development Process	Source control, external dependencies, build, test, tools
Change Management / Deployment	Process, technology, cadence, gates, rollback
Configuration Management	Process, technology, source control
Demand Forecasting, Capacity Management	How do you shift load, or scale? How do you load test? Can you shed load?
SLAs, SLI, SLOs, KPIs, etc.	What are your targets? Are you meeting them?
Monitoring, Logging, Diagnostics, Tickets	How do you monitor, diagnose? How noisy?
Incident Response, production playbook, disaster recovery, backup/restore	How do you respond to issues? What is your waste case plan? Do you use it regularly?
Review of Past Outages, War Stories	What has gone wrong previously? How was it fixed?



# Meeting Closure

- At the end of the meeting:
  - The next topic is chosen
  - Adjustments are discussed for future sessions (new topics, participants, etc.)
  - The scribe summarizes key learnings and opportunities identified in a centralized doc
- At the end of the series
  - Postmortem the engagement
  - Improvement items are jointly prioritized and bugs/tasks opened

# Gotchas

- Things can be said in the room that don't leave (except the fix)
- Don't do this if you think it will degrade relationships between the teams
- Don't compare one service to another
  - Each service will be at different maturity points - that's ok!*
- When the product team is talking to each other, don't stop them - listen harder

# Summary

- A Service Roast can be a great tool to safely gain E2E service understanding
- Expectations and tone are critical success components
- Managing emotions is critical to a safe discussion environment
- Multiple, 45 minute meetings are best to cover all areas
- The roast master role helps smooth over bumps in the process

The background features several circular gauges and arrows. One large gauge on the left has a scale from 140 to 260. Other gauges are smaller and scattered across the page. Arrows indicate various directions of movement, some solid and some dashed.

# Questions?

Jake Welch

[jawelch@microsoft.com](mailto:jawelch@microsoft.com)

[@jaketwelch](#) / [#AzureSRE](#)