Dynatrace Training - Day 1 Notes

# Dynatrace Overview & Value Proposition

What is Dynatrace?  
Dynatrace is a comprehensive full-stack observability platform that provides monitoring, analytics, and automation for applications, infrastructure, and user experiences. It is powered by artificial intelligence (Davis AI) and automation capabilities that simplify cloud complexity and optimize performance.

Capabilities Overview:  
- APM (Application Performance Monitoring): Deep code-level visibility for Java, .NET, PHP, Node.js, etc.  
- Infrastructure Monitoring: Tracks servers, VMs, containers, and cloud environments.  
- RUM (Real User Monitoring): Tracks actual user behavior, performance metrics, and interactions.  
- Synthetic Monitoring: Simulates user journeys and API availability.  
- Log Monitoring: Centralized log ingestion and analytics.  
- Davis AI: Automatic root cause detection and anomaly analysis.

Monitoring Modern Apps:  
Dynatrace excels at monitoring cloud-native architectures, microservices, Kubernetes clusters, containerized workloads, and multi-cloud environments. It ensures observability in dynamic environments using OneAgent and AI-driven insights.

# Dynatrace Architecture

Key Architectural Components:  
- OneAgent: Installed on each host; collects metrics, traces, logs, and topology.  
- ActiveGate: Acts as a communication proxy between agents and the cluster; used for DMZ, cloud integrations, and remote traffic.  
- Cluster Node (SaaS/Managed): Processes data and hosts Dynatrace UI/API.

**OneAgent in Dynatrace**

**1. Definition**

* **OneAgent** is the core monitoring component in Dynatrace.
* It is a lightweight agent installed directly on hosts (physical, virtual, cloud, or containerized).
* It automatically discovers applications, processes, services, and infrastructure components without manual configuration.

**2. Key Capabilities**

* **Auto-Discovery**: Detects processes, applications, services, databases, and dependencies in real-time.
* **Full-stack Monitoring**: Covers infrastructure (CPU, memory, disk, network), applications, processes, containers, and user experience.
* **Smartscape Mapping**: Automatically builds the service topology and dependency map.
* **Deep Code-Level Visibility**: Captures PurePath traces, method hotspots, and service flows.
* **Metrics & Logs**: Collects system metrics, application metrics, and logs for observability.
* **Security Context**: Detects vulnerabilities and suspicious code execution (with Application Security module).

**3. Installation & Operation**

* Installed as a single package → deploys appropriate modules depending on environment (Linux, Windows, Kubernetes, Cloud).
* Auto-updates (if enabled) from Dynatrace cluster or ActiveGate.
* No manual configuration needed for most common environments.
* Works with containers (injects itself into pods/containers dynamically in Kubernetes/OpenShift).

**4. Use Cases**

* Application Performance Monitoring (APM).
* Real User Monitoring (RUM) via injected JS code.
* Infrastructure Monitoring (servers, VMs, cloud instances).
* Container & Kubernetes monitoring.
* Log ingestion and correlation.
* Security vulnerability detection.

**5. Best Practices**

* Always install OneAgent on **all relevant nodes** (app servers, DB servers, container hosts, etc.).
* For Kubernetes/OpenShift → deploy via Dynatrace Operator/Helm.
* Ensure **network connectivity** to Dynatrace Cluster or ActiveGate for reporting.
* Use **auto-update** for reduced maintenance.
* Control footprint with **Monitoring Rules** (exclude irrelevant processes if needed).

**🔹 ActiveGate in Dynatrace**

**1. Definition**

* **ActiveGate** is a Dynatrace software component that acts as a **gateway/proxy** between monitored environments and the Dynatrace Cluster.
* It is not mandatory in every setup, but required in distributed, cloud, or restricted network scenarios.

**2. Types of ActiveGate**

1. **Environment ActiveGate**
   * Connects OneAgents to the Dynatrace cluster when direct communication is not possible.
   * Handles **traffic routing, data compression, and encryption**.
   * Used for **cloud integrations** (AWS, Azure, GCP, VMware, etc.).
   * Can ingest logs and metrics from external sources.
2. **Cluster ActiveGate**
   * Installed in Dynatrace Managed deployments.
   * Routes traffic between Dynatrace Cluster nodes and external components.
   * Used for multi-datacenter setups and cluster communications.

**3. Key Capabilities**

* **Traffic Relay**: Routes OneAgent traffic to Dynatrace Cluster securely.
* **API Endpoint**: Provides local endpoint for OneAgents and integrations.
* **Cloud & Third-Party Integrations**: Collects metrics from AWS CloudWatch, Azure Monitor, GCP APIs, VMware vCenter, etc.
* **Synthetic Monitoring Execution**: Runs synthetic tests (HTTP, browser) close to your apps.
* **Security**: Avoids direct internet connectivity from monitored servers → only ActiveGate needs external access.
* **Load Balancing & Scalability**: Distributes OneAgent traffic across multiple ActiveGates.

**4. Deployment Scenarios**

* **DMZ or Restricted Network**: Place ActiveGate in DMZ to route traffic from internal OneAgents.
* **Cloud Monitoring**: Use ActiveGate to fetch data from cloud APIs (without deploying agents on cloud services).
* **Kubernetes**: Works with Dynatrace Operator for cluster communication.
* **Synthetic Monitoring**: Install in required locations for transaction monitoring.

**5. Best Practices**

* Use **Environment ActiveGate** for **internet-facing communication** (agents → Dynatrace SaaS).
* Use **multiple ActiveGates** for high availability and failover.
* Place ActiveGate in **close proximity to monitored systems** to reduce latency.
* Keep ActiveGate **updated** (it auto-updates if configured).
* Secure with **firewall rules**: only allow required outbound/inbound ports.
* Monitor ActiveGate health in Dynatrace console (CPU, memory, load).

**🔹 OneAgent vs ActiveGate – Quick Comparison**

| **Feature** | **OneAgent** | **ActiveGate** |
| --- | --- | --- |
| Installation Location | On each monitored host/container | On dedicated VM/host (shared gateway) |
| Purpose | Collects monitoring data locally | Routes, aggregates, and integrates data |
| Mandatory? | Yes (for host-level monitoring) | No, only if required for integrations/routing |
| Cloud API Monitoring | ❌ No | ✅ Yes |
| Synthetic Execution | ❌ No | ✅ Yes |
| Direct Host Monitoring | ✅ Yes | ❌ No |
| Security / DMZ Setup | ❌ Not suitable | ✅ Designed for it |

**In short:**

* **OneAgent = on-host data collection & monitoring brain**
* **ActiveGate = smart proxy + integration hub + synthetic executor**

Communication Flow:  
OneAgent -> ActiveGate (optional) -> Cluster -> Dynatrace UI -> Davis AI Engine

Deployment Models:  
- SaaS: Managed by Dynatrace, hosted in the cloud.  
- Managed: Self-hosted; requires on-prem or private cloud infrastructure.

**Dynatrace Deployment Models**

Dynatrace offers **two main deployment models** for the Dynatrace Platform:

* **Dynatrace SaaS** → hosted and operated by Dynatrace in the cloud.
* **Dynatrace Managed** → deployed and operated by the customer on-premises or in their own data center/cloud.

Both models provide the **same monitoring capabilities** (OneAgent, ActiveGate, AI, Smartscape, PurePath, etc.), but differ in **where the cluster is hosted, who manages it, and how it integrates with your environment**.

**1. Dynatrace SaaS (Software-as-a-Service)**

**Overview**

* The **Dynatrace Cluster** is hosted, maintained, and upgraded by Dynatrace in their cloud (AWS, Azure, GCP).
* Customer only installs **OneAgent** and optionally **ActiveGate**.
* All data is sent securely to Dynatrace SaaS environment.

**Key Characteristics**

* **Zero cluster maintenance** → Dynatrace takes care of scaling, updates, and security.
* **Fastest onboarding** → just deploy OneAgents and optionally ActiveGates.
* **High availability** built-in → redundancy across cloud regions.
* **Frequent updates** → always on the latest version.
* **Data location** → stored in Dynatrace’s cloud (in regions chosen by customer).

**Best Suited For**

* Organizations that prefer **minimal operational overhead**.
* Cloud-first strategies or hybrid environments.
* Teams that want to **start quickly** without managing infrastructure.
* When data sovereignty/compliance allows data to leave the organization’s premises.

**Limitations**

* **Data residency concerns** → Some industries (finance, defense, healthcare) may not allow data outside their premises.
* Less control over **upgrade timing** (Dynatrace manages schedule).
* Requires **internet connectivity** for OneAgent/ActiveGate to reach Dynatrace SaaS cluster.

**2. Dynatrace Managed**

**Overview**

* Dynatrace Cluster runs **on-premises or in customer’s private/public cloud**.
* Customer is responsible for installing, operating, scaling, and upgrading the cluster nodes.
* Provides the same monitoring features as SaaS, but with **local control**.

**Key Characteristics**

* **Full data ownership** → data never leaves your environment.
* **Cluster control** → you choose when to update/upgrade.
* **Scalability** → customer must ensure enough resources for growth.
* **Integration flexibility** → suited for air-gapped networks or compliance-heavy industries.
* **Cluster ActiveGate** required for node communication and integration.

**Best Suited For**

* Large enterprises with **strict compliance/data sovereignty** needs.
* Industries with regulations that **forbid cloud-hosted monitoring** (e.g., defense, finance, government).
* Companies with strong IT operations teams capable of **running and maintaining** clusters.
* Environments requiring **air-gapped / offline monitoring**.

**Limitations**

* **Higher operational effort** → customer must maintain servers, backups, and upgrades.
* **Hardware investment** required for cluster nodes.
* **Slower adoption** compared to SaaS (more setup needed).

**3. SaaS vs. Managed – Quick Comparison**

| **Feature** | **SaaS (Cloud Hosted)** | **Managed (Self-Hosted)** |
| --- | --- | --- |
| **Cluster Location** | Dynatrace Cloud (AWS/Azure/GCP) | Customer’s data center / private cloud |
| **Management** | Dynatrace (fully managed service) | Customer (installation, scaling, updates) |
| **Data Storage** | Dynatrace Cloud | Customer infrastructure (local) |
| **Upgrades** | Automatic, handled by Dynatrace | Customer decides when to upgrade |
| **Operational Overhead** | Very low | High (requires skilled admins) |
| **Security/Compliance** | Data leaves premises | Data stays within customer boundaries |
| **Scalability** | Handled by Dynatrace | Customer responsibility |
| **Use Cases** | Fast adoption, cloud/hybrid monitoring | Strict compliance, air-gapped, sensitive data |

**4. Role of OneAgent & ActiveGate in Each Model**

* **SaaS**
  + OneAgent → installed on monitored hosts.
  + ActiveGate (optional) → used for traffic routing, cloud API integrations, synthetic monitoring.
  + Communication → OneAgent/ActiveGate → Dynatrace SaaS Cluster.
* **Managed**
  + OneAgent → installed on monitored hosts.
  + Environment ActiveGate → for routing & cloud integrations.
  + Cluster ActiveGate → required for cluster node communication.
  + Communication → OneAgent/ActiveGate → Managed Cluster (in customer infra).

**5. Choosing Between SaaS & Managed**

* Choose **SaaS** if:
  + You want **simplicity, fast adoption, and low ops overhead**.
  + Regulatory rules allow **data to leave premises**.
  + You prefer **always latest updates** without admin effort.
* Choose **Managed** if:
  + You have **strict compliance/data residency requirements**.
  + Your environment is **air-gapped or restricted from internet**.
  + You want **full control** over updates, storage, and scaling.
  + You have the IT capability to manage the infrastructure.

✅ **In short**:

* **SaaS = simplicity & speed, but less control**.
* **Managed = control & compliance, but more responsibility**.

**Dynatrace Cluster Architecture (High-Level)**

The **Dynatrace Cluster** is the central brain of the platform. Whether you deploy **SaaS** (Dynatrace-hosted) or **Managed** (self-hosted), the cluster provides:

* **Data processing and storage** (metrics, traces, logs, events).
* **AI-powered analytics** (Davis AI for problem detection & root cause).
* **Smartscape & PurePath processing** (topology + end-to-end tracing).
* **APIs and UI access** (user interaction, dashboards, management).

At a high level, the cluster is composed of:

**1. Cluster Nodes**

* Each node is a server (VM/physical) that runs the **Dynatrace cluster software**.
* Nodes handle **ingestion, storage, processing, and querying**.
* Can run as a **single-node** (PoC, dev) or **multi-node cluster** (prod, enterprise).

**2. Data Storage Layers**

* **Cassandra/Elasticsearch-like distributed stores** (under the hood).
* Store **timeseries metrics, logs, traces, events, topology metadata**.
* Replication across nodes ensures **redundancy and fault tolerance**.

**3. Communication**

* OneAgents and ActiveGates send monitoring data → cluster nodes.
* Cluster nodes exchange data via **Cluster ActiveGates** (in Managed) or directly (SaaS).
* API/UI requests are distributed across nodes.

**4. High-Level Flow**

OneAgent → ActiveGate (optional) → Dynatrace Cluster Node(s) → Storage + Davis AI

**Cluster Sizing**

Cluster sizing is crucial for **performance, stability, and scalability**. It depends mainly on:

* **Number of monitored hosts** (via OneAgent).
* **Metrics per second (MPS)** ingested.
* **Number of services/processes**.
* **Synthetic monitoring load**.
* **Retention requirements** (metrics, logs, traces).

**General Guidelines:**

* **Small Deployment (PoC, <50 hosts)** → Single node, ~16–32 GB RAM, 8–16 vCPU.
* **Medium Deployment (50–500 hosts)** → 3 nodes, each 64–128 GB RAM, 16–32 vCPU.
* **Large Deployment (>500 hosts / enterprise)** → 5+ nodes, scaling per growth (hundreds of GBs RAM and many vCPUs per node).

Dynatrace provides a **Cluster Sizing Calculator** during planning (input: hosts, processes, services, retention).

**Rule of Thumb** → Always deploy **minimum 3 nodes** in production to ensure high availability.

**Cluster Availability**

High availability is built into the architecture:

* **Multi-node architecture** → at least 3 nodes, data replicated across nodes.
* **Automatic Failover** → if one node fails, others continue serving traffic.
* **Load Balancing** → ActiveGates and OneAgents distribute load across nodes.
* **Data Replication** → ensures no single point of failure.
* **Backup & Recovery** → essential for Managed deployments (SaaS is handled by Dynatrace).

**SaaS**: Availability is handled by Dynatrace (multi-region redundancy, SLAs ~99.5%).  
**Managed**: Customer is responsible for deploying cluster nodes across different datacenters or zones for resilience.

**Cluster Scaling**

Scaling is **horizontal (add nodes)** and **vertical (increase node resources)**.

**1. Horizontal Scaling (Preferred)**

* Add new cluster nodes to distribute processing & storage load.
* Load automatically balanced across nodes.
* Supports **multi-datacenter setups** in Managed deployments.

**2. Vertical Scaling**

* Increase CPU/RAM on existing nodes.
* Used when adding more nodes is not feasible.
* Limited compared to horizontal scaling.

**3. Elastic Scaling**

* In **SaaS**, scaling is fully managed by Dynatrace → nodes are scaled in/out automatically based on load.
* In **Managed**, customer must **monitor node health** (CPU, memory, disk usage) and add capacity proactively.

**Best Practices for Cluster Sizing, Availability & Scaling**

1. **Start with 3+ nodes** in production (never single-node for prod).
2. **Distribute nodes** across **different racks/zones/datacenters** for redundancy.
3. **Use ActiveGates** to buffer traffic and protect clusters from direct exposure.
4. **Monitor node load** regularly (Dynatrace UI → Cluster Management Console).
5. **Plan for growth**: overestimate sizing by ~20–30% for future scalability.
6. **For Managed**: Implement **backup & restore strategy** for cluster storage.
7. **Enable auto-updates** (or scheduled updates) to keep nodes healthy.
8. **For SaaS**: choose region carefully (close to monitored systems, compliance needs).

**In summary**:

* Dynatrace Cluster = multi-node distributed system for data ingestion, processing, and storage.
* **Sizing** depends on monitored hosts, services, and metrics volume.
* **Availability** requires at least 3 nodes + replication across datacenters/zones.
* **Scaling** is best done horizontally (adding nodes), SaaS handles this automatically, while Managed requires manual scaling.

# 3. Dynatrace Deployment on Azure

Azure-specific Considerations:  
- Enable Azure Monitor integration for resource metrics.  
- Use Network Zones to segregate monitoring domains.

Integration with Azure Monitor and AKS:  
- Connect Azure subscriptions via Dynatrace Hub.  
- Enable AKS integration using Helm and DaemonSets.  
- Collect container, node, and workload metrics from AKS.

Cloud-native Monitoring Practices:  
- Use tagging to auto-detect resource groups.  
- Enable auto-discovery for scalable, ephemeral environments.

**Cluster Management Console (CMC) – Overview**

The **Cluster Management Console (CMC)** is the **central administration interface** for Dynatrace **Managed** deployments.

* Available at: https://<cluster-node-address>:8021 (default port).
* Provides a **web-based UI** to administer and monitor the Dynatrace cluster.
* Not applicable for **SaaS**, since cluster operations are managed by Dynatrace itself.

**Purpose:**

* Manage cluster nodes, licensing, security, and global settings.
* Monitor cluster health, resource utilization, and connectivity.
* Perform upgrades, backups, and scaling operations.

**Key Functions of CMC**

**1. Node Management**

* View and manage all cluster nodes (active and standby).
* Add or remove nodes for **horizontal scaling**.
* Assign roles to nodes (processing, storage, communication).
* Monitor **node resource usage** (CPU, RAM, disk I/O, network).
* Handle **node replication and synchronization** for HA.
* Manage **Cluster ActiveGate** connections.

**2. Licensing Management**

* View **license details**:
  + License type (host units, Digital Experience Monitoring, log monitoring, AppSec, etc.).
  + Total quota, used quota, and remaining quota.
* Assign **license quotas to environments** (e.g., Prod vs Test).
* Ensure compliance with purchased capacity.
* Monitor **license consumption trends** to plan for scaling or purchasing additional capacity.

**3. Cluster Health & Updates**

* **Health Dashboard**:
  + CPU, memory, and storage usage for each node.
  + Disk space alerts (important for logs/traces).
  + Connectivity issues between nodes.
  + Database replication status.
* **Alerts & Notifications**:
  + Node down / unreachable alerts.
  + License overuse warnings.
  + Upgrade readiness notifications.
* **Updates**:
  + CMC shows available **Dynatrace cluster version updates**.
  + Admins can **schedule upgrades** (choose maintenance windows).
  + Supports **rolling updates** → one node at a time (ensures HA).
  + Provides **release notes** and compatibility checks before update.

**Additional Features of CMC**

* **Security & Access Control**:
  + Manage administrator accounts (local or LDAP/SSO integration).
  + Configure TLS/SSL certificates for cluster communication.
* **Backup & Restore**:
  + Configure automated backups of cluster configuration & data.
  + Trigger manual backups.
* **Environment Management**:
  + Create, configure, and isolate monitoring environments (e.g., Dev, Test, Prod).
  + Assign quotas and access per environment.
* **Integration Management**:
  + Manage connections to external services (email, ServiceNow, APIs).

**Best Practices for Using CMC**

1. **Node Health Monitoring**
   * Regularly check CPU, RAM, and disk utilization.
   * Ensure replication status is “healthy” across nodes.
2. **License Tracking**
   * Monitor usage against quota to prevent service degradation.
   * Adjust quotas between environments dynamically.
3. **Update Management**
   * Always test updates in staging before production.
   * Use rolling updates for zero downtime.
   * Keep cluster close to the **latest LTS version** for stability/security.
4. **High Availability**
   * Run at least 3 cluster nodes.
   * Distribute nodes across racks/zones/datacenters.
5. **Backups & Disaster Recovery**
   * Schedule automated backups.
   * Periodically test restore procedure.

**In summary**:  
The **Cluster Management Console (CMC)** is the control plane for **Dynatrace Managed clusters**. It allows administrators to:

* Manage **nodes** (scale, monitor, replace).
* Oversee **licensing** (allocate, track usage).
* Ensure **cluster health** (alerts, replication, performance).
* Perform **updates & upgrades** in a controlled manner.

**User Management in Dynatrace Managed**

In **Dynatrace Managed**, all **user and access management** is done via the **Cluster Management Console (CMC)**. Unlike SaaS (which uses Dynatrace Account Management UI), Managed gives you **full control** over identity, roles, and federation within your own infrastructure.

**1. Access Management via CMC**

* Admins log in to the **CMC** (https://<cluster-node>:8021) to configure user and group access.
* You can **create users manually** or **integrate with enterprise identity providers** (LDAP, SAML, OIDC).
* CMC manages **cluster-wide access** (who can log in, what environments they can see, and at what role level).
* Permissions apply to **environments** (e.g., Dev, Test, Prod) and **cluster-wide settings** (e.g., licensing, node management).

**Typical workflow:**

1. Create user or connect directory (LDAP/SSO).
2. Assign roles (Cluster admin, Environment admin, Viewer, etc.).
3. Optionally group users into **groups/teams** for easier management.
4. Apply **scoped access** to specific environments.

**2. Role-Based Access Control (RBAC)**

RBAC in Dynatrace Managed ensures **principle of least privilege**.

**Cluster-Level Roles (via CMC)**

* **Cluster Administrator**
  + Full control over all cluster settings (nodes, licensing, security, environments).
  + Can create/delete environments.
* **Cluster Operator**
  + Limited to monitoring cluster health, performing updates.
  + Cannot manage licensing or delete environments.

**Environment-Level Roles (within CMC & Environment)**

Each monitoring environment has its own RBAC assignments. Common roles include:

* **Environment Admin**
  + Full access within one environment.
  + Can configure settings, dashboards, alerting, integrations.
* **Environment Editor**
  + Can create dashboards, alerts, manage monitoring configs.
  + Cannot manage environment-wide settings or user access.
* **Environment Viewer**
  + Read-only access → can view dashboards, metrics, problems, Smartscape.

**Custom Roles (Fine-grained Permissions)**

* Dynatrace allows custom roles with specific permissions (e.g., only manage dashboards, only view logs, etc.).
* Useful for **segregating duties** between Dev, Ops, SRE, Security teams.

**3. Federation / SSO Integration Options**

Dynatrace Managed supports **enterprise identity federation** for **seamless and secure access**.

**Supported Integration Methods**

1. **LDAP / Active Directory**
   * Connects to internal AD/LDAP for authentication.
   * Users/groups synced automatically.
   * Assign AD groups to Dynatrace roles for centralized control.
2. **SAML 2.0 Federation**
   * Supports major IdPs (Okta, Azure AD, Ping Identity, ADFS, etc.).
   * Enables **Single Sign-On (SSO)** with corporate credentials.
   * Allows MFA policies from the IdP.
3. **OpenID Connect (OIDC)**
   * Lightweight modern alternative to SAML.
   * Integrates with IdPs like Azure AD, Google Identity, Keycloak, etc.

**4. Best Practices in User Management**

* **Use Groups, not individuals** → Assign roles to AD/SAML groups rather than users.
* **Enforce SSO** → Disable local accounts where possible (use IdP with MFA).
* **Segregation of Duties** → Separate **Cluster Admins** (infra team) from **Environment Admins** (DevOps/SRE team).
* **Audit Regularly** → Periodically review user roles & license usage.
* **Principle of Least Privilege** → Start with minimal access (Viewer) and grant elevated roles only when needed.

**Summary**

* **Access via CMC**: Central control of users, groups, and permissions in Managed clusters.
* **RBAC**: Fine-grained role-based access at **Cluster** (admin/operator) and **Environment** (admin/editor/viewer/custom) levels.
* **Federation**: Supports **LDAP, SAML, OIDC** for enterprise SSO/MFA.
* **Best Practices**: Group-based access, enforce SSO, limit cluster admins, review roles regularly.

**1. Entity Hierarchy in Dynatrace**

Dynatrace automatically builds a **Smartscape topology** of your environment. At its core is an **entity hierarchy**, where monitoring objects are discovered and linked.

**Key Entity Types**

* **Hosts**
  + Physical servers, VMs, or cloud instances where **OneAgent** runs.
  + Provide infrastructure-level metrics: CPU, memory, disk, network.
  + Form the foundation of monitoring.
* **Processes**
  + Running executables or services on a host (e.g., tomcat, java.exe, nginx).
  + OneAgent detects process groups automatically.
  + Each process is tied to the host and higher-level services.
* **Services**
  + Logical representation of applications (detected automatically via PurePath tracing).
  + Examples: Web services, databases, APIs, microservices.
  + Built from processes → grouped into end-to-end service flows.
* **Applications** (above services)
  + Represent front-end user-facing systems (web, mobile apps).
  + Powered by RUM (Real User Monitoring).

**Hierarchy Flow**

Hosts → Processes → Services → Applications

* Hosts run processes.
* Processes form services.
* Services support applications.
* Dynatrace links all dependencies via **Smartscape**.

**2. Tagging**

**Tags** are labels you assign to entities for **organization, filtering, automation, and scoping**.

**Tag Types**

1. **Manual Tags**
   * Applied by users directly in the UI or via API.
   * Example: Tagging a host Production, Finance-Team, or Linux-DB.
2. **Auto-Tag Rules**
   * **Best practice**: Define rules so tags are applied dynamically.
   * Conditions can be based on:
     + Metadata (cloud provider, Kubernetes labels, environment variables).
     + Process/host/service properties (name, IP, OS, executable).
   * Example Rule:
     + If host name contains "prod-" → Tag = "Production".
     + If process group name contains "nginx" → Tag = "Web-Frontend".
3. **Metadata-Based Tags**
   * Dynatrace ingests metadata from cloud providers (AWS tags, Azure Resource Groups, Kubernetes labels/annotations).
   * Tags are automatically synchronized and available for filtering.

**Use Cases for Tags**

* Filtering dashboards or problem views.
* Grouping entities (e.g., “Environment: Prod/Dev/Test”).
* Auto-assigning workloads to **Management Zones**.
* Automation with APIs and alert routing.

**3. Management Zones**

**Management Zones (MZs)** are logical partitions of the environment that control:

* **Scope of visibility** → what users can see.
* **Scope of alerting & dashboards** → what data applies to.
* **Access management** → tie MZs to RBAC roles.

**How MZs Work**

* Defined by **rules & filters** (based on tags, properties, metadata).
* Example:
  + MZ “Production Apps” → includes hosts tagged Production, services tagged CustomerPortal.
  + MZ “Team A” → includes entities tagged Team=A.
* A single entity can belong to **multiple management zones** (flexible scoping).

**Use Cases**

* **Multi-tenant monitoring** → Give each team visibility into only their apps.
* **Environment separation** → Prod vs Test vs Dev.
* **Access control** → Assign RBAC roles per management zone.
* **Dashboards/Reports** → Create MZ-specific views for different teams.

**Best Practices for Organizing Your Environment**

**Entity Hierarchy**

* Rely on Smartscape for dependency mapping.
* Validate processes/services grouping (adjust process group detection if needed).

**Tagging**

* Use **auto-tagging rules** as much as possible (consistent + scalable).
* Ingest **cloud metadata** to auto-apply cloud resource context.
* Create a **tagging taxonomy** (e.g., Environment, Team, Application, Region).

**Management Zones**

* Design around **teams and responsibilities** (Team A, Team B, Infra Ops, SRE, Security).
* Use tags as the **basis for MZ rules**.
* Limit dashboards and alerts to MZ scope for clarity.
* Ensure a **Global View** MZ for admins.

**Summary**

* **Entity Hierarchy**: Hosts → Processes → Services → Applications (auto-discovered via OneAgent).
* **Tagging**: Labels for entities (manual, auto, metadata-based) → essential for filtering, organizing, and automation.
* **Management Zones**: Logical partitions to **scope access, views, and dashboards** → critical for multi-team and multi-environment setups.