What is Monitoring and Why it is needed ?

Monitoring

Continually watch the performance of an entity by evaluating the metrics collected from it. For example - website, software, infrastructure, servers, cloud services etc.

What to monitor

- Website, Network latency
- Uptime, Throughput
- Success rate, Error rate, Request rate
- CPU usage

Advantages

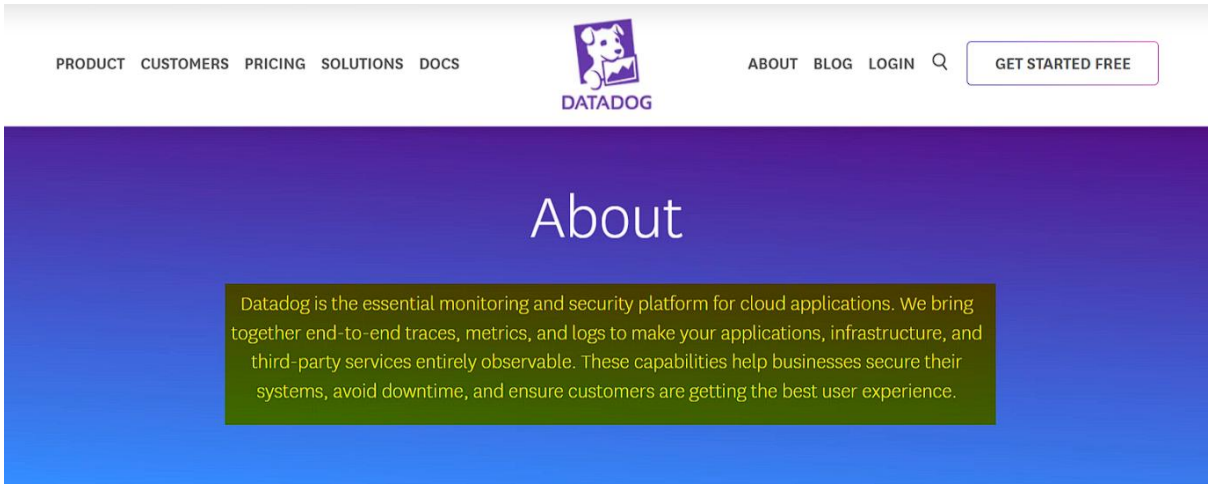
Underlying problems in an application can be detected **before they have an adverse effect.**

Reduced downtime, Higher conversion rate, Improved productivity, Ensuring SLA compliances etc

Need of it

Applications have moved from using few tools and a central infrastructure to a **wider range of tools and dispersed granular infrastructures.**

Monitoring several moving parts in a single application requires a strong monitoring solution.


PRODUCT CUSTOMERS PRICING SOLUTIONS DOCS
DATADOG
ABOUT BLOG LOGIN
GET STARTED FREE
About
Datadog is the essential monitoring and security platform for cloud applications. We bring together end-to-end traces, metrics, and logs to make your applications, infrastructure, and third-party services entirely observable. These capabilities help businesses secure their systems, avoid downtime, and ensure customers are getting the best user experience.

Here's a concise breakdown of **logs**, **metrics**, and **traces** — the three pillars of observability:

---

1. Logs

What: Timestamped records of discrete events.
Format: Unstructured (text) or structured (JSON).
Use: Debugging, audits, root-cause analysis.

Example:

2025-08-04T10:00:00Z GET /login 500 Internal Server Error

Key Characteristics:

- Tell **what** happened and **when**.
- Often very **detailed** but can be **voluminous**.
- Used after something goes wrong.

---

2. Metrics

What: Numeric measurements over time.
Format: Time-series data (e.g., Prometheus).
Use: Monitoring system health and performance.

Example:

http_requests_total{status="200"} = 3450

cpu_usage_percent = 78.2

Key Characteristics:

- Lightweight and **efficient** for aggregation.
- Good for **alerting and dashboards**.
- Do not show detailed context or errors.

---

3. Traces

What: Records of a single request's journey across services.
Format: Distributed trace (spans).
Use: Identifying bottlenecks and latency in microservices.

Example:
A user request traced across:

- API Gateway → Auth Service → Payment Service → DB

Key Characteristics:

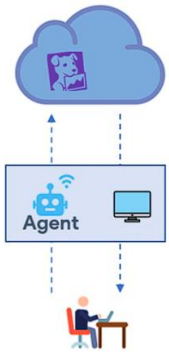
- Visualize **end-to-end flow** of a request.
- Show **duration** and **dependencies**.
- Crucial for understanding **performance issues** in distributed systems.

Summary Table

Aspect	**Logs**	**Metrics**	**Traces**
Type	Event records	Time-series numbers	Request journeys
Structure	Text (structured/unstructured)	Numeric (key/value + tags)	Hierarchical (spans)
Use	Debugging, RCA	Monitoring, alerting	Performance, latency analysis
Granularity	High	Aggregated	Per-request
Volume	High	Low to medium	Medium

DATADOG

It is an **observability service** for cloud-scale applications, **providing monitoring** of servers, databases, tools, and services, through a **SaaS-based data analytics platform.**



DATADOG

It is an **observability service** for cloud-scale applications, **providing monitoring** of servers, databases, tools, and services, through a **SaaS-based data analytics platform.**

End-to-End monitoring solution covering the collection of metrics, alerting system, troubleshooting and built-in dashboarding for visualizations of metrics.

- Infrastructure monitoring
- Database monitoring
- Cloud monitoring
- Log monitoring
- Application performance monitoring
- Real user monitoring
- Container monitoring
- Security monitoring
- Synthetic monitoring

DATADOG

Comes with **500+ built-in integrations** with pred-defined dashboard templates.

- Operating systems → Windows, Linux, and Mac.
- Cloud → AWS, Azure, Google cloud.
- Containers → Docker, Helm, Container-d
- Messaging services → Kafka, Apache active MQ, Hive MQ
- Security → Alcide, Apptrail, Okta, Hashicorp vault.

DATADOG

About DataDog Project

Was founded in **year 2010** by Olivier Pomel .

Datadog's agent code is **open-sourced on Github.**

Thousands of customers including Samsung, Shell, Sony, HashiCorp, Nikon, Deloiite and many more.

Very active developer and user community around the globe.

Metric Types

Work metrics

Indicate **the top-level health** of your system by measuring its useful output.

- **Throughput** – Ex: No. of requests or queries per second
- **Success rate** – Ex: Percentage of queries successfully executed
- **Error rate** – Ex: Percentage of queries failed executed
- **Performance** – Ex: Percentile response time

Resource metrics

Indicates **timely information of physical resource** components such as CPU, memory, disks.

- **Utilization** – Ex: Percentage time that device was busy
- **Saturation** – Ex: Wait queue length, Swap usage
- **Availability** – Ex: % of time resource responded to requests
- **Errors** – Ex: Internal errors or device errors

Basic Terminologies

Host in an entity which Datadog has to monitor. For example : Servers, VMs, Containers, IOT devices, Desktops etc.

Metric is a **time bound information** (the data value) pertaining to a system captured at a certain point in time.

Events happen at infrequent occurrences that usually provides the details of a **change that happened in the system**.

Agent is a service that runs alongside the application software system / host to **collect various events and metrics** from it and **sends it to the datadog backend** via internet

Alert a triggering event produced by a monitor when it determines a **metric value crosses a threshold.**

Architecture