MASTERING FABRIC CAPACITIES

Advanced Strategies for Planning, Monitoring, and Scaling

Benni De Jagere

Slides

Microsoft

Benni De Jagere

Principal Program Manager | Azure Data Customer Advisory Team (Azure Data CAT)

Azure Data CAT

dataMinds .be Member

/bennidejagere

sessionize /bennidejagere

/bennidejagere



Premium sponsors

VENZO_

INSPARI

valantic companies

itm8®



Eddytor™

dataon

Power your data

Standard sponsors

Backstage

Columbus



twoday

Raffle Prizes

EUROPEAN MICROSOFT FABRIC Community Conference





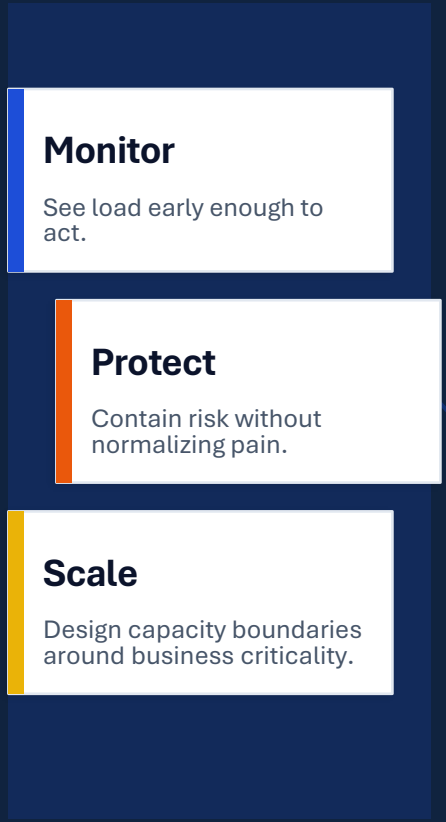




Fabric Capacities are an operating model, not just a procurement decision

As adoption grows, capacity health becomes a service-management discipline across performance, cost, and trust.

- Capacity health shapes user trust and platform adoption.
- Admin controls must balance reactive protection with architectural follow-through.
- The best outcome is fewer heroics and better repeatable operating discipline.



SESSION OUTCOMES

Let’s focus on a playbook you can run today.

We move from signal detection, to incident response, to long-range capacity architecture and governance.

1. **Monitor**
2. **Diagnose**
3. **Protect**
4. **Plan**
5. **Scale**

Monitoring

Use Metrics app and Real-Time Hub signals to spot pressure while it is still explainable.

Reactive actions

Decide when to optimize, scale up, scale out, or use emergency protection safely.

Proactive design

Separate workloads, tier service levels, and govern growth intentionally.

CORE BEHAVIOR

Interactive and background operations behave differently, each needing their own strategies

Both consume the same capacity, but they create risk in different ways and demand different admin responses.

Interactive operations

User-facing

- Report queries, Direct Lake access, and SQL exploration behave like latency-sensitive demand.
- Problems surface as delays, rejections, and degraded user confidence.
- Protection focus: preserve interactivity and reduce concurrency collisions.

Background operations

Smoothed

- Refreshes, pipelines, Spark, AI jobs, and many data movement operations build longer-lived pressure.
- Pain often arrives later through throttling thresholds and carry-forward debt.
- Protection focus: shape schedule, isolate heavy jobs, and tune thresholds thoughtfully.

Shared Fabric capacities

Same CU pool

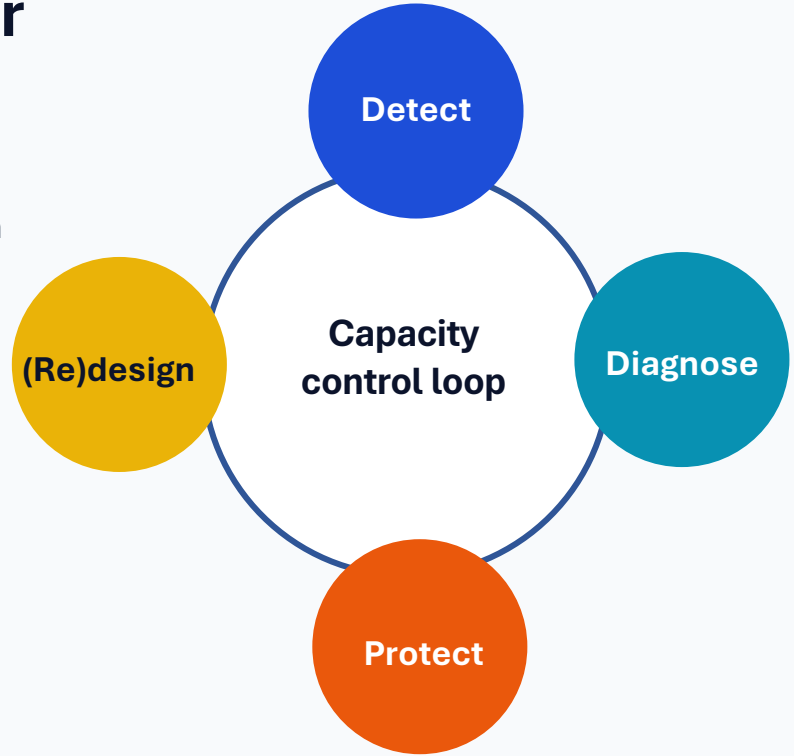
https://learn.microsoft.com/en-us/fabric/enterprise/fabric-operations

OPERATING MODEL

Capacity admins should have four control loops: detect, diagnose, protect, and redesign

The platform stays healthy when short-term response and long-term architecture stay connected.

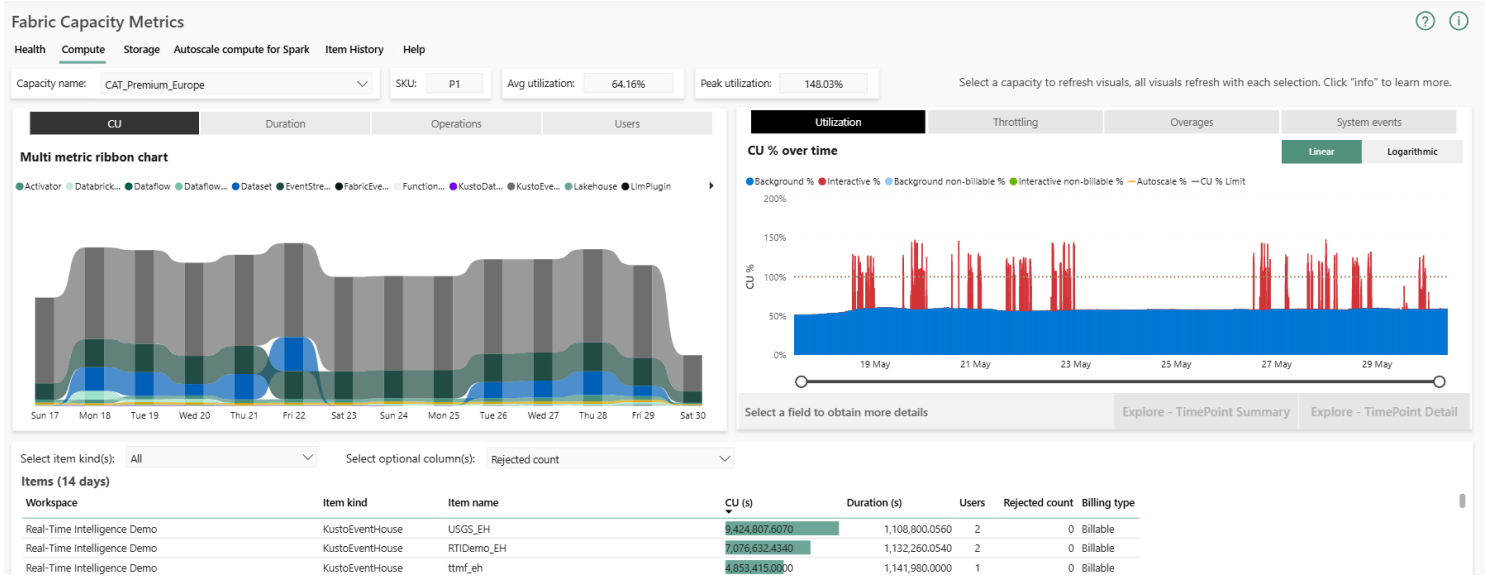
- Detect with thresholds, trend review, and real-time events.
- Diagnose by narrowing to item, operation type, and timepoint.
- Protect with surge protection, isolation patterns, and emergency capacity actions.
- Redesign using tiering, workload separation, and SKU right-sizing.



DEFAULT DASHBOARD

The Capacity Metrics app should be your default admin dashboard

A healthy operating rhythm starts with one place to review utilization, top consumers, throttling signals, and system events.



https://learn.microsoft.com/en-us/fabric/enterprise/metrics-app

FIRST INVESTIGATION STEP

Start every incident review with the 14-day Compute view

The broad view keeps admins from optimizing the wrong item or chasing an isolated spike without context.

Use the 14-day view to separate chronic pressure from isolated noise

Before you optimize anything, confirm whether the same workspace and item stay dominant across multiple days. Broad context prevents overreacting to one dramatic but unrepresentative hour.

1

Sort by CU

Find the workspaces and items that dominate the window before looking at individual jobs or queries.

2

Check workload type

Classify the top offenders into interactive, refresh, pipeline, Spark, or mixed behavior before picking a response.

3

Keep context columns

Bring size, failure, and consumer clues into the review so the eventual fix is a design choice, not a guess.

TIME DRILLDOWN

Drill from day to hour before blaming the wrong workload

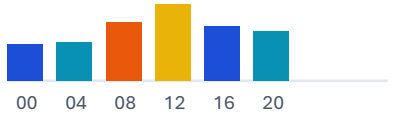
The question is not only what is expensive, but when it becomes expensive enough to matter.

14-day trend



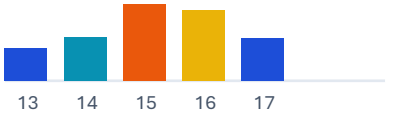
Find the peak day first

1-day focus



Spot the collision window

1-hour peak



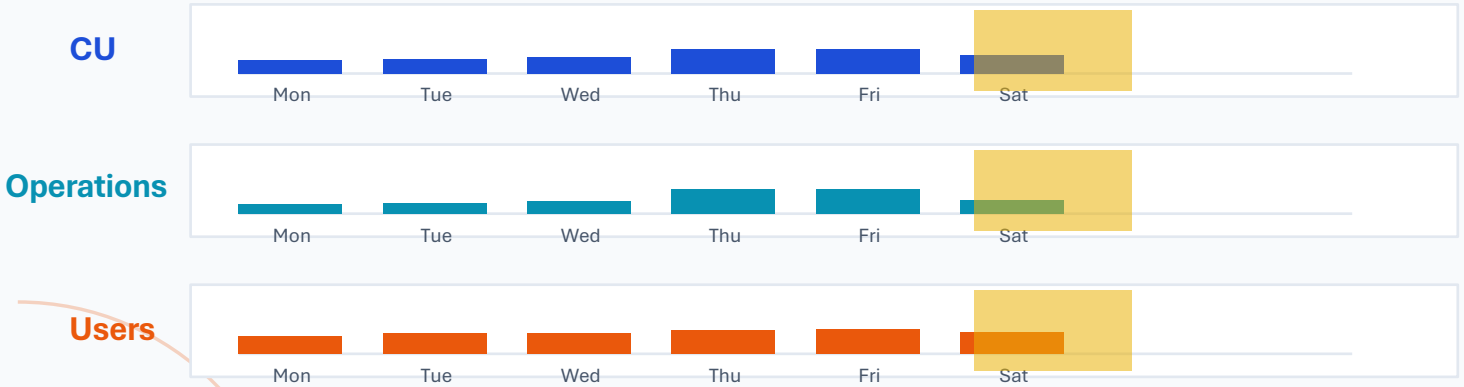
Now isolate the exact hour

Ask one follow-up question at each zoom level: did the same item stay dominant, or did a different offender emerge?

ROOT CAUSE SIGNALS

CU, Operations, and Users together tell the real story

One metric alone can mislead. The trio reveals whether you have concurrency pressure, background debt, or both.



Pattern A

CU + Users -> concurrency issue

Pattern B

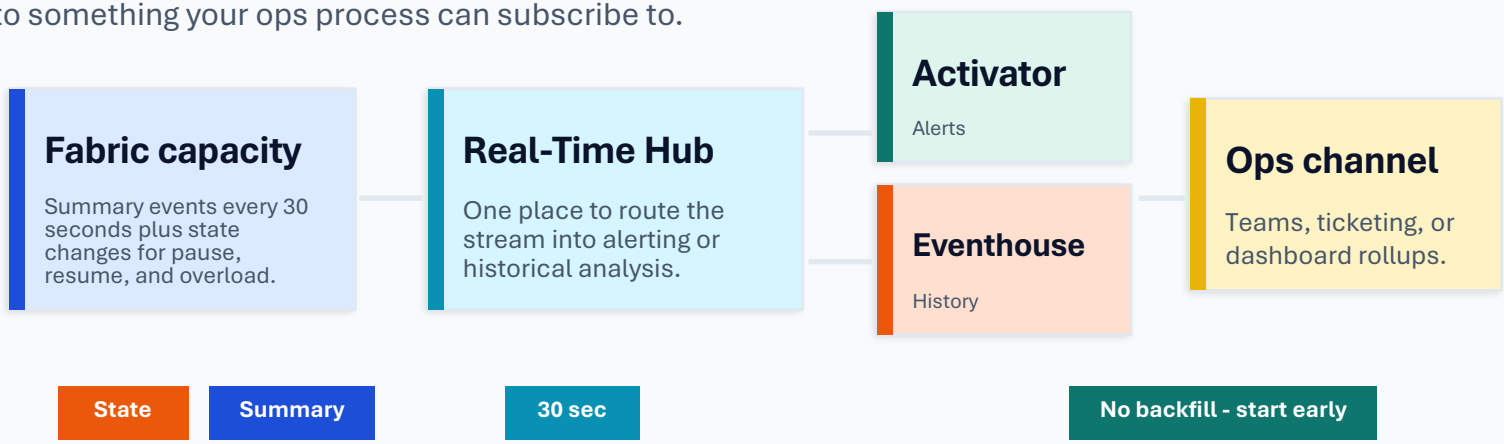
CU + low Users -> background load

REAL-TIME SIGNALS

Public Preview

Capacity events in Real-Time Hub let you move from polling to alerting

Thirty-second summary events and state-change events turn capacity health into something your ops process can subscribe to.



https://learn.microsoft.com/en-us/fabric/real-time-hub/create-streams-fabric-capacity-overview-events?context=/fabric/context/context-enterprise

REAL-TIME SIGNALS

Potential Use Cases

What **really** goes on the darkest corners of my head ..

DIY AutoScale

Use Summary signals with InteractiveDelay%

DIY *Load Balancing*

Move items, workspaces across capacities to keep spend at optimal level

Custom Chargeback

Perform Chargeback on attributes not in the current semantic model, correlate with Azure Cost Management Data

External Alerting

Integrate signals into monitoring tooling

Baseline CU

Determine a *regular use* baseline for items, alert on anomaly

CU Transparency

Provide job level CU transparency to developers, analysts

Seasonality Analysis

Retain history for longer period of time, plan for seasonal patterns

Growth Analysis

Similar items used *x*, aid in capacity planning

OPS BASELINE

A minimum alerting stack should exist before the first real outage

The goal is not more notifications. The goal is a small set of routed signals tied to named owners and expected actions.

Thresholds

High utilization, interactive delay, rejection, and pause/resume state changes.

Routing

Teams channel, on-call list, and exec-facing escalation path for tier 1 capacities.

Runbook

What gets checked first, what gets paused, and when a resize or workspace move is allowed.

Owner

Named admin and business contact for every critical capacity.

Small, specific, and actionable

FIRST DECISION

The right first question is optimize, scale up, or scale out?

Treat incident response like a decision tree. Otherwise teams keep paying twice: once in performance and again in cost.

What removes the most risk with the least collateral damage?

1 Optimize

- One item, operation is clearly dominant.
- A schedule, design defect is visible.
- You need a permanent fix, not just headroom.

2 Scale up

- The whole capacity is growing together.
- You need more headroom quickly.
- Isolation is not the real problem.

3 Scale out

- Noisy-neighbor risk returns
- Workloads need different policies or admins.
- SLA boundaries matter more than raw size.

Pausing?

BEST RELIEF

The best relief usually comes from optimizing the top offender(s)

If one workload dominates CU consumption, the best capacity action is usually a design action.

Semantic model and report

- Trim complex DAX and excess visuals.
- Use star schema and incremental refresh.
- Tune high-cardinality report pages.

Warehouse and SQL

- Reduce heavy joins and data type bloat.
- Refresh statistics and review query history.
- Separate expensive compute from live reporting paths.

Pipelines and dataflows

- Push query folding where possible.
- Avoid over-frequent refresh cycles.
- Reschedule consecutive jobs that collide.

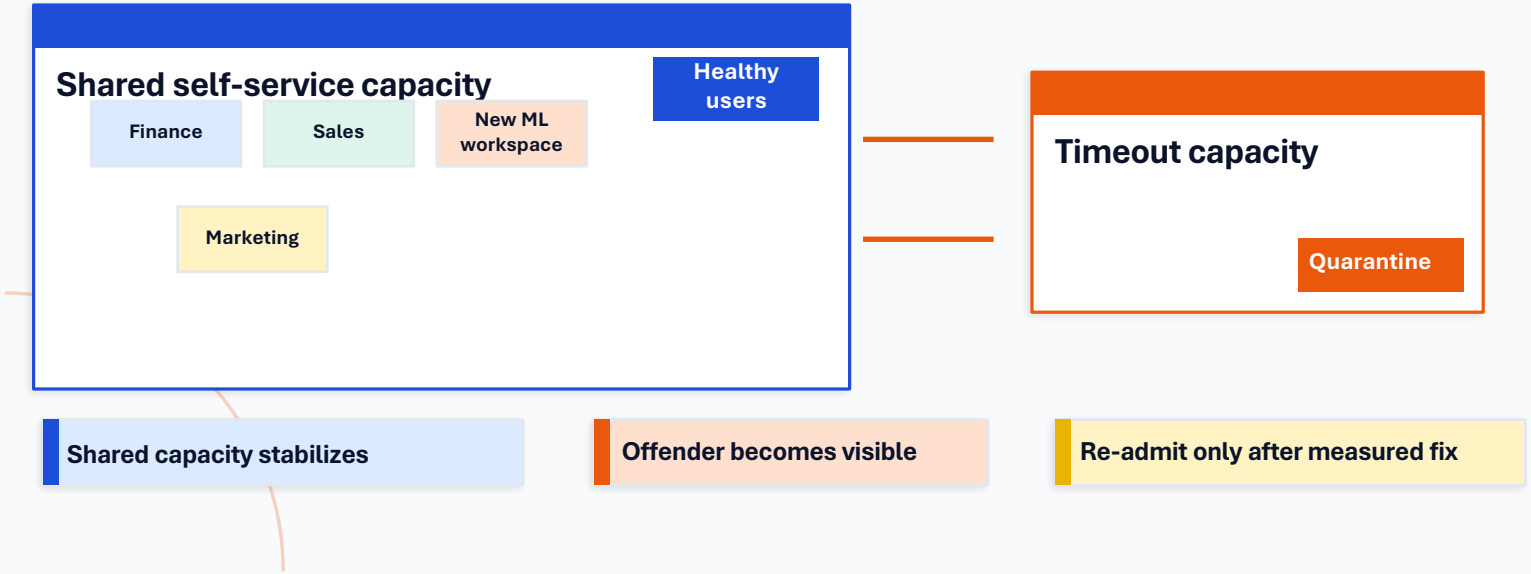
Spark and AI workloads

- Reserve only the executors you need.
- Stop idle sessions quickly.
- Treat burst factor as shared budget, not free concurrency.

ISOLATION PATTERN

Quarantining a noisy neighbor is better than punishing the whole platform

A timeout capacity protects everybody else while giving one team a controlled place to fix its workload.



STANDBY CAPACITY

A paused rescue capacity is cost effective insurance for short-lived business spikes

This is the admin pattern for legitimate but temporary peaks such as quarter close, board review, or campaign launch analytics.

Rescue capacity is a standby lane for legitimate short-lived peaks

Use it for close, campaign, or board-cycle spikes that are important but not constant. The capacity stays paused most of the time, then returns to paused once the event window ends.

1 Detect the peak

Know the date and business owner before the spike begins.

2 Resume and assign

Move the urgent workspace only for the event window.

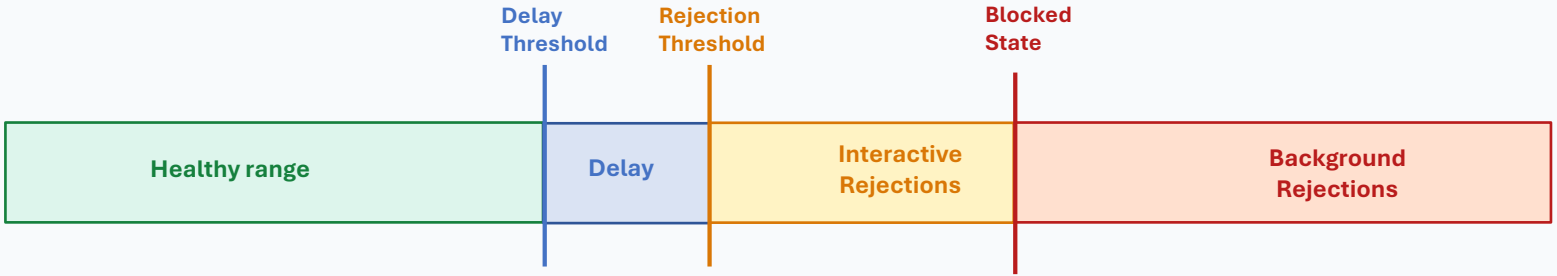
3 Return and pause

Move it back and pause again so emergency capacity stays exceptional.

PROTECTIVE THRESHOLDS

Surge protection protects interactive users, but it is a tradeoff

It brings background rejection forward to preserve responsiveness, but it does not replace correct sizing or isolation.



Use metrics to set the initial threshold
Read background rejection, interactive rejection, and utilization together.

Expect rejected background jobs
Protection improves interactivity at the cost of background completion.

Still isolate true tier-1 workloads
Surge protection does not replace dedicated capacity for critical paths.

https://learn.microsoft.com/en-us/fabric/enterprise/surge-protection#capacity-level-surge-protection-thresholds
https://learn.microsoft.com/en-us/fabric/enterprise/throttling

WORKSPACE BRAKE

Workspace-level surge protection gives you a per-workspace brake

Capacity admins can keep one workspace from monopolizing a shared estate while still exempting mission-critical paths.

Available

Default state. The workspace follows the capacity rule and can be auto-blocked if it exceeds the configured rolling 24-hour spend.

Mission Critical

Exempt from workspace consumption limits, but still not immune from overall capacity-level throttling.

Blocked

All new operations are rejected for the block period or until the admin restores availability.

Use for repeated offenders, shared capacity fairness, and temporary containment

https://learn.microsoft.com/en-us/fabric/enterprise/surge-protection#workspace-level-surge-protection

SAFETY NET

Capacity overage is a safety net, not a substitute for sizing

It prevents throttling during occasional overloads, but it does not add speed and it does not excuse chronic under-sizing.



Best use case

Rare unexpected spikes or small regular overloads where uptime matters more than the cost of a brief paydown.

Budget truth

Billed at 3x pay-as-you-go for the excess usage that avoids throttling.

Rule of thumb

If overage becomes common, resize or redesign. Do not normalize emergency spending.

https://learn.microsoft.com/en-us/fabric/enterprise/capacity-overage-overview

CARVING OUT CU STREAMS

Separate usage into different billing modes, reducing pressure on the capacity model

Spark workloads, Copilot in Fabric, and Data Agents can be separated out from the capacity model, to follow a separate billing path

AutoScale Billing for Spark

No more bursting and smoothing, only to be controlled by a single CU ceiling slider. Great for bursty workloads

Copilot Capacity

Send CUs consumed by these workloads to a dedicated capacity, to avoid the item capacity from being overwhelmed

Depending on the use cases, these can be a true lifesaver

https://learn.microsoft.com/en-us/fabric/data-engineering/autoscale-billing-for-spark-overview
https://learn.microsoft.com/en-us/fabric/enterprise/fabric-copilot-capacity

REALISTIC INCIDENT

Scenario walkthrough: Month-end finance spike exposes a capacity design gap

The point of the story is not that finance is heavy. The point is that multiple legitimate workloads collide on the same capacity boundary.

06:00

Refresh starts. Month-end semantic model and close pipeline both begin.

08:30

Executives arrive. Board and finance users hit the same model interactively.

09:00

Alerts fire. Interactive delay climbs and top CU items become obvious.

09:20

Boundary exposed. One shared capacity is serving incompatible priorities.

INCIDENT PLAYBOOK

The response sequence matters more than the heroics

The objective is a stable sequence: identify, narrow, protect, isolate or resize, then redesign the boundary so next month is quieter.

1. **Identify**
2. **Drill**
3. **Protect**
4. **Move or resize**
5. **Optimize**
6. **Redesign**

Key principle

Use protection to buy time, not to avoid the architecture conversation. The redesign decision is part of the incident response, not a separate project that may never happen.

OPERATING RUNBOOK

Pause, resume, and scheduled resizing only work when tied to a runbook

These are strong controls. Without preapproved triggers and rollback rules, they become risky improvisation.

1 Trigger

Specific threshold, planned peak, or user-facing pain.

2 Authority

Capacity admin, on-call lead, and business approver.

3 Action

Pause, resume, resize, or workspace move.

4 Rollback

When to revert and how to confirm health after the event.

Cost note: scheduled resizing and pause/resume save money only when the smoothed usage profile actually supports the smaller state after the event.

https://www.linkedin.com/pulse/fabric-billing-part-4-implications-pause-restart-matthew-farrow-fznse/

RIGHT-SIZING

Right-sizing starts with 30-second peak demand, not averages

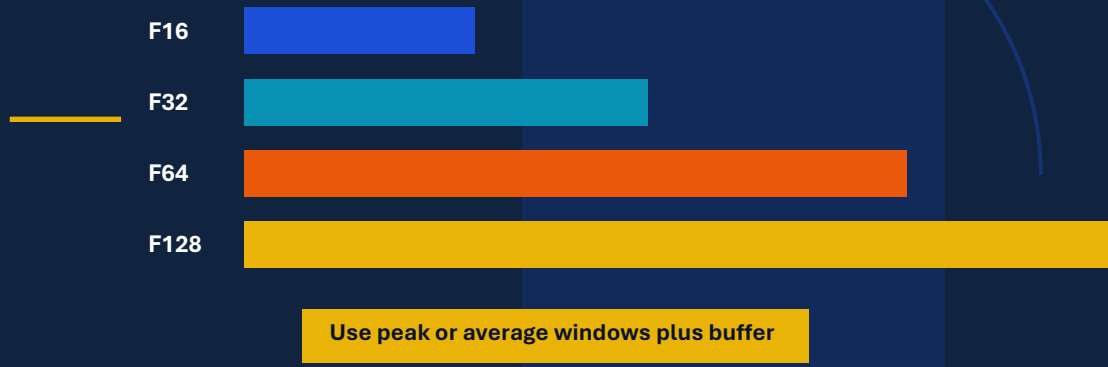
The wrong instinct is to average away the moments that actually trigger throttling. Fabric measures capacity against short windows, so planning must respect them.

Planning formula (Peak)

SKU CU budget in a 30-second window = base capacity units x 30. That is the budget your peak event must respect with headroom.

Planning formula (Average)

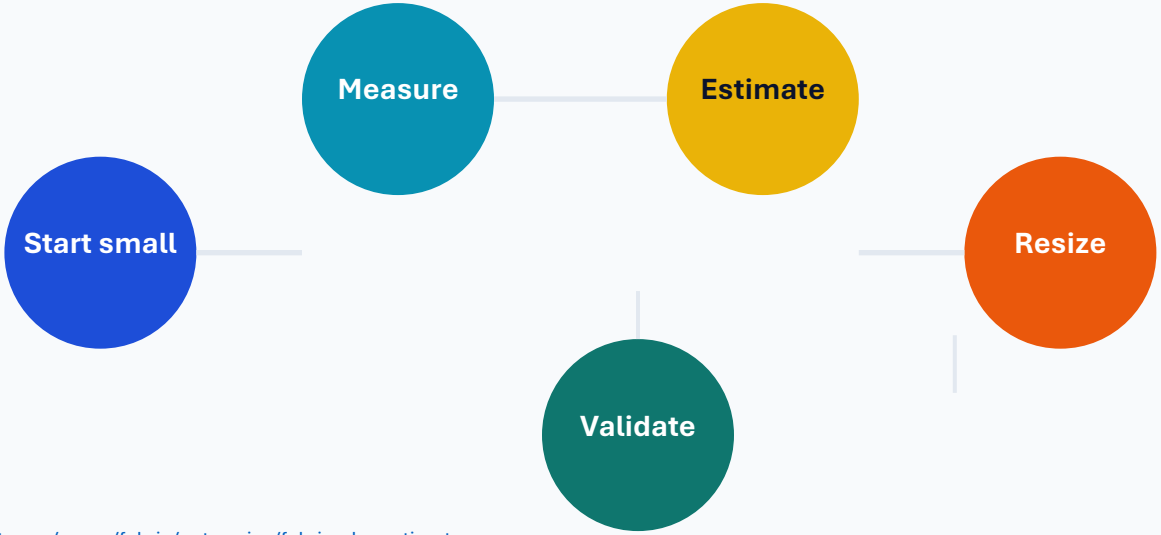
SKU CU budget in a 30-second window = base capacity units x 30. That is the budget your average usage must respect with headroom. Rely on overage protection for peaks



LEARN THE SHAPE OF DEMAND

Trial capacities and the SKU Estimator are for learning, not guessing

Start small, measure real usage, validate with the estimator, then resize deliberately instead of buying permanent headroom too early.



Estimator

Good for a first recommendation and for pressure-testing whether your real measurements feel plausible.

https://learn.microsoft.com/en-us/fabric/enterprise/fabric-sku-estimator
Fabric Tech Talks - Estimating Capacity in Microsoft Fabric
Blog – Master SKU Estimations

ALLOCATION MODELS

Shared and dedicated capacities solve different problems

Most enterprises end up hybrid, but the tradeoff only becomes clear when cost efficiency and isolation are discussed together.

Model	Best for	Strength	Risk
Dedicated	Tier-1 or domain-critical workloads	Isolation and clean accountability	Idle spend and admin overhead
Shared	Lower criticality or complementary teams	Utilization efficiency	Noisy-neighbor pressure
Hybrid	Most enterprise estates	Lets you isolate what matters	Requires explicit policy and naming discipline
Dynamic	Non-production, development workloads. Low-frequency heavy burst workloads	Control spend when not using	Requires dynamic scaling/pausing

Practical rule: share by default only when teams have compatible usage patterns and comparable service expectations.

NON-NEGOTIABLE BOUNDARY

Environment separation is cheaper than production instability

Dev, test, and prod should not compete for the same operational confidence - especially once a workload matters to leadership.

Capacity names should make environment obvious

Dev Try-out workspaces, POCs, tuning, and new item onboarding.

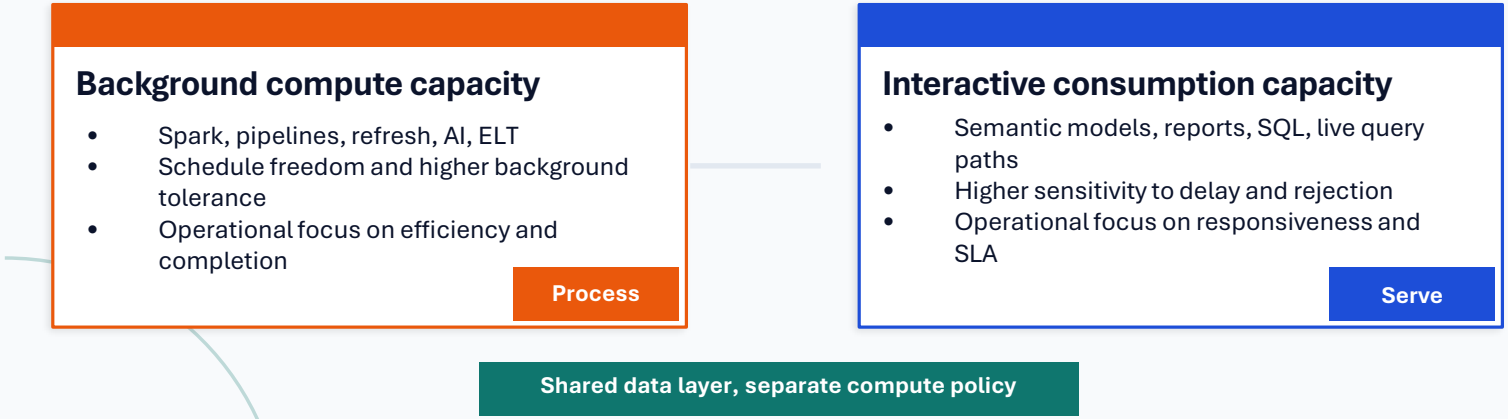
Test Validation, release rehearsal, and operational signoff before production move.

Prod Only business-owned, supportable, monitored workloads with clear SLAs.

REFERENCE ARCHITECTURE

Separate background compute from interactive consumption when SLAs matter

If the same capacity serves heavy processing and executive interactivity, you are relying on scheduling luck more than design.



SERVICE TIERS

Tier capacities by business criticality, not by org chart politics

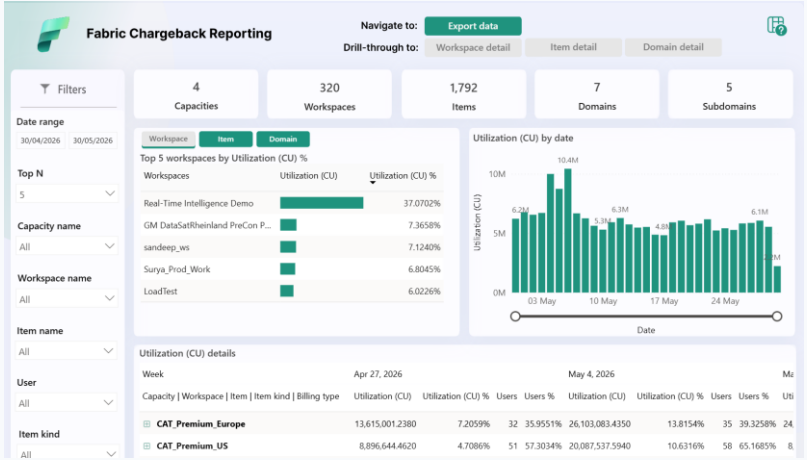
Tiering makes the cost and isolation conversation explicit. Without it, everything becomes urgent and nothing is clearly protected.

Tier 1 - mission critical	Dedicated capacity, strong headroom, named owner, alerting, and proactive scaling.
Tier 2 - business operational	Shared medium capacity with monitoring, selective isolation, and acceptable risk.
Tier 3 - ad hoc or sandbox	Cost-efficient consolidation where occasional slowness is acceptable.

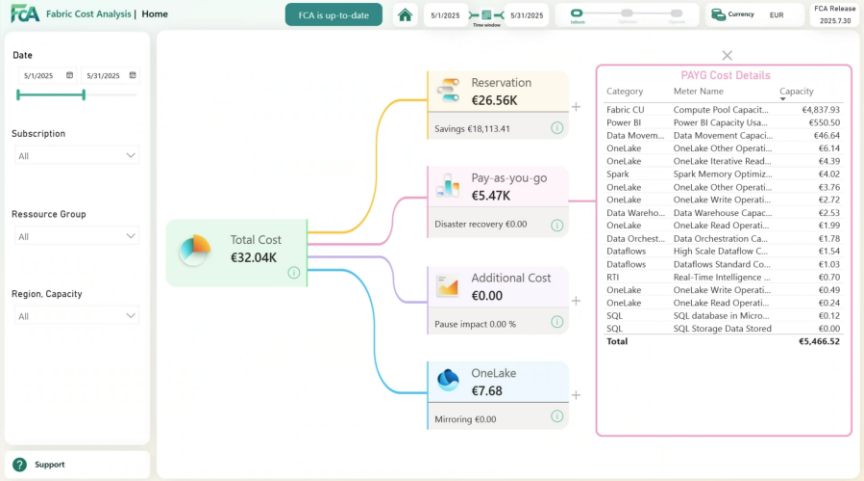
ECONOMIC SIGNALS

Chargeback and Cost Analysis can make scale decisions objective

When people can see who consumed what, more capacity requests become evidence-based conversations instead of intuition battles.



https://learn.microsoft.com/en-us/fabric/enterprise/chargeback-app



https://github.com/microsoft/fabric-toolbox/blob/main/monitoring/fabric-cost-analysis/README.md

GROWTH DECISION

Scale up for uniform growth and scale out for isolation

If you cannot explain the growth pattern, you should hesitate before buying a larger SKU just to make the alert go away.

Decide based on the pattern you can explain, not the alert you want to silence

Scale up when the whole capacity needs more headroom together. Scale out when the real problem is incompatible workload behavior, business criticality, or ownership boundaries.

Steady growth, shared behavior

Scale up. The workloads behave similarly and need more of the same headroom.

Steady growth, clear boundaries

Scale out by domain or SLA so each lane can be tuned independently.

Spiky growth or conflict

Use rescue, isolation, or overage briefly, then redesign the boundary before the next event.

OPERATING CADENCE

A quarterly capacity review prevents emergency architecture changes

Growth becomes easier to fund and easier to explain when capacity review is a standing operating motion.

Monthly

- Peak trend
- Top consumers
- Throttling count

Quarterly

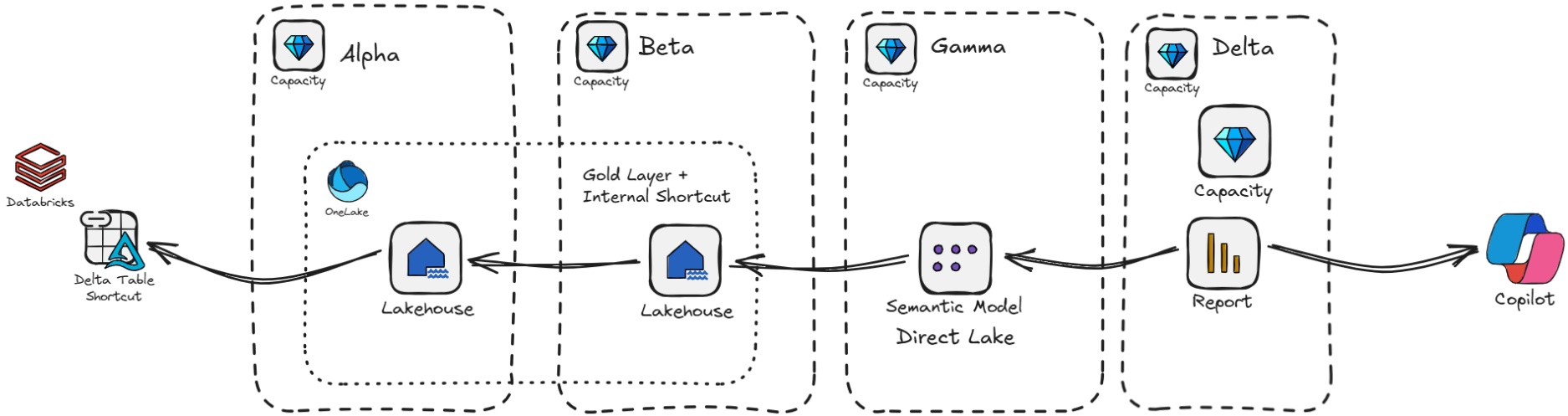
- Scale up or split candidates
- Chargeback review
- Quota headroom and runway

Annual

- Reserved vs pay-go mix
- Region and compliance changes
- Tier model refresh

Inputs: Metrics app, event history, system events, growth forecasts, and business calendar

EXERCISE: WHERE DO CU'S FLOW?



User asks Copilot (Sidepanel) to Summarize the Report,
based on a Direct Lake semantic model,
bringing together data from two different Gold Lakehouses

- Copilot usage (Copilot in Fabric, background) gets attributed to the Report (Delta)
- Semantic Model usage (Query, interactive) gets attributed to the Semantic Model (Gamma)
- OneLake usage (OneLake Read, background) gets attributed to Beta for both Lakehouses
- External Shortcut usage (OneLake, background) gets attributed to Alpha

NEXT 30 DAYS

A 30-day action plan turns the session into execution

If attendees do only five things, they should leave with fewer surprises, cleaner decisions, and better evidence for every scaling move.

1. Enable alerts and identify named owners for every critical capacity.
2. Review the top CU consumers and classify the dominant operation types.
3. Decide which workspaces need timeout or rescue-capacity patterns.
4. Tier workloads and separate the ones that should not share compute.
5. Schedule the first monthly and quarterly capacity review now.

Healthy capacity operations = fewer incidents, better trust, and evidence-based growth

Best practices for deploying your capacities

Plan your capacities

Ensure your capacities are correctly sized

Use dedicated capacities to optimize quality of experience and costs

Isolate production, development, testing in separate capacities

Budget for variability

Manage Resources

Enable Surge Protection

Monitor usage using metrics app

Adjust workload limits like pools, memory, and timeouts

Share best practices with colleagues

Optimize experiences and costs

Consider Autoscale billing for Spark

Consider Fabric Copilot Capacities

Leverage pause/resume **appropriately**

Resize capacities as needed

Move problematic content to rescue, time-out, or testing capacities

How do you prevent overloading your capacities…

Use multiple capacities and strategies to operate your capacities

Capacity A

For general purpose compute needs

Sized for typical needs

Capacity B

For Self-Service Reporting workloads

Capacity C

For large periodic workloads

Paused when not needed

Capacity D

For testing newly built content

Small size to avoid large costs

Surge protection, Overage Protection

Resize

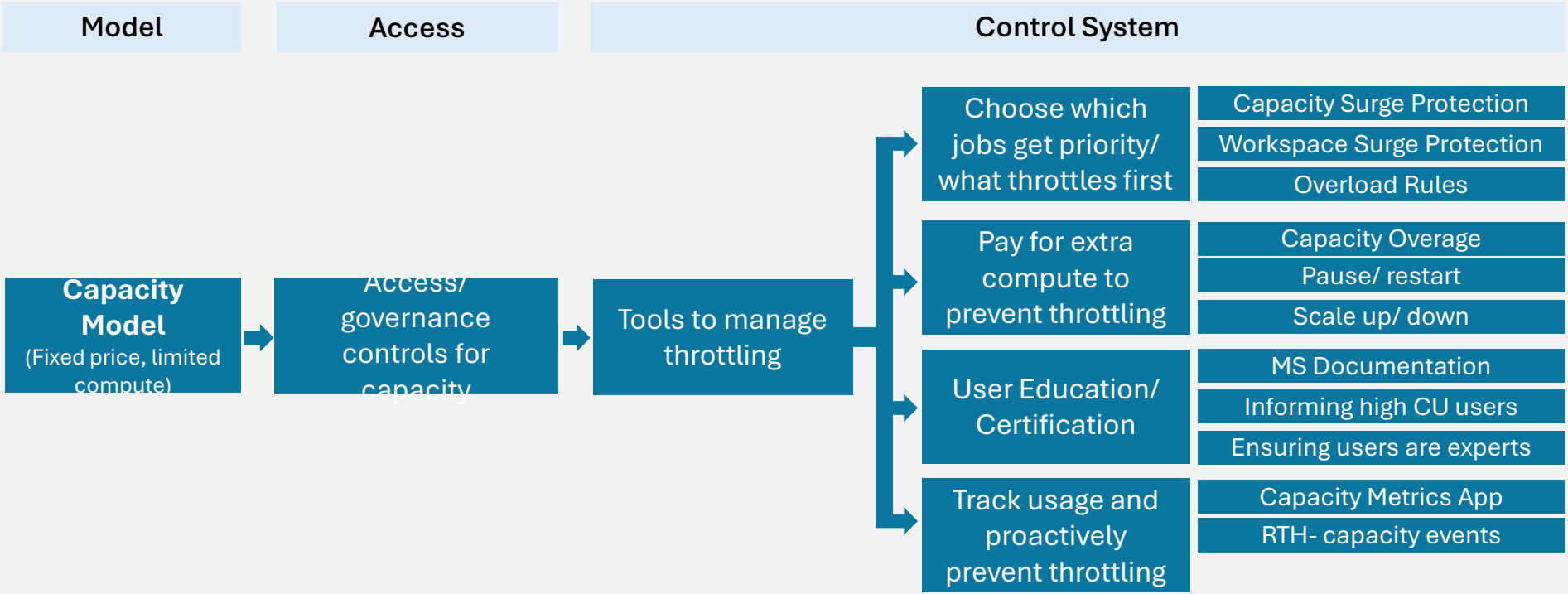
Pause and Resume

Autoscale Billing for Spark, Copilot Capacity

OneLake Shortcuts

Capacity Management Overview

Controlling how much CU is consumed and where



Slides

https://github.com/BenniDeJagere
/Presentations
/{Year}
/{Date}_{Event}

Feedback

Mastering Fabric Capacities: Advanced Strategies for Planning, Monitoring, and Scaling