KadelLabs
serving through technology

TECH BLUEPRINT

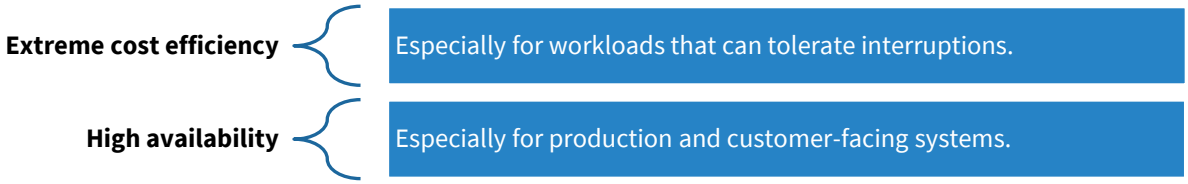
Smart Spotting — An Intelligent Spot–On-Demand Orchestration Framework for AWS Fargate

A Cost-Optimized, Event-Driven, Highly Available Container Runtime Strategy

Table Of Contents

1. Executive Summary

Cloud-native businesses demand two outcomes simultaneously:



AWS Fargate Spot offers savings up to 70% compared to On-Demand, but comes with the risk of termination with a two-minute interruption notice. Traditional deployments rely either entirely on Spot (risky) or entirely on On-Demand (expensive).

Smart Spotting is Kadel Labs' advanced IaC-driven orchestration framework that uses a dual-track service strategy to dynamically balance Spot and On-Demand capacity in real time, ensuring:

- Maximum savings through optimal Spot utilization
- Guaranteed uptime by automatically launching On-Demand tasks during interruptions
- Full automation of detection → failover → recovery
- Event-driven infrastructure based on Fargate Spot interruption notifications
- Seamless rollback to Spot as capacity becomes available again

Smart Spotting has delivered significant cost reductions for enterprise-scale container workloads while maintaining operational stability.

2. Introduction: Why Smart Spotting

Fargate Spot is an excellent compute strategy for stateless, fault-tolerant workloads. However:

- Fargate Spot can be reclaimed by AWS anytime.
- There is no native auto-failover from Spot → On-Demand.
- Organizations require predictability of workload uptime, even when Spot capacity disappears.
- Manual intervention introduces human error and downtime.

Smart Spotting solves all of the above through a fully autonomous workflow.

3. What Agentic AI Actually Is

Smart Spotting uses two or more Fargate Spot tasks as the baseline configuration for an ECS Service. When AWS sends a Spot interruption notice:

1. A CloudWatch event fires immediately.
2. A Lambda function (or Step Function) receives the event
3. The system automatically launches replacement On-Demand Fargate tasks.
4. The system keeps retrying Spot placements in the background.
5. Once Spot capacity becomes available again, Spot tasks start.
6. On-Demand tasks are gracefully scaled down to maintain cost efficiency.

This creates an intelligent, self-healing Spot–On-Demand equilibrium.

KadelLabs
serving through technology

4. Code

Below is a full working IaC example with Terraform, along with the Lambda Python code provided in a separate file (click the link to access each):

- **How-to-use document :** View the setup and execution steps.
- **main.tf :** Contains all AWS infrastructure for Smart Spotting (ECS Fargate + Spot, EventBridge, Lambda, Step Functions, SNS, SQS DLQ, etc.)
- **lambda_smartspot.py :** Lambda controller that:
    - reacts to Spot interruption events
    - scales up On-Demand
    - starts the Step Functions retry workflow

4.1 Components Involved

Component	Purpose
Amazon ECS (Fargate)	Runs containerized workloads using Spot & On-Demand capacity.
ECS Service with multiple task definitions	Spot task set + On-Demand task set (same container image).
EventBridge Rule	Captures Spot interruption notifications.
SNS Topic (optional)	For messaging/debugging.
Lambda Function / Step Function	Performs the orchestration logic: scale up On-Demand, retry Spot.
CloudWatch Metrics + Alarms	Observability and automated guardrails.
IaC (Terraform / CloudFormation)	Infrastructure automation and deployability.

5. Spot Interruption Notification Flow (Deep Technical Breakdown)

When AWS plans to reclaim Spot capacity, the following happens:

5.1. ECS Fargate Spot Interruption Signal

AWS emits a structured event:

```
{
 "version": "0",
 "id": "abcd-efgh-1234-5678",
 "detail-type": "ECS Task State Change",
 "source": "aws.ecs",
 "account": "123456789012",
  "time": "2025-01-01T10:00:00Z",
 "region": "us-east-1",
 "resources": ["arn:aws:ecs:us-east-1:123456789012:task/cluster-name/task-id"],
 "detail": {
  "lastStatus": "RUNNING",
  "desiredStatus": "STOPPED",
  "stopCode": "SpotInterruption",
  "stoppedReason": "Your Spot Task is being reclaimed."
 }
}
```

This is pushed to EventBridge.

5.2. EventBridge Rule

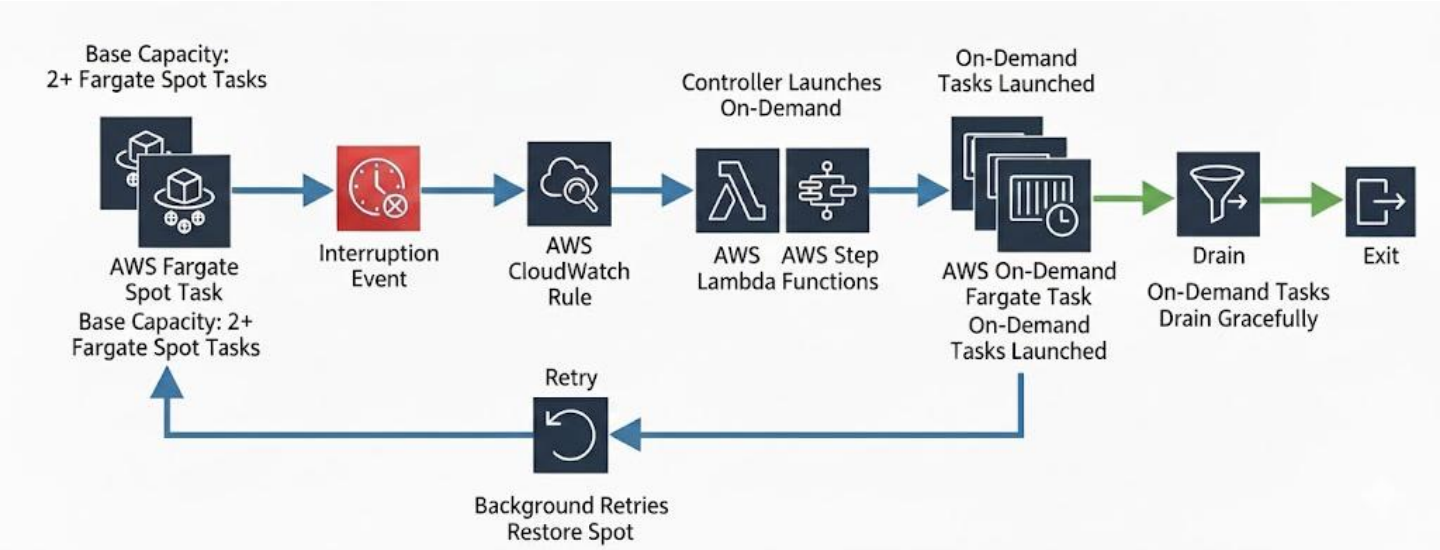
Event Pattern:

```
{
 "source": ["aws.ecs"],
 "detail-type": ["ECS Task State Change"],
 "detail": {
  "stopCode": ["SpotInterruption"]
 }
}
```

This rule triggers a Lambda.

KadelLabs
serving through technology

6. Smart Spotting Workflow



STEP 1 — Spot Task Interruption Detected

EventBridge → Lambda invocation.

Lambda loads metadata:

ECS Cluster	Service ARN	Task ARN
Task definition revision	Current running Spot tasks	

STEP 2 — Launch On-Demand Replacement

Lambda calls:

```
UpdateService
 desiredCount = existingDesiredCount + 1
 capacityProviderStrategy:
  - FARGATE_OD: weight 1
```

This ensures a new On-Demand task starts immediately.

STEP 3 — Retry Spot Placement in the Background

Lambda schedules:

- A retry job (CloudWatch scheduler)
- Or triggers Step Function with backoff retry

Retry logic checks every 30–60 seconds:

- Attempt to scale Spot tasks back to original count
- Stop retry only after placement success

STEP 4 — When Spot Returns

Spot placement success detected by:

- ECS Service deployment events
- Desired count achieved

Lambda then reduces On-Demand:

```
UpdateService
 desiredCount = originalBaselineSpotCount
```

This ensures max savings.

STEP 5 — Graceful On-Demand Scale Down

Smart Spotting ensures safety:

- Drains traffic from On-Demand task
- Waits for connections to terminate (load balancer deregistration delay)
- Verifies Spot tasks remain healthy before terminating On-Demand

7. IaC Implementation (Terraform + Optional CloudFormation)

Below is a fully detailed breakdown of how to implement this in Terraform.

7.1. ECS Cluster + Capacity Providers

```
resource "aws_ecs_cluster" "main" {
 name = "smart-spotting-cluster"
}
Capacity Providers
resource "aws_ecs_capacity_provider" "fargate_spot" {
 name = "FARGATE_SPOT"
}

resource "aws_ecs_capacity_provider" "fargate_od" {
 name = "FARGATE"
}
```

7.2. ECS Service with Mixed Capacity Strategy

```
resource "aws_ecs_service" "app" {
 name          = "smartspot-service"
 cluster       = aws_ecs_cluster.main.id
 task_definition = aws_ecs_task_definition.app.arn
 desired_count   = 2

capacity_provider_strategy {
 capacity_provider = "FARGATE_SPOT"
  weight          = 2
}

capacity_provider_strategy {
 capacity_provider = "FARGATE"
  weight          = 0
 }
}
```

This ensures:

- Primary placement on Spot (weight 2)
- On-Demand acts as fallback (weight 0 initially)

7.3. EventBridge Rule

```
resource "aws_cloudwatch_event_rule" "spot_interrupt_rule" {
 name      = "ecs-spot-interruption"
 description = "Triggers when Spot tasks are interrupted."

 event_pattern = <<EOF
{
 "source": ["aws.ecs"],
 "detail-type": ["ECS Task State Change"],
 "detail": {
  "stopCode": ["SpotInterruption"]
 }
}
EOF
}
```

7.4. Lambda for Smart Spotting Controller

Key Lambda responsibilities:

- Detect interruption
- Scale up On-Demand
- Start retry loop for Spot
- Scale down On-Demand after Spot recovery
- Logging & metrics

IAM Role must include:

ecs:UpdateService	ecs:DescribeServices	ecs:ListTasks
cloudwatch:PutMetricData	events:PutRule	events:PutTargets

Lambda Code Outline (Python)

```
def lambda_handler(event, context):

  service_name = "smartspot-service"
  cluster_name = "smart-spotting-cluster"

   ecs = boto3.client('ecs')
```

```
Step 1: detect interrupted task
detail = event['detail']
if detail.get("stopCode") != "SpotInterruption":
  return

print("Spot interruption detected")

Step 2: Scale up On-Demand
ecs.update_service(
 cluster=cluster_name,
 service=service_name,
 capacityProviderStrategy=[
   {"capacityProvider": "FARGATE_SPOT", "weight": 2},
   {"capacityProvider": "FARGATE", "weight": 1}
],
 desiredCount=3  # add one OD instance
)

Step 3: Schedule Spot retry
schedule_retry_job()
(Where schedule_retry_job is implemented using Step Functions or
CloudWatch scheduler.)
```

8. Reliability, Load Balancing & Health Checks

To ensure seamless failover:

8.1. ALB Target Group Config

- Health check grace period: 30 seconds
- Deregistration delay: 60 seconds
- Healthy threshold: **2**
- Unhealthy threshold: **2**

8.2. ECS Service Deployment Config

minimumHealthyPercent = 50

maximumPercent = 200

This allows:

- Launching On-Demand tasks without causing outages
- Maintaining traffic flow even during Spot churn

9. Monitoring & Observability

Smart Spotting includes visibility features:

CloudWatch Metrics	Dashboards	Alerts
• SpotInterruptionCount • OnDemandFallbackCount • SpotRestorationTime • CostSavingsEstimate	• Live container distribution: Spot vs On-Demand • Interruption frequency • Cost optimization chart	• Excessive failover (Spot unstable) • Spot not available for > N minutes • Lambda exceptions • ECS service running < N tasks

10. Security Considerations

- IAM least privilege roles for Lambda
- Logging all orchestration actions
- Encryption: CloudWatch logs, Lambda env vars, ECS secrets
- Restrict EventBridge and SNS access tightly
- Use VPC-connected Lambda if workloads are private

11. Cost Optimization Impact

Smart Spotting typically produces:

- 50–70% cost reduction for compute workloads
- Guaranteed workload continuity
- Reduced manual operations
- Predictable performance + full automation

12. Failure Scenarios & Recovery Strategies

Failure Scenario	Smart Spotting Response
Spot unavailable for long duration	System remains on On-Demand until recovery
Lambda failure	EventBridge DLQ + Restore Controller Lambda
ECS deployment failure	Automatic rollback
Scaling error	Lambda retries with exponential backoff

13. Roadmap Enhancements

Future extensions:

- AI-based prediction for Spot interruptions (machine learning using historical logs)
- Integration with Kadel Labs' Agentic CloudOps agents
- Support for multi-region Spot balancing
- Dynamic cost-based capacity strategy adjustments
- Integration with Datadog, Prometheus, OpenTelemetry

14. Conclusion

Smart Spotting solves a critical cloud infrastructure challenge: achieving high availability while maximizing Spot cost savings.

It is:

- Fully automated
- IaC-driven
- Secure and scalable
- Designed for production workloads
- Created for enterprises seeking operational and financial efficiency

This white paper provides everything necessary to implement Smart Spotting in any AWS environment using ECS Fargate.

15. Contact Us

Email: contact@kadellabs.com

Phone: +91-9116593100

Web: www.kadellabs.com