

Design Patterns for Scalable LAMP Infrastructure

Jon Topper
Mike Griffiths



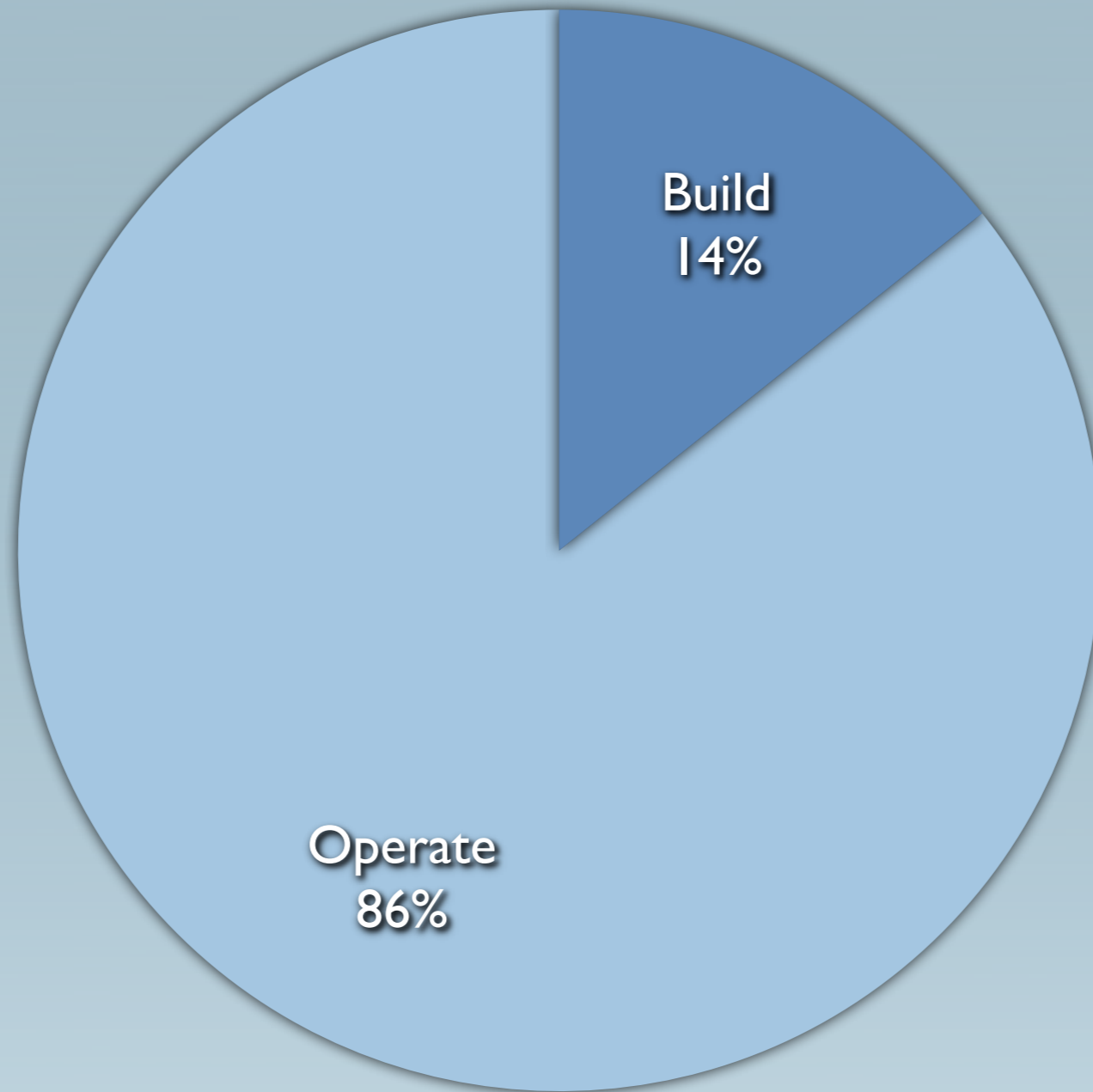
The **Scale** Factory

Who Are We?

- Mike Griffiths
 - Yahoo!
 - Proven Scaling
- Jon Topper
 - Designer Servers / Legend Communications
 - Trutap
- The Scale Factory
 - Founded 2009



3 Year Infrastructure Lifecycle



Design Pattern

“Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

Christopher Alexander



Good infrastructure

- Scalable
- Robust and available
- Manageable
- Operationally Visible
- Cost effective



Design



Stable Distribution

Avoid unnecessary change by selecting a long-term supported distribution on which to base your platform.

- RHEL / CentOS
- Ubuntu LTS
- Debian Stable



Capacity Plan

Model expected traffic to help plan your resource requirements.

Use data from:

- Existing, similar sites
- Business requirements
- Functional prototype



Carefully Consider Cloud

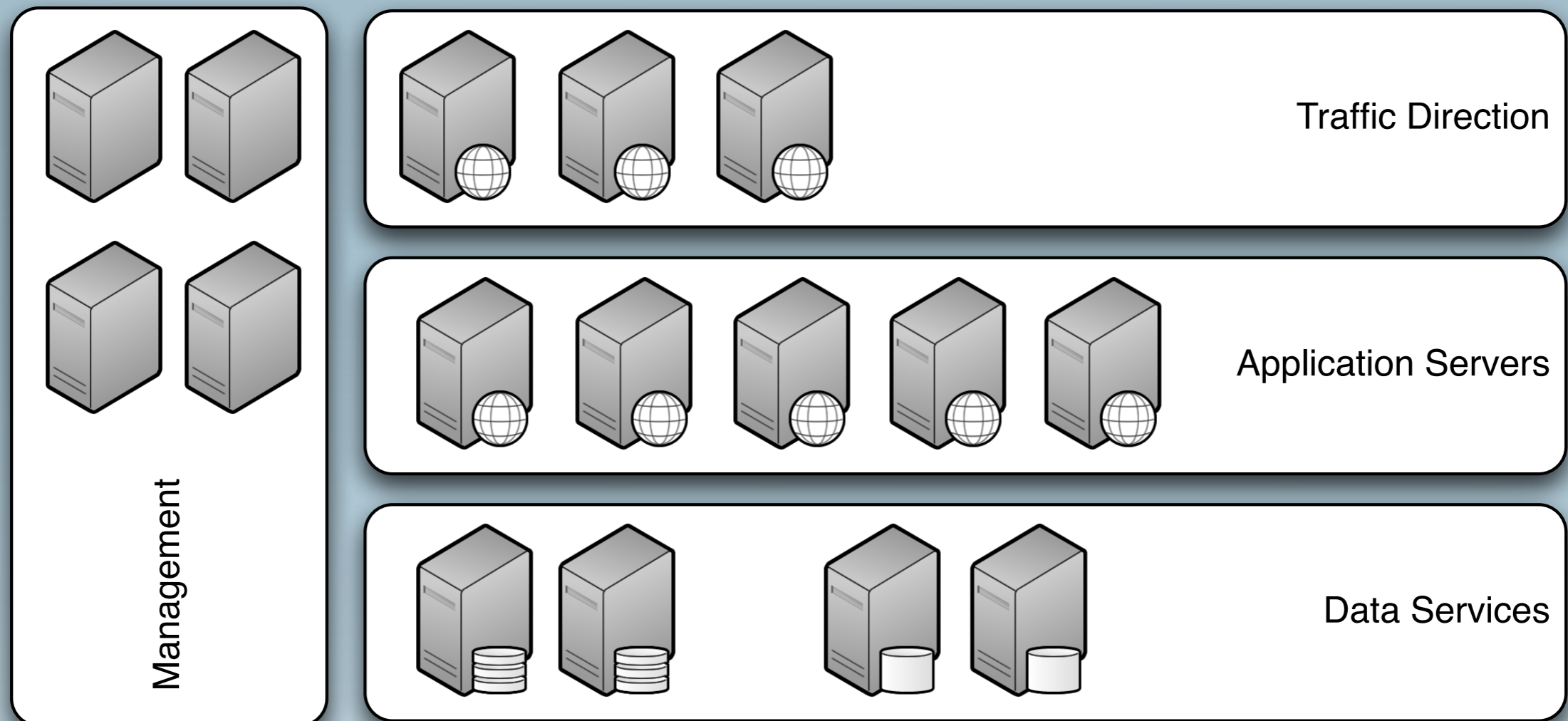
Use your capacity model to drive a decision on how you build infrastructure

- 100% dedicated hardware
- 100% cloud
- Hybrid



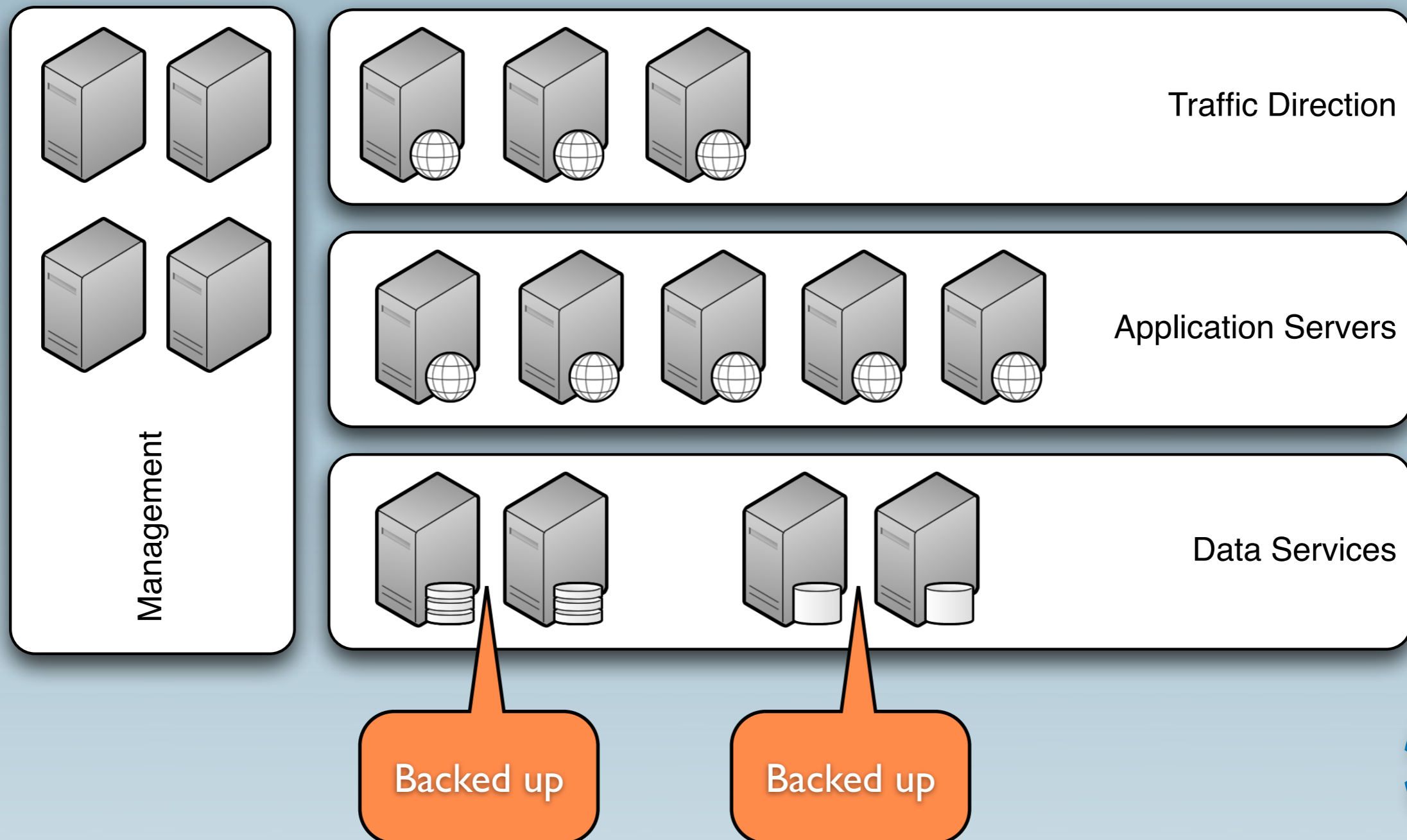
Separation of Concerns

Split each service out across its own set of servers for easier scale-out and management.



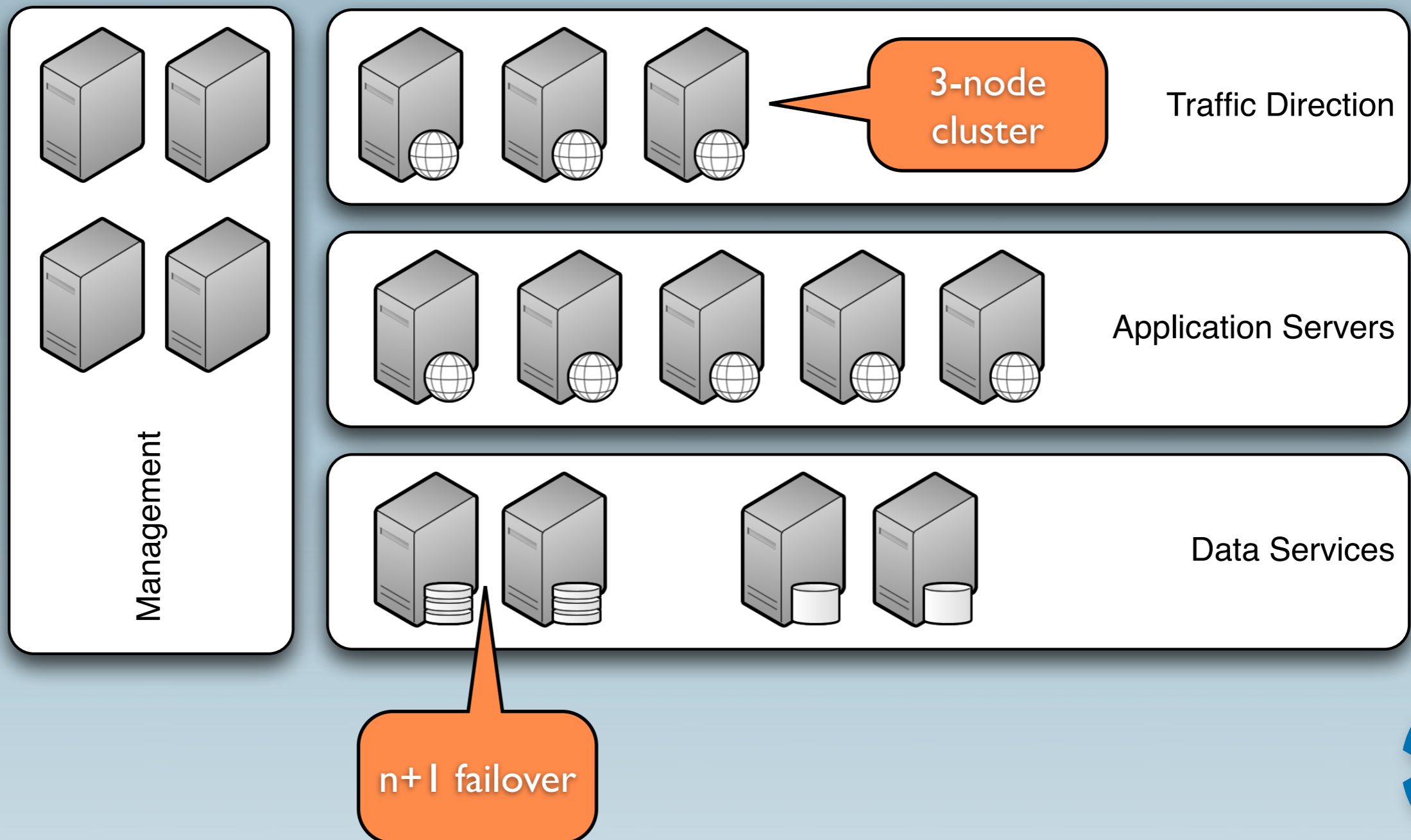
Minimise Distribution of State

Keep services that require storage of state to a minimum, for ease of backups and management.



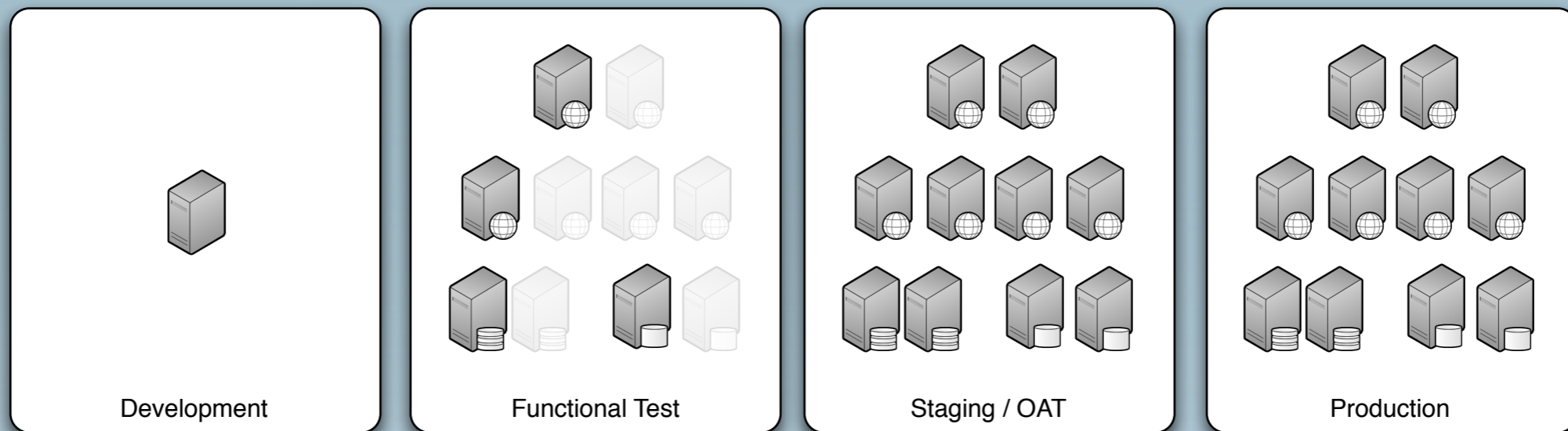
Redundancy

Use clustering or failover to ensure availability of service.



Separate Environments

Build development and staging platforms to prove application and configuration changes before they go live



Build



Application

Server Environment

Operating System



Automatically Deploy Servers

Use network boot and installer tools; or templated provisioning to build servers identically

- PXE Boot + Kickstart / Preseed
- VMWare ESXi Templates
- Amazon AMI

Application

Server Environment

Operating System



Automatically Deploy Configuration

Use configuration management tools to deploy configuration automatically from a central location.

- Puppet
- Chef
- bcfg2
- SCCM on Windows

Application

Server Environment

Operating System



Centralise Identity Management

Use a central service for identity and password management

- OpenLDAP
- Active Directory

Application

Server Environment

Operating System



Automatically Deploy Code

Use continuous integration and deployment tools to test and release software

- Jenkins / TeamCity / Go
- Capistrano / Fabric / Vlad The Deployer

Application

Server Environment

Operating System



Run



Monitor Everything

Capture as much data as possible. Store time-series data for trend analysis, and alert when thresholds are breached.

- CPU / RAM / IO / Network usage per server
- Application metrics
- Disc space usage
- Network bandwidth
- MySQL numbers
- ...etc



Capacity Plan... Continuously

Continue to plan your resource requirements based on growth expectations, new features and performance targets

Use data from:

- Your monitoring system!
- Business requirements



Aggregate Logs Centrally

Use syslog and batch processes to store logs centrally for analysis.

- Syslog data
- HTTP access / error logs
- Application-specific logs



Continuously Improve

Use available operational data to measure against KPIs and keep getting better

- Profile applications and reduce resource usage
- Feed a “Top 10” hitlist back to developers
- Review performance against capacity model
- Provision new hardware before it becomes necessary



Summary

- Don't reinvent the wheel
- 90%+ of LAMP infrastructures can follow these patterns
- Aim for simplicity over novelty



Ask for Help

There are few unique problems in this space, and several people solving them well:

- Freenode IRC ##infra-talk
- <http://www.planetdevops.net/>
- <http://devopsweekly.com/>





The **Scale** Factory

info@scalefactory.com