Brian Loomis

Director Of Architecture
at Progress

Unlocking Business Insights with OpenTelemetry: Beyond Logging for Cloud-Native Applications like Chef360™

Conf42 Cloud Native

March 6 2025 • Online

CONF42

AGENDA

- Application and telemetry lifecycle
- So, what's wrong with the non-OTEL approach?
- Design of a system with OTEL and feedback
- Our example application
- A short animation
- What metrics do we care about in enterprise systems?
- Can't I just do this with logging?
- References and the sample app

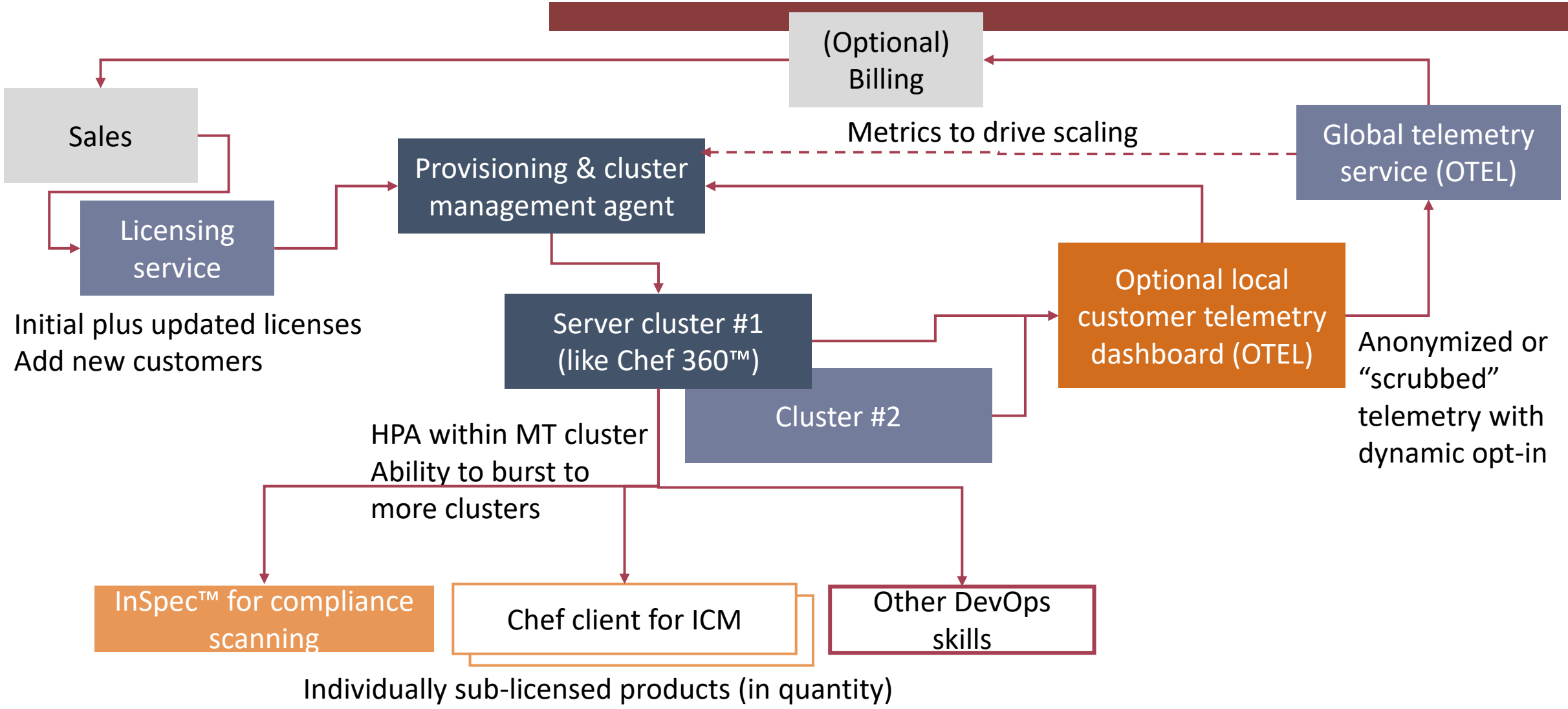
AN ENTERPRISE APPLICATION LICENSING, PROVISIONING AND TELEMETRY LIFECYCLE

- Customer buys a license with a contract in ERP for “boxed” product(s), online subscription(s) to hosted services, potentially with enumerated entitlements and limits on use
- Customer installs your application in their datacenter (possibly air-gapped), or in their cloud, or onboards to your software-as-a-service
- In each environment, the “customer” is not monolithic, nor may they be the only one
- The customer then performs operations (in Chef360™, this generates **events** like scanning a remote server, running a cookbook locally, remediating a set of nodes remotely, or running a job across a number of nodes (their infrastructure)
- They may change their original license over time, and the multi-tenant services have their own physical capacity limits
- Our previous system reported these events back to a central telemetry service (customer ID, application, license ID, event, event time) which is a simple API in front of a database with Tableau reporting

SO, WHAT'S WRONG WITH NON-OTEL?

- How many systems start with debug tracing only – effectively verbose logs (in case something goes wrong)?
  - This doesn't help product design, only helps (sometimes) in the doomsday scenario where you have to send out your developers to fix a down customer. Distributed logs are even worse – correlating different ogs in different time zones, and DataDog "diving".
- How many systems have bolted on "events" for when a feature API is called?
  - What if the logs have sensitive info in them – can your customer see their own tenant logs? (In GHE, they can!)
- How many times have we added a reporting service on top of the logs to try and calculate business metrics like "how much of the license is my customer using?" "How many of my product instances are deployed?" "How many operations per day is each customer using (engagement)? How many transactions are they doing (in-app purchases / license usage)?"
  - Wouldn't it be great if we showed that to the customer in the application? In their SaaS tenancy on the portal?
- How many times has this telemetry just been useful when the sales-person (human) calls on the customer for a renewal?
  - Customer either overbuys (awkward!) or under-buys and tries to get by (self-limiting, especially with dynamic workloads) without going back to procurement with data, driving exotic behavior
- Even in the cloud native world of autoscaling, what do the operators set the limits to? "HPA only gets you so far"

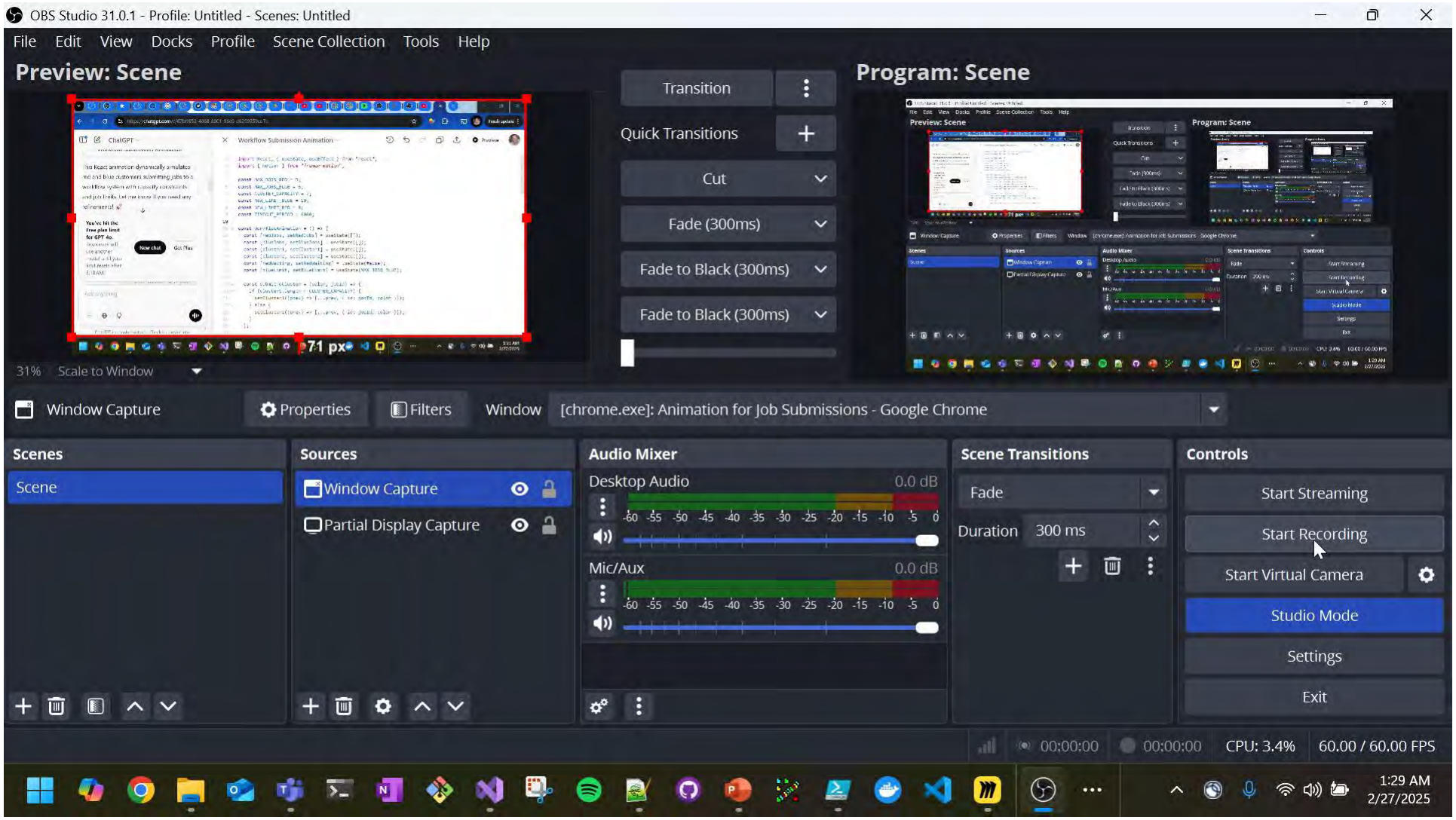
BRIEFLY, WHAT IS OUR OTEL DESIGN?



THE EXAMPLE APPLICATION

- A client tool (submits jobs periodically for multiple customers)
- (A routing service which load balances client requests to clusters with available capacity)
- A job service API which takes client requests and has limits by customer and its own capacity
- A cluster management service which provisions new clusters when an existing one is at capacity
- (An observability platform like Prometheus / Grafana where operators of multiple clusters can watch metrics by customer, cluster or management service, as well as walk through traces for specific customer debugging)

- This is a prototype with representative events
- .NET is easier to get up and running
- Our Chef 360™ code is generally GoLang in “scratch” EKS containers



SIMPLE ANIMATION SHOWING FEEDBACK OF TELEMETRY BACK TO OPERATIONS

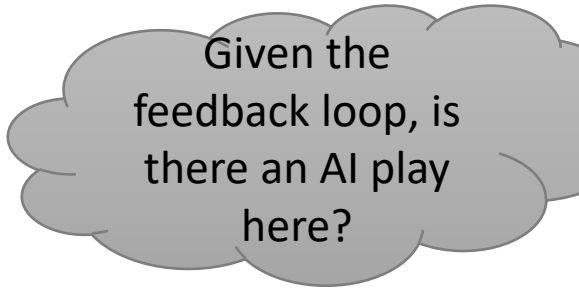
WHAT LOGGING, TRACES AND METRICS DO WE CARE ABOUT?

Legacy solution

- Usage only on a few features (events when used) with metric calculation done centrally
  - Very hard to analyze at the 10000 foot level (we lose the context for why a customer may have increased usage)
- (No remote visibility into on-prem performance)
- Metadata
  - Customer contract ID
  - Installation ID

New solution

- Usage telemetry for API accesses (**anonymized** parameters)
- Usage telemetry guided by opt-in means we **can dial it up or down** to have our engineers look at customer on-prem when necessary & customer requests
- Tracing through method call sequences within multi-tenant logs
- **Metrics** combine business operations and performance data
  - Number of client nodes enrolled (under management)
  - Number of jobs submitted / completed per hour – compare to what the customer purchased
- Kubernetes performance counters – compare to what we tested our kit to achieve
- Metadata
  - Customer contract ID
  - Tenant ID
  - Installation ID / type (SaaS, other)



CAN YOU DO THIS IN LOGGING ALONE?

- Of course, but you need to have strong conventions and possibly a single module implementing how the logs are structured.

- Many tricky details where tiered OTEL collectors would work better – for example, when the customer needs to see un-anonymized data onsite but does not want to opt in to sending that to the service provider (or wants that telemetry to be anonymized when sent, like the Microsoft Customer Experience Improvement Program ☺)
- Multitenancy makes it hard to show back the audit log to the specific customer (have to separate all the other customer data out of the logs to see who in my org performed which operations yesterday…)

HOW DID CHAINING TELEMETRY COLLECTORS AND FEEDBACK HELP?

- Local collector emits OTLP either to a file, providing more insight into actual events and performance as well in airgaps, or over-the-air to our global reporting endpoint (scaled)
  - Supports offline and online licensing, multiple clusters of services, counts of events with dynamically changing enforcement ceiling
- Our SaaS environments are **decoupled** from their individual scale maximum
  - Customer can add/subtract nodes, jobs and be provisioned across multiple clusters in AWS
  - Could we do in-app billing? Yes.
- Events allow more flexibility than just raw counts in logs since we can categorize them
  - What used to be a license for 500 nodes for the whole duration of the contract, we can not make exceptions for nodes which are ephemeral and allow some airgap reserved instances alongside online instances
  - Can we move workloads from cloud to on-prem or migrate a whole on-prem workload to a hosted solution? Not yet, but we're working on it.
- Metrics help our cloud operations team see the load on specific clusters, and feedback allows the system to eventually burst at the macro level to multiple clusters based on scaling rules (if I'm a GDPR customer, I'm authorized to go up to 3 clusters in an EU country…)

WHAT IS DIFFERENT IN THE NEW DESIGN? IN A WORD, FLEXIBILITY

Legacy solution

- Single customer deployment (no tenancy)
- Hub and spoke design for customer on-prem logging and usage information (all customers to single global service)
- Usage “telemetry” hits a custom endpoint globally, but is tightly coupled to application *and* reporting (very little room for upgrading, enrichment on the fly without redeploys)
- Only application debug info is logged (verbose) with a custom tool to gather each service’s text logs – customer has to collect and submit directly to us (no anonymization)
- Tableau as visualization (custom reports)

New solution

- Multiple tenants in online SaaS environment, in addition to customers having one or more on-premises environments
- Application usage across multiple apps, performance (Kubernetes perf counters), metering per contract
- Collecting structured logging in CloudWatch so that we can identify **traces** through individual tenants in MT environment (actionable logging for our cloud operations team)
- Visualization option for customer on-premises to see what they are sending

REFERENCES

- Open Telemetry .NET Getting Started - https://opentelemetry.io/docs/languages/net/
- OWASP logging security checklist - https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
- GitHub repository for sample application (Apache 2.0) - https://github.com/brianLoomis/otel-demo
- Chef 360™ Documentation – https://docs.chef.io/360/1.2/

THANK YOU