Implementing Effective Change Management

Karun Subramanian
IT Operations Expert

@karunops www.karunsubramanian.com

Overview

Why manage change?

Three tenets of effective change management

Rolling out changes progressively

Detecting problems with changes

Creating practical rollback procedures

Why Manage Change?

It is estimated that 75 % of the production outages are due to changes.

Inherent Problems with Changes

Infrastructure and platforms are rapidly evolving

Complexity of numerous sub-systems

Impossible to analyze every interconnection and dependency

Cannot possibly test for unknown scenarios

Tenets of Effective Change Management

Three Tenets of Change Management

The foundational aspect of effective change management is automation

AUTOMATION

Progressive rollouts

Implement progressive rollouts instead of big-bang

Monitoring

Quickly and accurately detect any issues with changes

Safe rollback

Ability to quickly and safely rolling back changes when needed

Role of Automation

To increase velocity of releases, manual operations must be eliminated

CI/CD is only effective when most of the operations are fully automated

Prevents human errors due to fatigue and carelessness

By virtue, auto-scaling requires no manual intervention

Progressive Rollouts

Deploying Changes Progressively

- **Changes to configuration files and binaries have serious consequences**
- **Reduced impact when things go wrong**
- **If we need to roll back, effort is smaller**
- **Emergency changes are an exception**

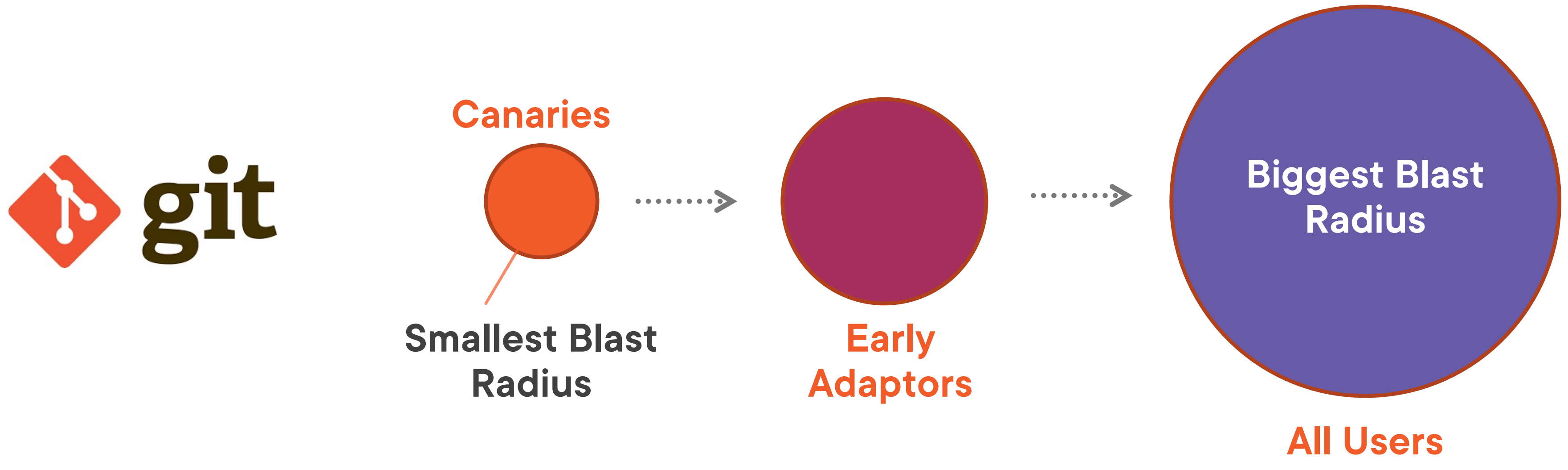
Pitfalls of Progressive Rollout

- **Rollout and rollback can get complex**
- **Lack of required required traffic can undermine the effectiveness**
- **Complicated release pipeline**
- **Release can get much longer compared to one single (big) change**
- **Detailed documentation requirements**

High Level Overview of Progressive Rollout

Release Pipeline

Code Commit → Canary Release → Early Adaptors → All Users



Canaries

Smallest Blast Radius

Early Adaptors

Biggest Blast Radius

All Users

Options for the Progression

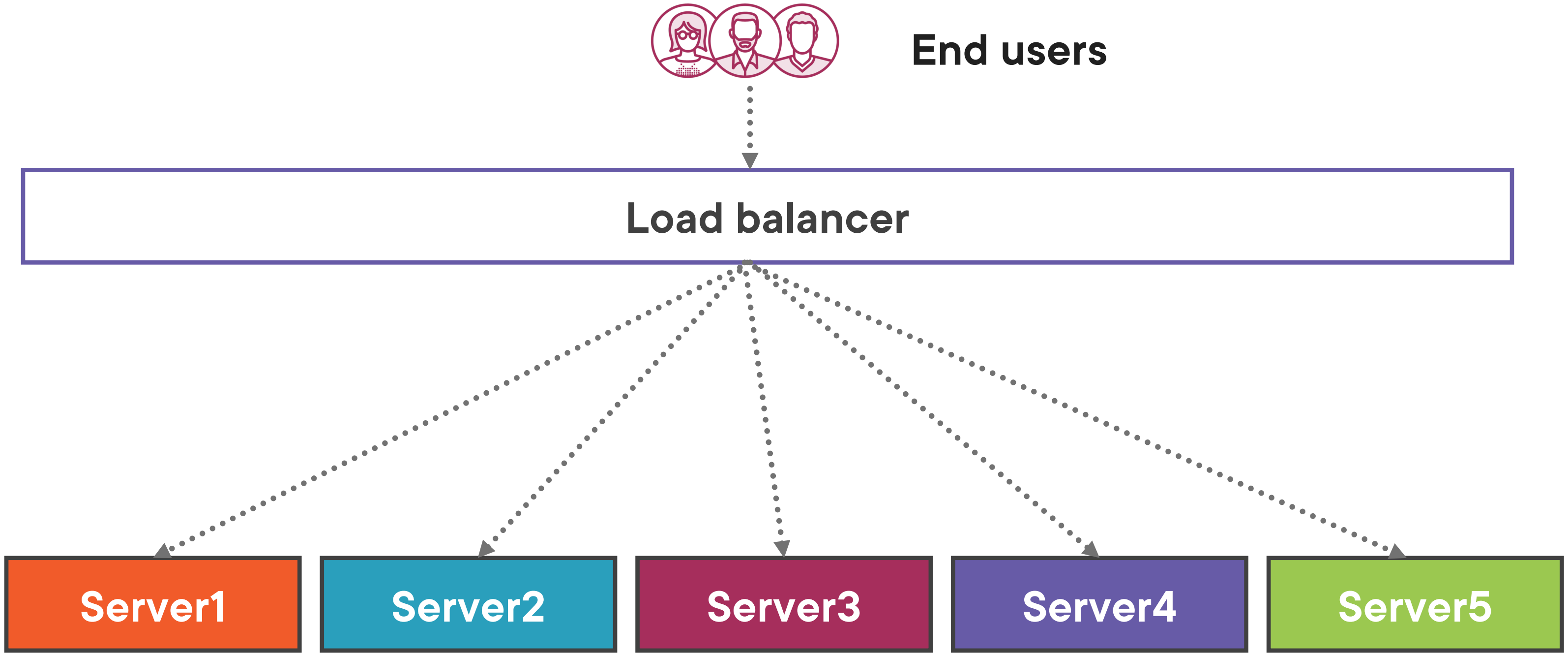
Highly application/organization dependent

For global applications, geography-based can be an option

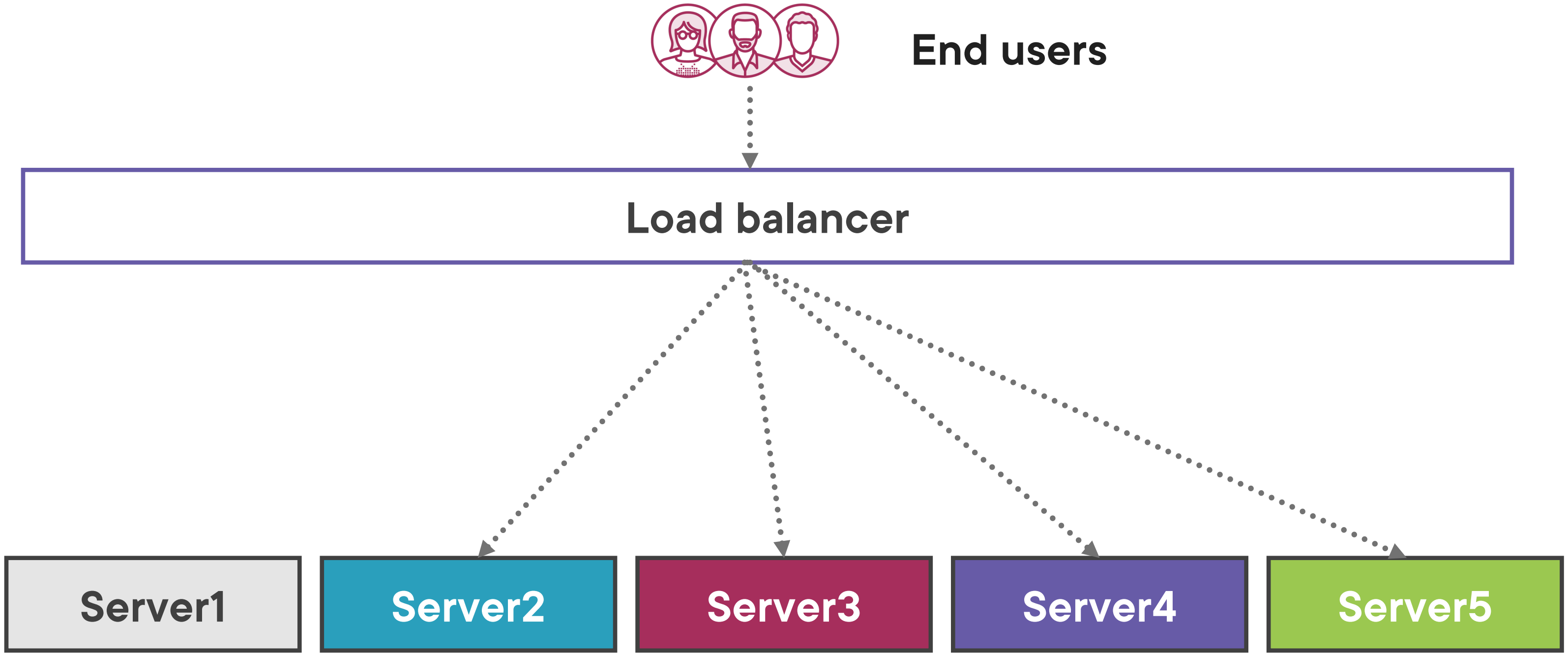
Department based:

- Canaries
- HR
- Marketing
- Customers

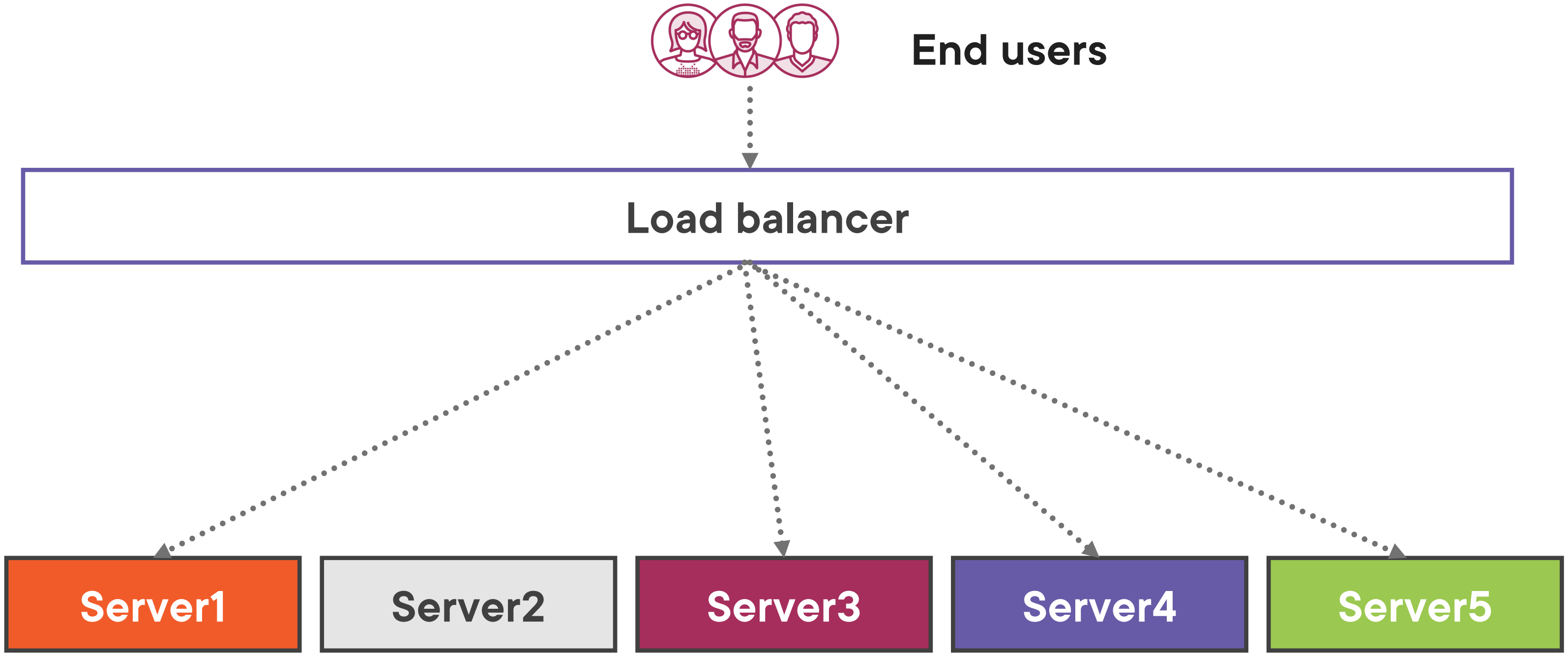
Mechanics of Progressive Rollout



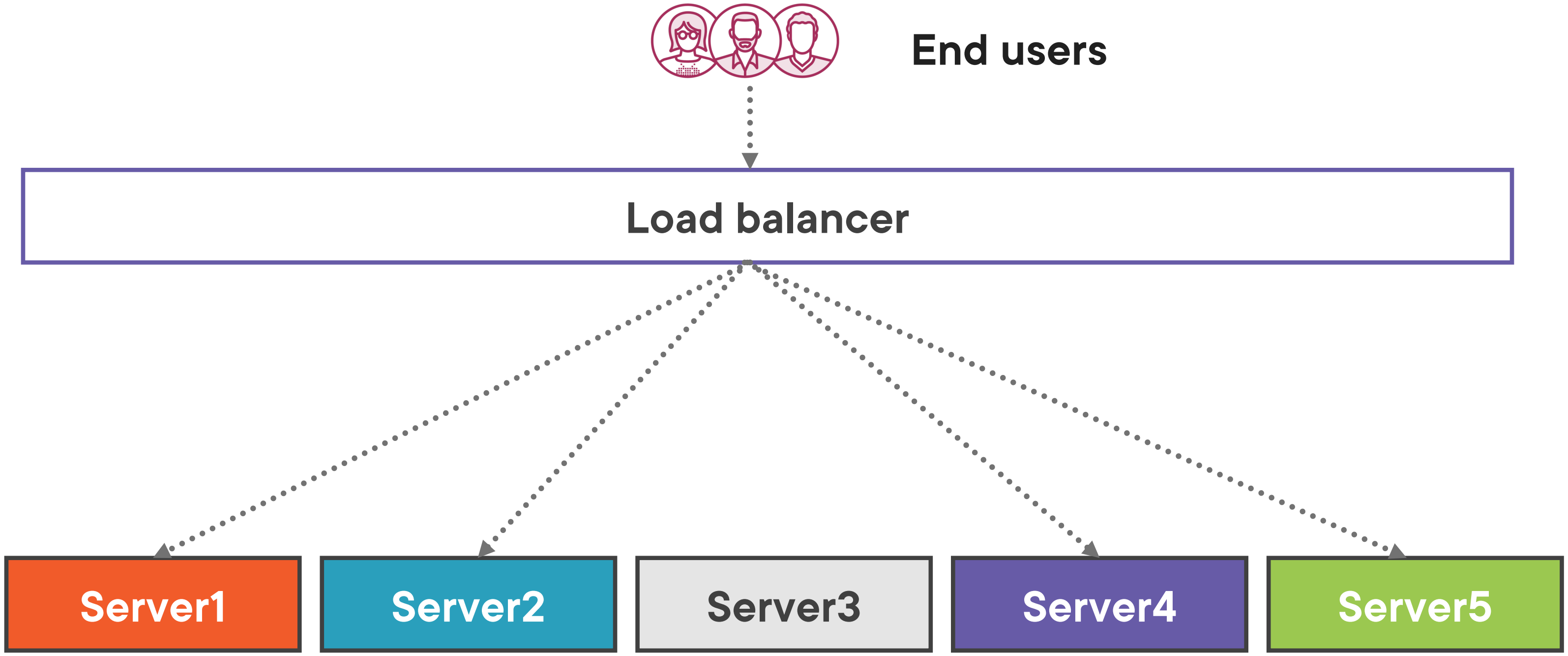
Mechanics of Progressive Rollout



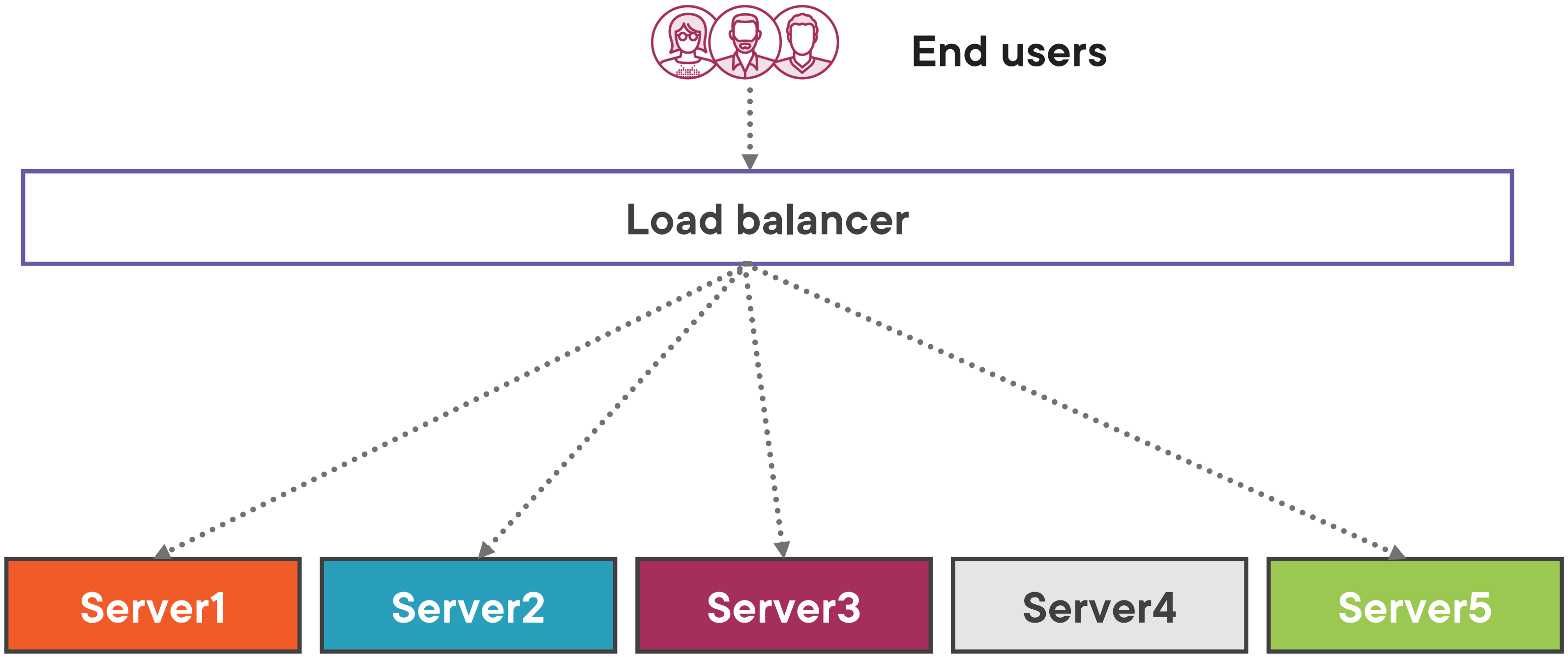
Mechanics of Progressive Rollout



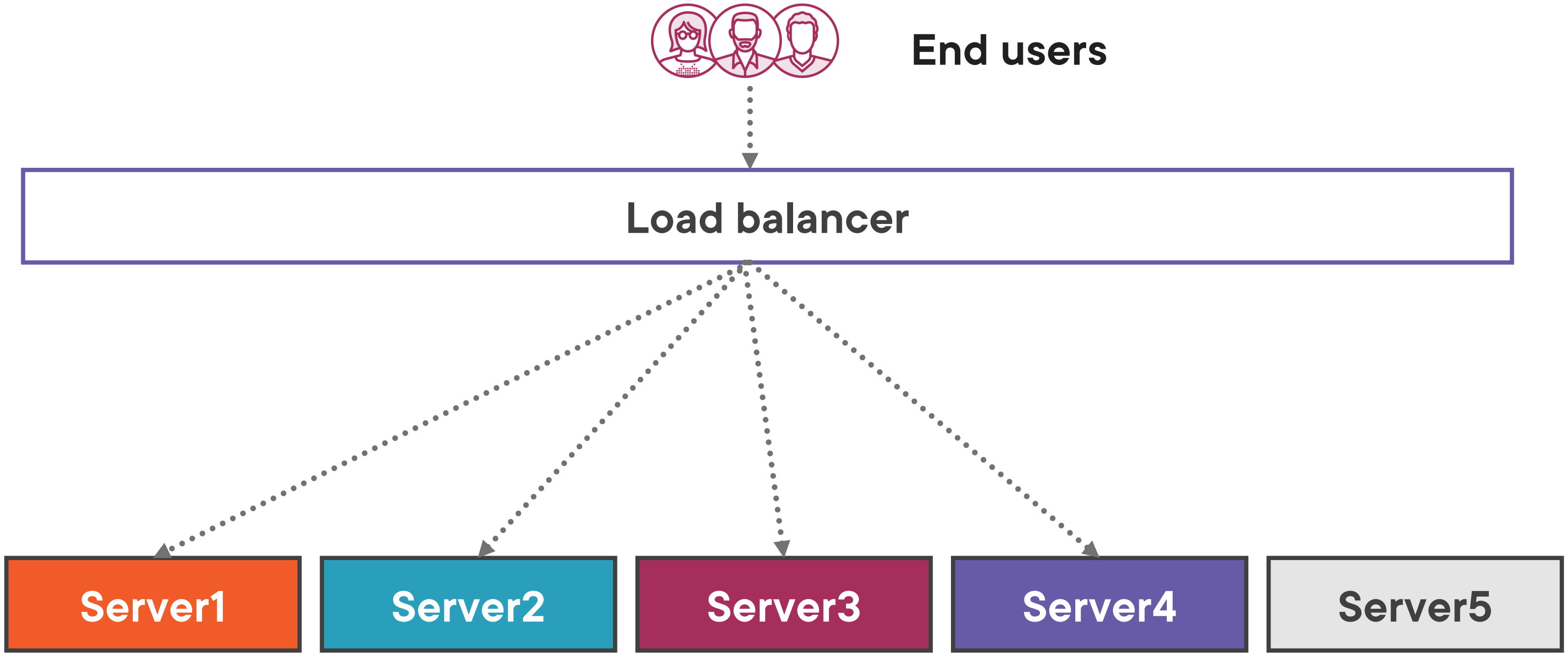
Mechanics of Progressive Rollout



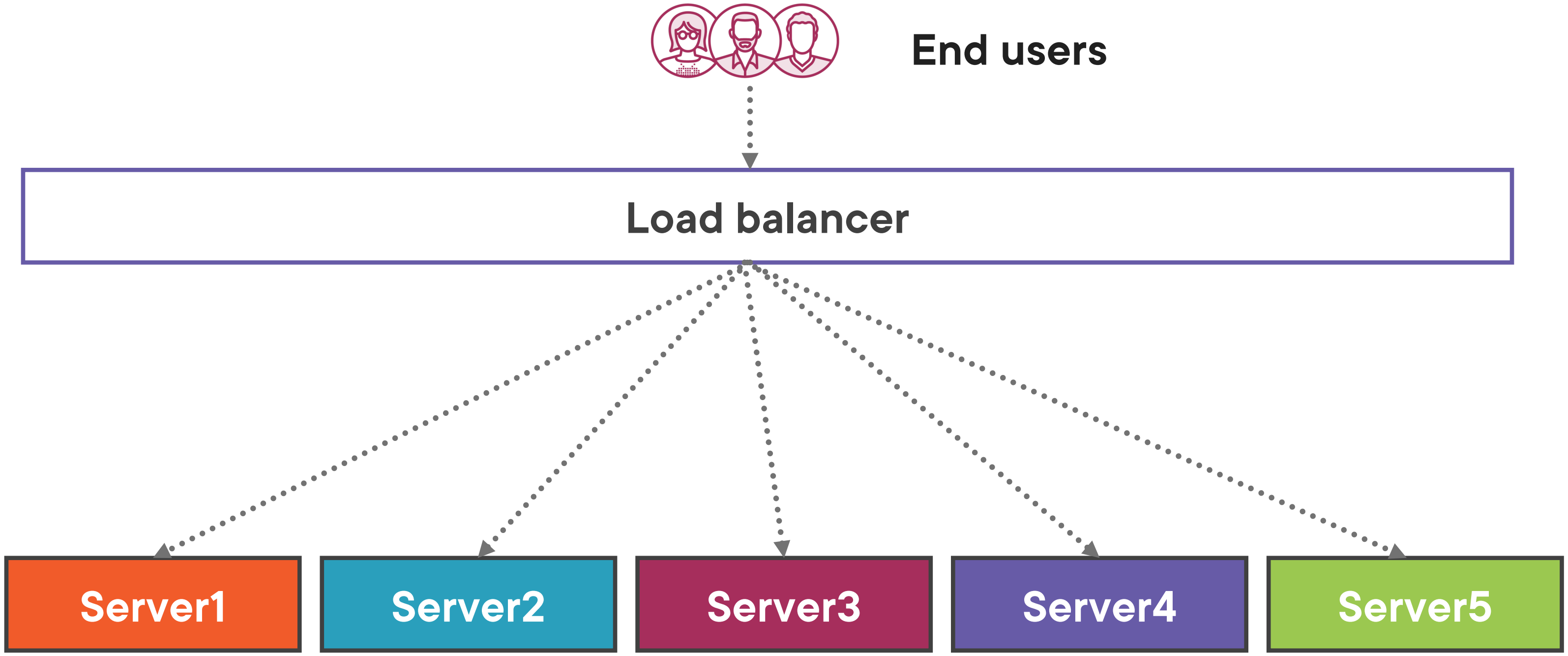
Mechanics of Progressive Rollout



Mechanics of Progressive Rollout



Mechanics of Progressive Rollout



Binary and Configuration Packages

Components of a system

- Binary (Software)
- Dataset
- Configuration

Keep binary and configuration files separate

Version controlled configuration

Hermetic configuration

Configuration as code

Monitoring

Monitoring is a foundational capability of an SRE organization

Functions of Monitoring

Visibility into service health

Alerting based on custom threshold

Trend analysis/Capacity planning

Detailed insight into various subsystems

Code-level metrics to understand behavior

Visualization and reports

Data Sources for Monitoring

Raw logs

Generally unstructured

Structured event logs

Easy to consume

Metrics

Numeric measurement of a component

Distributed tracing

Provides context

Event introspection

Examine properties at runtime

Four Questions to Ask

Speed

Resolution

Alerting

Interface

Speed

How fresh the data should be?

Ingesting data and alerting of real-time data can be expensive

Consider your SLO to determine how fast the monitoring system should be

Querying vast amounts of data can be inefficient

Resolution

Do you really need to record data every second?

Use aggregation wherever possible

Use sampling if it makes sense

Metrics are suited for high-resolution monitoring

Alerting

Integration
Can the monitoring system be integrated with other event processing tools?

Classifying
Can the alerts be classified with different severity levels?

Suppressing
Can the alerts be easily suppressed to avoid alert flooding?

Interface

Rich visualization tools

Time series data as well as custom charts

Can it be easily shared?

Can it be managed using code?

Metrics vs. Logs

Metrics	Logs
Numerical measurement of a property	Raw text data
A counter accompanied by attributes	Arbitrary text, usually with debug data
Efficient to ingest	Generally parsing is required
Efficient to query	Generally slower than metrics
May not be efficient in identifying the root cause	Most of the times you will need raw logs to determine the root cause
Suitable for low-cardinality data	No strict requirments

Alert with metrics;
Analyze with metrics and logs.

Four Golden Signals to Monitor

Latency

Errors

Traffic

Saturation

Monitoring Resources

CPU

Memory

Disk i/o

Disk volume

Network bandwidth

Three Best Practices

Configuration as code

Makes it easy deploy monitoring to new environments

Unified dashboards

Enables us to reuse dashboards

Consistency

Naming convention for objects

Rolling Back Changes

Why Rollback?

Avoid user impact

Buy time to fix bugs

With fine grained rollback, minimize the overall impact

To support canary testing

Combined with progressive rollouts, possible to eliminate user impact

Rollback fast; Rollback often.

Rollback

- Automation is the key
- Toggle flags to dynamically rollback
- Often preferred to rollback the entire release
- Use package management with version numbers and/or labels
- A rollback is s still a change

Roll Forward

Upgrade software that includes the fixes

May not be always possible; May have to run the system in degraded status until upgrade is available

Roll forward may be safer than Rollback

Summary

Three tenets of an effective change management system

- Progressive rollout
- Monitoring
- Safe and fast rollbacks

Automated builds, tests and releases

Use canaries for catching issues earlier (canaries are not a replacement for testing)

Monitoring should be designed to meet SLO

Alert with metrics, analyze with metrics and logs

Up Next:
Implementing SRE Best Practices