Dynatrace Day 5: Settings, AppSec, SLOs & Best Practices – Detailed Notes

# 34. Monitoring Settings

- General Monitoring Configurations:  
 - These include the global parameters and feature toggles that determine how Dynatrace captures and reports data.  
 - Settings include full-stack vs infrastructure-only monitoring, enabling/disabling log ingestion, and configuring JavaScript injection for RUM.  
 - Define default behaviors for newly added hosts or environments.  
  
- Network Zones & Data Retention:  
 - Network zones allow routing of OneAgent traffic through preferred ActiveGates, especially useful in distributed or hybrid cloud environments.  
 - Retention policies define how long data is stored (metrics, logs, sessions), balancing performance insights vs. license consumption.  
  
Example: Configure a 35-day retention for high-priority metrics and 7-day for debug-level logs.

# 35. Web & Mobile Settings

- Privacy Controls:  
 - Manage how user information is collected and stored.  
 - Control visibility of IPs, user IDs, session IDs, or specific UI fields.  
 - Toggle session replay capture globally or per application.  
  
- Masking & User Tracking Policies:  
 - Mask sensitive parameters in URLs or form fields (e.g., passwords, credit card numbers).  
 - Define sampling rules for sessions (e.g., capture 20% of mobile sessions).  
  
Example: Mask query strings for all URLs except those that contain public documentation endpoints.

# 36. Process Group Settings

- Optimizing PG Detection Rules:  
 - By default, Dynatrace uses executable path and process command-line parameters to group processes.  
 - Override rules to create logical groupings (e.g., differentiate by environment or container label).  
  
- PG Naming and Custom Metadata:  
 - Use naming templates like `TeamName-ServiceName-Environment`.  
 - Add metadata to assist with tagging, filtering, and alerting.  
  
Example: Rename all Tomcat processes to reflect owning microservice and Kubernetes namespace.

# 37. Server-side Services Settings

- Custom Service Detection:  
 - Helps isolate APIs under a common process into separate logical services.  
 - Define split by HTTP path, URL regex, or request headers.  
  
- Additional Service Monitoring:  
 - Monitor queued or background jobs (e.g., Kafka consumers, batch jobs).  
 - Customize what constitutes a request failure (e.g., HTTP 400+ or business exception).

# 38. DAVIS AI Assistant Deep Dive

- Using Davis Assistant in Workflows:  
 - Embed Davis insights into your incident workflows and escalation paths.  
 - Use Davis REST API to fetch root cause analysis programmatically.  
  
- Integrating with Chat Tools (Slack, MS Teams):  
 - Install Davis chatbots to interact in real time.  
 - Query live performance issues or trigger problem notifications.  
  
Example: Slack channel auto-notified by Davis when cart service slows down by 40%.  
  
Diagram: Davis posting a problem card to MS Teams with impact analysis and suggested root cause.

# 39. AppSec (Application Security Monitoring)

- Vulnerability Detection in Runtime:  
 - Dynatrace detects open-source library vulnerabilities by scanning actual binaries loaded at runtime.  
 - Cross-references against the Snyk vulnerability database.  
  
- Security Overview Dashboard:  
 - Shows vulnerable components by severity, exploitability, and actual usage.  
 - Aggregates exposure metrics across environments.  
  
- DevSecOps Use Cases:  
 - Alert only on used vulnerable methods, not just presence of libraries.  
 - Tag services with security compliance status (e.g., PCI-compliant).  
  
Example: Alert triggered on Log4Shell CVE (CVE-2021-44228) affecting a running backend instance.

# 40. SLOs & SLIs

- Defining Service-Level Objectives and Indicators:  
 - SLIs include metrics such as request success rate, latency percentiles (p90/p95), and availability.  
 - SLOs define targets, such as 99.9% success rate for checkout service over 30 days.  
  
- Setting Thresholds and Tracking Compliance:  
 - Display burn-down charts and daily SLO achievement percentages.  
 - Get alerted on error budget depletion.  
  
- Shift-Left Strategies for Performance:  
 - Enforce SLO checks during pre-production validations.  
 - Block releases when SLO targets aren’t met.  
  
Example: Integrate SLO compliance in Jenkins pipeline to fail deployments if error rate > 1%.

# 41. Best Practices & Wrap-up

- Governance, Monitoring Strategy:  
 - Use tagging standards and Management Zones to segregate responsibilities.  
 - Centralize control of alert routing and dashboard visibility.  
  
- Scaling Dynatrace in Large Environments:  
 - Use environment APIs to deploy monitoring as part of infrastructure automation.  
 - Deploy ActiveGates per region or VPC to ensure data locality and scale.  
  
- Training Resources and Certifications:  
 - Leverage Dynatrace University courses, webinars, and Perform certification paths.  
 - Recommended certs: Associate (intro), Professional (hands-on), and AppSec Specialist.  
  
Example: Use Terraform modules to deploy OneAgent and configure alerting policies across 200+ hosts.

This concludes Day 5, now featuring fully expanded detailed explanations, use cases, examples, and diagrams to enhance learning outcomes on Dynatrace settings, AppSec, SLOs, and scaling best practices.