Dynatrace Day 4: Advanced Monitoring & Integration – Detailed Notes

# Network Monitoring

**Overview: Network Monitoring in Dynatrace**

**1. Deep, Process-Level Network Visibility**

Dynatrace goes beyond traditional host-based metrics by monitoring network behavior at the **process level**, offering details on how different application components communicate across hosts and environments.

**2. Performance & Health Metrics at a Glance**

You get a consolidated view combining:

* Data throughput (traffic in/out)
* Connection quality (dropped packets, errors)
* Retransmission rates (critical thresholds: <0.5% for LAN, <2% for cloud; >3% impacts UX significantly)   
  These are integrated alongside CPU, memory, and disk metrics for full-stack context

**3. Dynamic Network Topology & Smartscape Mapping**

Dynatrace’s topology view (similar to Smartscape) visually maps:

* Hosts, processes, and their network connections
* Any changes in virtualized infrastructure including VMs, cloud instances, and network interfaces

**4. Connectivity & Retransmission Insights**

You can inspect:

* **Connectivity**: Ratio of successful TCP connections vs refused/timeouts
* **Retransmissions**: Indicates network reliability issues impacting user experience

**5. Minimal Overhead with Intelligent Throttling**

Monitoring is enabled by default with low resource impact. If network packet analysis exceeds 5% CPU, Dynatrace automatically applies throttling cycles to optimize performance without manual intervention

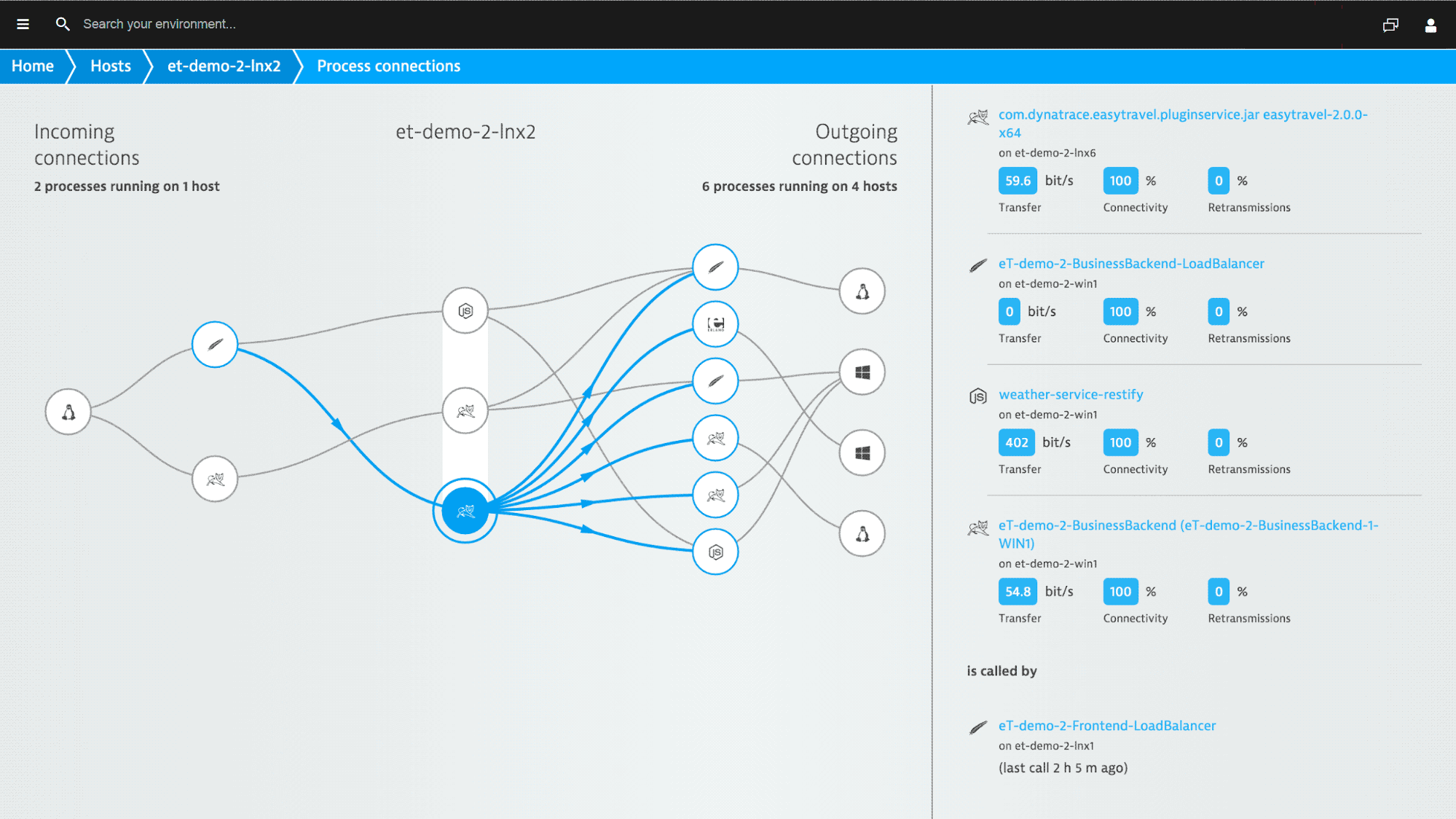
**6. Synthetic Network Availability Monitoring (NAM)**

For network-level synthetic testing, Dynatrace supports:

* **ICMP (ping)**: Verifies device reachability and network link quality
* **TCP**: Checks service port availability
* **DNS**: Ensures hostname resolution   
  These are executed from your existing private synthetic locations and integrated with Dynatrace's AI (Davis) for proactive root-cause analysis and alerting

**Why Choose Dynatrace for Network Monitoring?**

* **Unified Observability**: Consolidates network, application, infrastructure, and user experience monitoring within a single AI-powered platform
* **AI-Driven Root-Cause Analysis**: The Davis AI engine helps automatically identify the origin of network issues.
* **Scalability Across Hybrid Environments**: Supports dynamic cloud, virtual, and on-prem setups — auto-detecting new entities like VMs and interfaces.
* **Synthetic Monitoring for Non-HTTP Services**: NAM fills gaps in traditional synthetic testing — especially for non-HTTP-based infrastructure components.
* **Market Recognition & Integrations**: Named a leader in network monitoring and observability tools, with integrations with major cloud platforms and ITSM systems (e.g., ServiceNow)



**In Summary**

| **Feature** | **Benefit** |
| --- | --- |
| Process-level network visibility | Pinpoint network-heavy processes and inter-process issues |
| Integrated metrics | View network and host health metrics side by side |
| Dynamic topology mapping | Understand infrastructure dependencies at a glance |
| Synthetic NAM tests | Proactively check availability of hosts/services |
| AI-powered insights | Faster root-cause analysis and remediation |
| Scalability and extensibility | Adaptable to evolving environments and scales |

# Log Monitoring Using Dynatrace

**Log Monitoring with Dynatrace** (Grail + DQL). I’ll cover enablement, ingestion choices, processing, querying, and alerting—with copy-paste snippets and docs for each step.

**1) Enable & collect logs**

**Easiest path (OneAgent):**

* OneAgent’s log module auto-discovers app/system/container logs; configure globally in **Settings → Log Monitoring → Advanced log settings** or per-host (**Host → More (…) → Settings → Log Monitoring**). Defaults like *Detect open logs / system logs / container logs* are enabled.

**If log content isn’t visible from hosts:** enable content access on the host with the OneAgent CLI and restart the OneAgent service:

# Linux

sudo /opt/dynatrace/oneagent/agent/tools/oneagentctl --set-app-log-content-access=true

sudo systemctl restart oneagent

# Windows (run as admin, PowerShell)

& "C:\Program Files\Dynatrace\OneAgent\agent\tools\oneagentctl.exe" --set-app-log-content-access=true

Restart-Service "Dynatrace OneAgent"

(Host-level on/off is controlled by oneagentctl --set-app-log-content-access.)

**Alternative ingestion paths (when you can’t install OneAgent):**

* **Generic Log Ingestion API** (JSON/text) and **OTLP** (binary). Good for edge, serverless, or external shippers.
* **Fluent Bit** output → Dynatrace HTTP ingest (simple shipper config).

**Minimal examples**

* **cURL (JSON)**:

curl -X POST \

"$DT\_ENV\_URL/api/v2/logs/ingest" \

-H "Authorization: Api-Token $DT\_API\_TOKEN" \

-H "Content-Type: application/json" \

-d '[{"timestamp": "'$(date -u +"%Y-%m-%dT%H:%M:%SZ")'", "content":"hello from curl", "severity":"INFO", "dt.source":"curl"}]'

Endpoint/token per docs.

* **Fluent Bit (snippet)**:

[OUTPUT]

Name http

Host ${DT\_INGEST\_HOST} # e.g., abc.live.dynatrace.com

URI /api/v2/logs/ingest

Header Authorization Api-Token ${DT\_API\_TOKEN}

Format json

**2) Process logs (normalize, mask, enrich)**

Dynatrace’s **OpenPipeline** lets you define ordered processing rules for logs/events/metrics—filtering, parsing, field extraction, routing to buckets, etc. It’s the recommended path for scale and consistency.

**Mask sensitive data** at ingest with processing rules (works for OneAgent, API, and extensions). Add matcher → redact with pattern.

**3) Explore & query with DQL (Logs on Grail)**

Open **Observe & Explore → Logs** (or “Logs & Events Classic → Advanced mode”) and use **Dynatrace Query Language**. Quick patterns:

**Common queries**

-- Last 15 minutes, error logs by service

fetch logs, from: now()-15m

| filter log.level in ("ERROR","FATAL") or contains(content, "Exception")

| summarize count() by dt.entity.service.name

| sort desc

-- Parse JSON payload and extract fields

fetch logs

| parse content, "json", field:payload

| fieldsAdd userId:payload.user.id, orderTotal:payload.order.total

| summarize avg(orderTotal), count() by userId

| sort desc

-- Turn logs into metrics (rate of HTTP 5xx)

fetch logs

| filter contains(content, "HTTP/1.1\" 5")

| summarize rate = count()/duration(of:1m)

DQL supports precise time windows (from:now()-24h, to:now()-2h) and absolute timeframes.

**4) Create alerts, metrics & events from logs**

* **Log Events / Log-to-Metrics**: Use matchers/processing to emit events or derive metrics from specific log patterns (e.g., error spikes → problem).
* Feed results into **Davis AI** for problem detection/correlation alongside traces, metrics, topology.

**5) Troubleshooting checklist**

* **Agent on but no logs?** Verify **Log Monitoring** is enabled for the host/technology, and content access is on; check UI detection toggles.
* **Pipeline not applying?** Confirm your **OpenPipeline** rule order and matcher logic (rules run top-down).
* **External shipper**: test ingest with a small cURL; confirm you see dt.source or custom attributes.

**Quick start recipe (copy/paste)**

1. **Turn on** log content access (host) and confirm advanced settings.
2. **(Optional)** Add an **OpenPipeline** rule: matcher for your app logs → parse/redact → add attributes (env, team).
3. **Query** with DQL in Logs to validate fields.
4. **Create** a log event or metric from the matcher for alerting.
5. **(No OneAgent?)** Use API/OTLP or a shipper (Fluent Bit).

# Diagnostic Tools

**1. Diagnostic Tools Hub**

Dynatrace groups its core diagnostic features into one unified "Diagnostic tools" menu entry—streamlining access to essential analysis tools. These include:

* **CPU Profiler**
* **Memory Dump Analysis**
* **Process Crash Analysis**

This hub replaces the older dedicated “CPU profiler” menu for improved discoverability.

**2. CPU Profiler**

You can identify the top CPU-consuming processes and drill down to the method level (e.g., Java or .NET) to pinpoint hotspots.

**3. Memory Dump Analysis**

This tool helps capture and analyze memory dumps (e.g., from Java, .NET, Node.js), enabling root-cause analysis of memory leaks or crashes.

**4. Process Crash Analysis**

Automated detection of crashes on Windows and Linux, with support for analyzing core dumps to understand why services terminated unexpectedly.

**5. OneAgent Diagnostics**

Used to troubleshoot issues related to the Dynatrace agent itself:

* Run diagnostics at host or process level directly via the Dynatrace UI.
* Dynatrace collects diagnostic data for the past 24 hours (customizable), uploads to S3 for analysis, and provides probable solutions.
* You can also collect data manually using the oneagentctl CLI tool.

**6. ActiveGate Diagnostics**

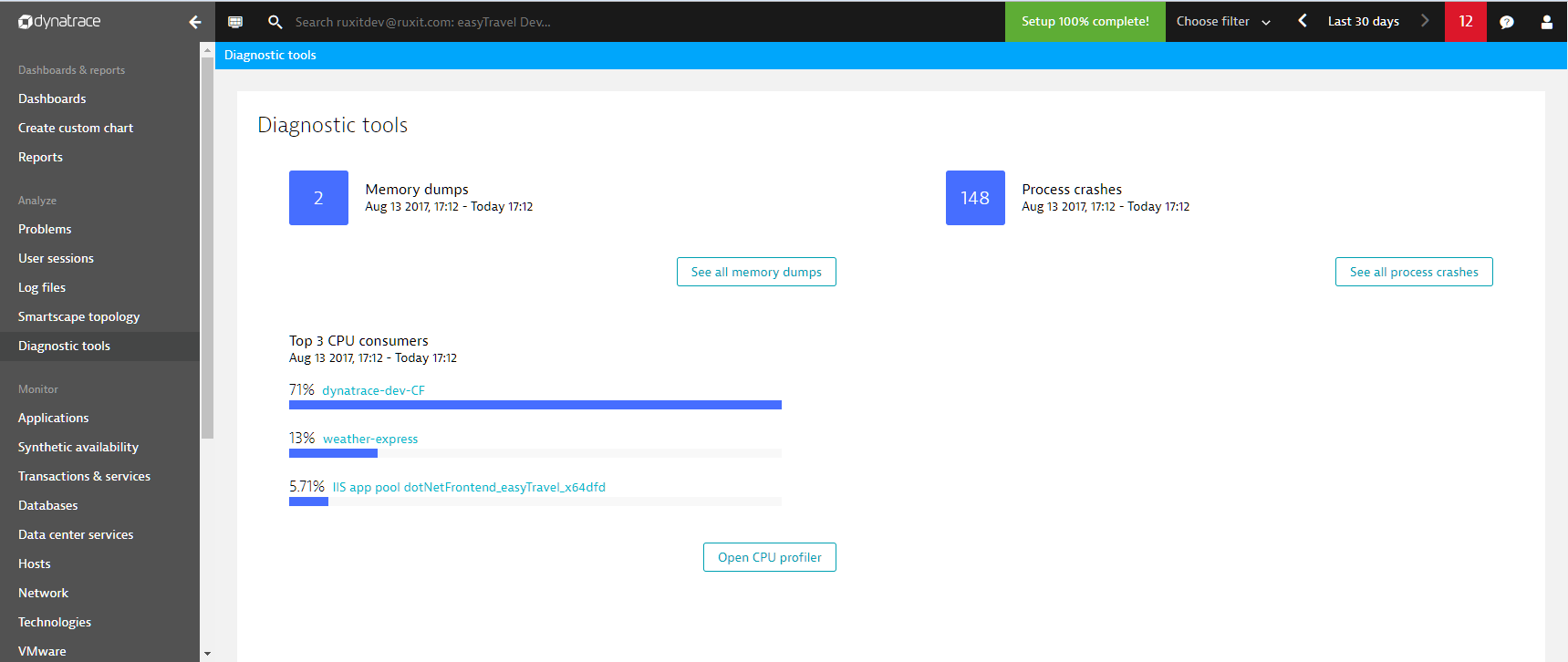
For troubleshooting issues related to ActiveGate connectivity or behavior:

* Use the UI to run diagnostics.
* By default, 7 days of operational data are collected and analyzed.
* Alternatively, use the agctl CLI to gather diagnostics locally.

**7. Developer Diagnostics (Optional Deep Dive)**

For developers focusing on performance optimization:

* Use CPU, thread, and memory profiling to detect code-level hotspots.
* There are hands-on training videos (like Dynatrace’s Tips & Tricks Episode 10) for practical guidance.



**Quick Summary Table**

| **Diagnostic Tool** | **Purpose** |
| --- | --- |
| Diagnostic Tools Hub | Centralized access to all diagnostic utilities |
| CPU Profiler | Identify high CPU usage down to method level |
| Memory Dump Analysis | Diagnose memory leaks or memory-related crashes |
| Process Crash Analysis | Analyze application crashes via core dump data |
| OneAgent Diagnostics | Troubleshoot monitoring agent issues on host/process |
| ActiveGate Diagnostics | Troubleshoot ActiveGate connectivity or malfunction |
| Developer Profiling Toolkit | Fine-tune performance using CPU/thread/memory profiling (for developers) |

# Reports & Alerts

**Alerting in Dynatrace**

**Problem Detection & Notifications**

* Dynatrace automatically detects anomalies, errors, and performance degradations, grouping related events into a **single problem** for clarity. Notifications are only sent when the problem opens or is resolved, reducing noise.
* You can push these alerts to external tools and services, including:
  + Incident management platforms (Opsgenie, PagerDuty, ServiceNow, Jira)
  + ChatOps tools (Slack, Microsoft Teams)
  + Email and webhook custom integrations

**Custom Alerting & Profiles**

* **Custom Alerts** allow you to define thresholds on any metric and trigger alerts when breached, down to the specific component.
* **Problem Alerting Profiles** give you fine control over how and when alerts are delivered. You can filter based on severity, tags, management zones, duration, or event type—ideal for targeting notifications by environment or team.

**AI-Powered Alerting with Davis AI**

* The **Davis AI engine** establishes dynamic baselines and can alert based on:
  + Real-time anomalies
  + Predicted future issues via forecasting
  + Root-cause insights using topology, transaction, and code context

**Log-Based Alerting Techniques**

Dynatrace offers three main ways to trigger alerts from log data:

1. **Log-Based Metrics**: Alert when log-derived metrics (e.g., response times, counts) exceed thresholds. Alerts are evaluated every minute.
2. **Log-Based Events**: Extract events directly from log entries for immediate alerting—ideal for sporadic or critical log entries.
3. **DQL-Based Anomaly Detection**: Use custom DQL queries for complex alert patterns. Useful but can be resource-intensive.

* Additionally, you can use **OpenPipeline** to extract Davis events from logs—triggering new problems based on log content with dynamic properties and timeouts.

**Integration & Automation Triggers**

* Beyond notifications, you can automate workflows or trigger custom code:
  + Configure alerts directly from SLOs, business processes, or metrics dashboards
  + Trigger third-party ticket creation or automation workflows
* Third‑party integrations include SIGNL4 (bi‑directional), BigPanda (via webhooks), and many more via webhooks or native integrations

**Reporting in Dynatrace**

* Dynatrace includes a **Reporting** feature that summarizes monitoring data over a set period (e.g., weekly), including an overall health or performance score.
* These reports provide a high-level view of system performance trends and can be used for leadership dashboards or retrospective reviews.

**Summary Table**

| **Feature** | **Description** |
| --- | --- |
| **Problem Detection** | AI‑powered anomaly detection with root-cause grouping |
| **Custom Alerts** | Threshold-based alerts on specific metrics/components |
| **Alerting Profiles** | Filter notifications by zone, severity, tags, duration |
| **Log-Based Alerts** | Alerts via log metrics, events, or custom DQL logic |
| **AI Forecasting Alerts** | Davis predicts future anomalies and alerts proactively |
| **Third-Party Integrations** | Connect alerts to Slack, Teams, PagerDuty, webhooks, SIGNL4, BigPanda, etc. |
| **Automation Hooks** | Trigger tickets or workflows based on alert conditions |
| **Reports** | Periodic summaries with monitoring data and overall score |