The Multi-Workload Challenge: Predictable Failover Across the Stack

01/22/26

Akshay Pratinav,
Staff Software Engineer, Intuit

Agenda

- Problem & Motivation
- Principles & Mental Model
- Architecture Diagram
- Outcomes & Takeaways

Problem & Motivation

Outages Are Normal

Regional failures are inevitable

Microservices amplify blast radius

Unpredictable recovery causes the real damage

Why Microservices Make It Worse

- APIs, databases, async pipelines, routing layers
- Each layer fails differently
- Recovery is gated by the slowest workload

Recovery, Not the Outage

Outages are unavoidable

Recovery determines customer impact

Humans under pressure are the weakest link

The Hidden Problem

Fragmented, team-specific failover approaches

Manual scripts and tribal knowledge

Tested only during real incidents

Why Runbooks Don't Scale

Context switching during incidents

Tribal knowledge risk

Error-prone manual execution

Principles & Mental Model

What We Mean by Boring Failover

- Predictable and repeatable
- Fully automated
- No heroics or guesswork

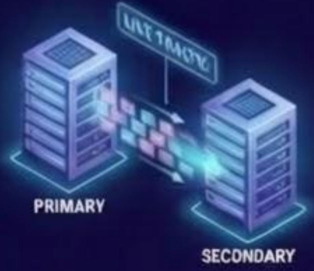
Declarative Intent

Services declare what they need

Platform decides how to execute

Consistent behavior across workloads

Fail Over the Entire Stack



Compute

Async and batch workloads

Datastores and routing

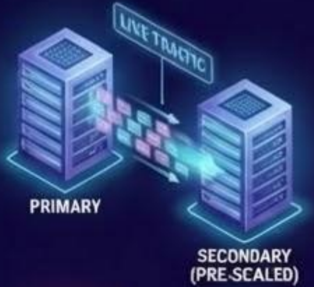
Workload Diversity Matters

Different RTO / RPO requirements

Customer-facing vs background workloads

Cost vs availability tradeoffs

Pre-Scale Capacity

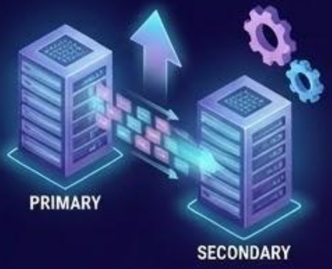


Secondary region pre-scaled using live traffic

Avoid cold starts during outages

Balance speed and cost

Data Is the Hard Part



Automated database and cache promotion

Safety and consistency checks

Correctness over speed

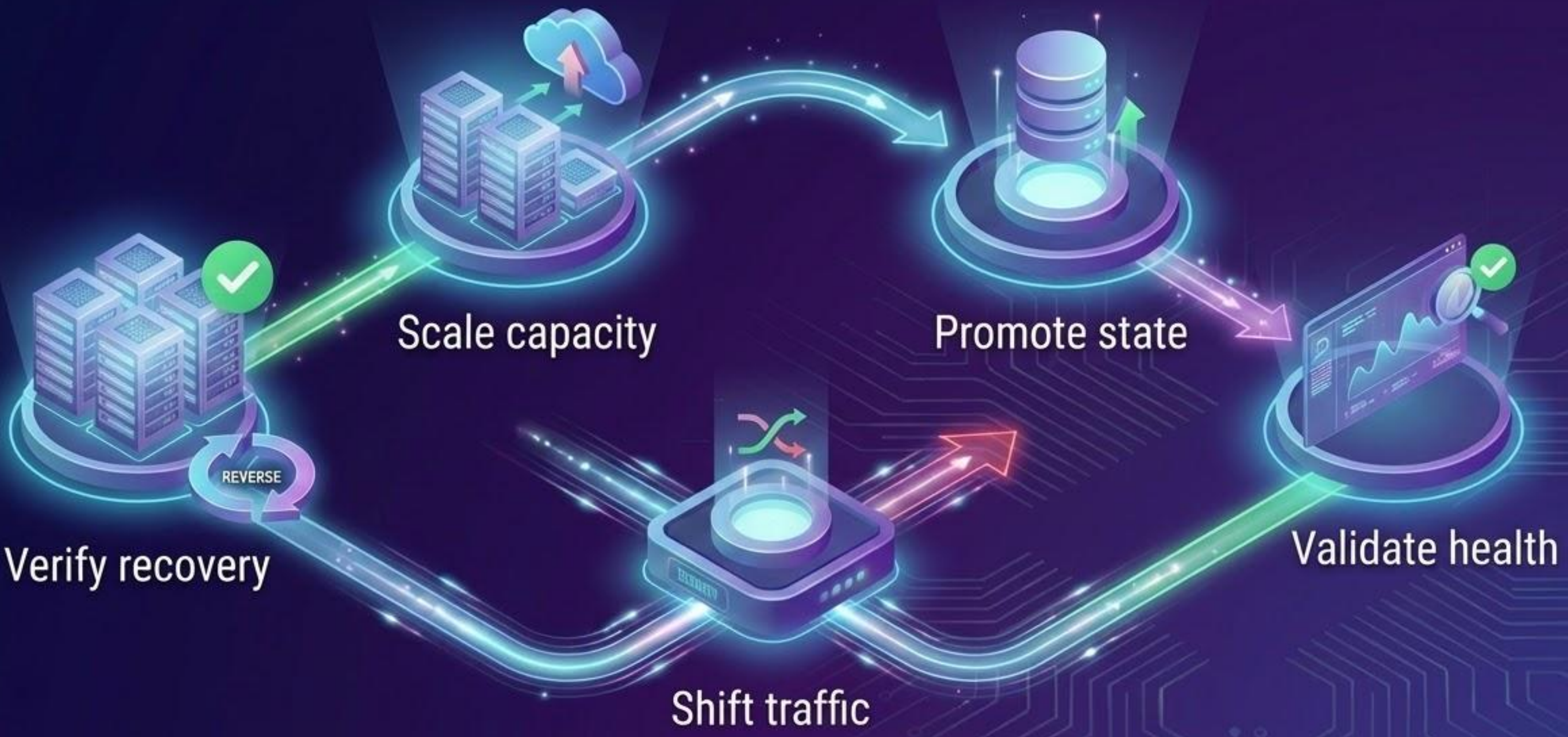
Traffic as a Control Plane

DNS and service mesh–based routing

Health–driven traffic shifts

Observable and reversible

Failover Is a Workflow



Architecture Diagram & YAML schema

DR Procedure Example

```
stages:
```

Stage 1

```
  prescale-east:
    agent-id: "armador/v1"
    capacity-compute:
      type: 'scale'
      cluster: 'cluster1'
      namespace: 'namespace1'
```

Stage 2

```
  switch-db-primary-to-east:
    agent-id: 'database/v1'
    fail-over-global-clusters:
      global-cluster-identifier: 'ewok-sample-app-global-cluster-e2e'
      assume-role: 'arn:aws:iam::682033466980:role/database-switchover-role'
      switch-over: true
```

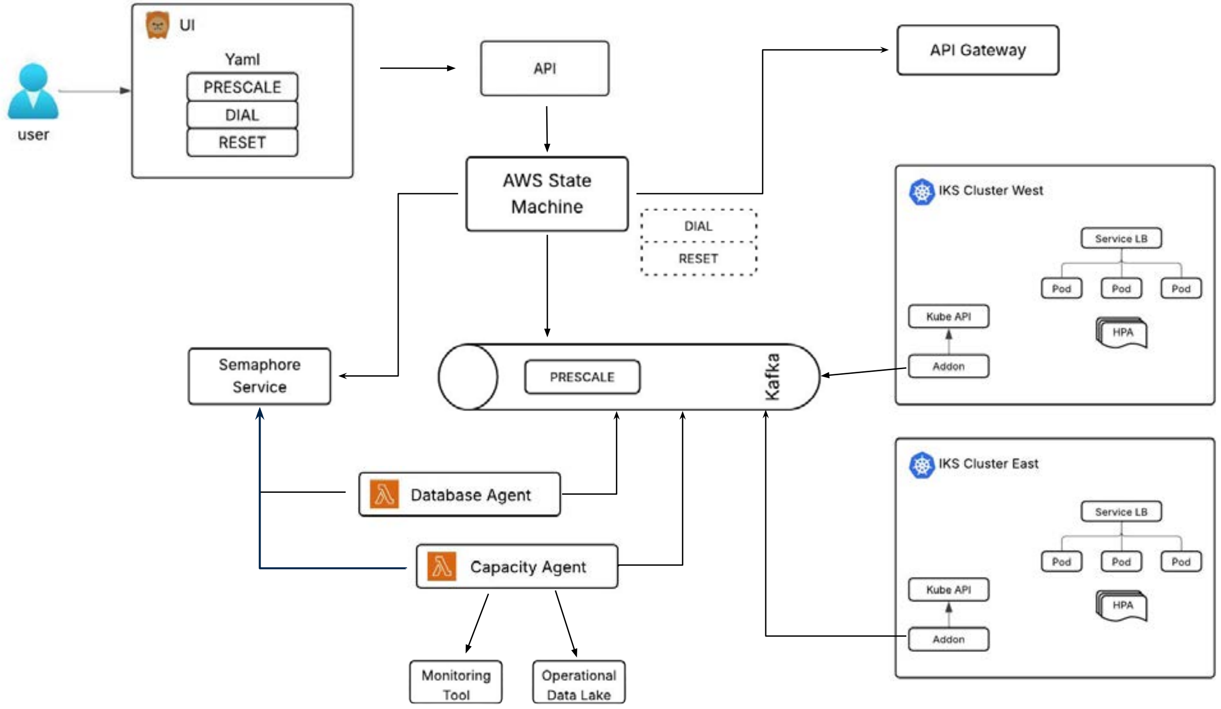
Stage 3

```
  dial-traffic:
    endpoints:
    - name: 'e2e.intuit.service1.com
      failover-to-region: 'us-east-2'
```

Stage 4

```
  reset-east:
    agent-id: 'armador/v1'
    capacity-compute:
      type: 'reset'
      cluster: 'cluster1'
      namespace: 'namespace1'
```

High Level Architecture



Safety & Operations

Guardrails Prevent Cascading Failures

Health gates before traffic shift

Dependency validation

Automated rollback paths

Operational Readiness

Health gates and dependency checks

Drift detection

No special DR-only configs

Game Days Without Fear

Regular regional failover drills

Low-stress, repeatable execution

Confidence during real incidents

Outcomes & Takeaways

Anti-Patterns to Avoid

Ad hoc scripts

Untested assumptions

High cognitive load during outages

Outcomes at Scale

Faster recovery times

Lower operational overhead

Predictable and boring incidents

Cultural Shift: From Fear to Confidence

Less incident stress

More willingness to practice

Higher engineering confidence

Key Takeaways

Standardize failover across workloads

Use declarative intent

Pre-scale intelligently

Practice regularly

Thank You

akshay_pratinav@intuit.com

www.linkedin.com/in/akshaypratinav