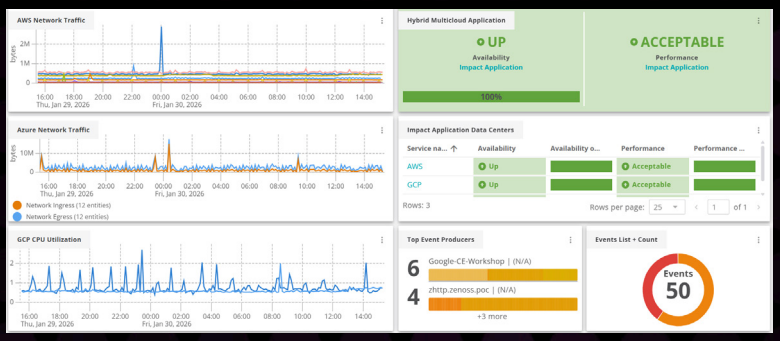
virtana | Solution Brief

Virtana Application Observability (AO)

Isolate user-impacting request degradation across services, Kubernetes, and the hybrid infrastructure those requests rely on.



Modern applications are not "just code." A single user request runs through services and APIs, Kubernetes, hosts or VMs, storage, network paths, and shared runtime capacity. When performance drops, most teams can prove a service is slow, but they still cannot explain which dependency caused it, who owns it, and what evidence will end the debate.

Virtana Application Observability connects request behavior to the orchestrator and the infrastructure that actually executes the request, so investigations do not stop at traces. APM-first tools nail spans and service timing, but stall on the questions that end incidents: where the latency started (service, Kubernetes runtime, or underlying infrastructure), what triggered it (deployments, autoscaling, noisy neighbors, storage, network changes), and who owns the fix with proof.

AO is built for hybrid reality, where constraints often live outside the service boundary and outside a single team's control.

How Virtana Application Observability Works

OBSERVE Capture distributed traces and critical user journeys with OpenTelemetry and synthetic monitoring, then correlate performance to the full delivery system: Kubernetes, hosts or VMs, bare metal, GPUs, storage, network paths, and shared services.

TRIAGE Follow the degrading request path through its dependencies to pinpoint where latency was introduced: service, Kubernetes runtime, infrastructure, or shared workload, and identify the owning team with evidence.

REMEDIATE Approve and trigger automated actions such as restarting pods, scaling resources, or rerouting traffic, keeping a human in the loop and closing the loop without switching tools.

virtana

What AO Includes

- Trace-aware topology and request flow visibility
- Trace-path analysis and automated root cause identification
- Logs correlated to performance
- Synthetic monitoring with trace context
- Dynamic service mapping and impact modeling

Business Outcomes

Faster isolation and fewer war rooms

Stop debating "app vs platform vs infrastructure" by tying request behavior to the dependency chain.

Clear ownership and stronger escalations

Send tickets to the right team faster with evidence that holds up.

More stable releases

Validate whether deployments or runtime changes altered critical paths and operations.

Better service assurance

Prioritize by service impact, not alert volume.

Common Use Cases

- Hybrid incident triage during critical disruptions and alert storms
- Application performance diagnosis that continues beyond the service boundary
- Kubernetes shared-capacity contention and noisy-neighbor isolation
- Service health and impact-based prioritization for service owners
- Operational integration with ITSM tools such as ServiceNow

The Virtana difference

- **Traces plus delivery system context:** application, orchestrator, and infrastructure investigated as one problem space.
- **Path and operation focus:** zero in on the route and operation that is failing, not a generic list of services.
- **Hybrid depth:** map and correlate across Kubernetes, hosts or VMs, storage, and network paths so investigations do not stop early.
- **Flexible log strategy:** integrate with what you already run, without forcing a rip-and-replace.