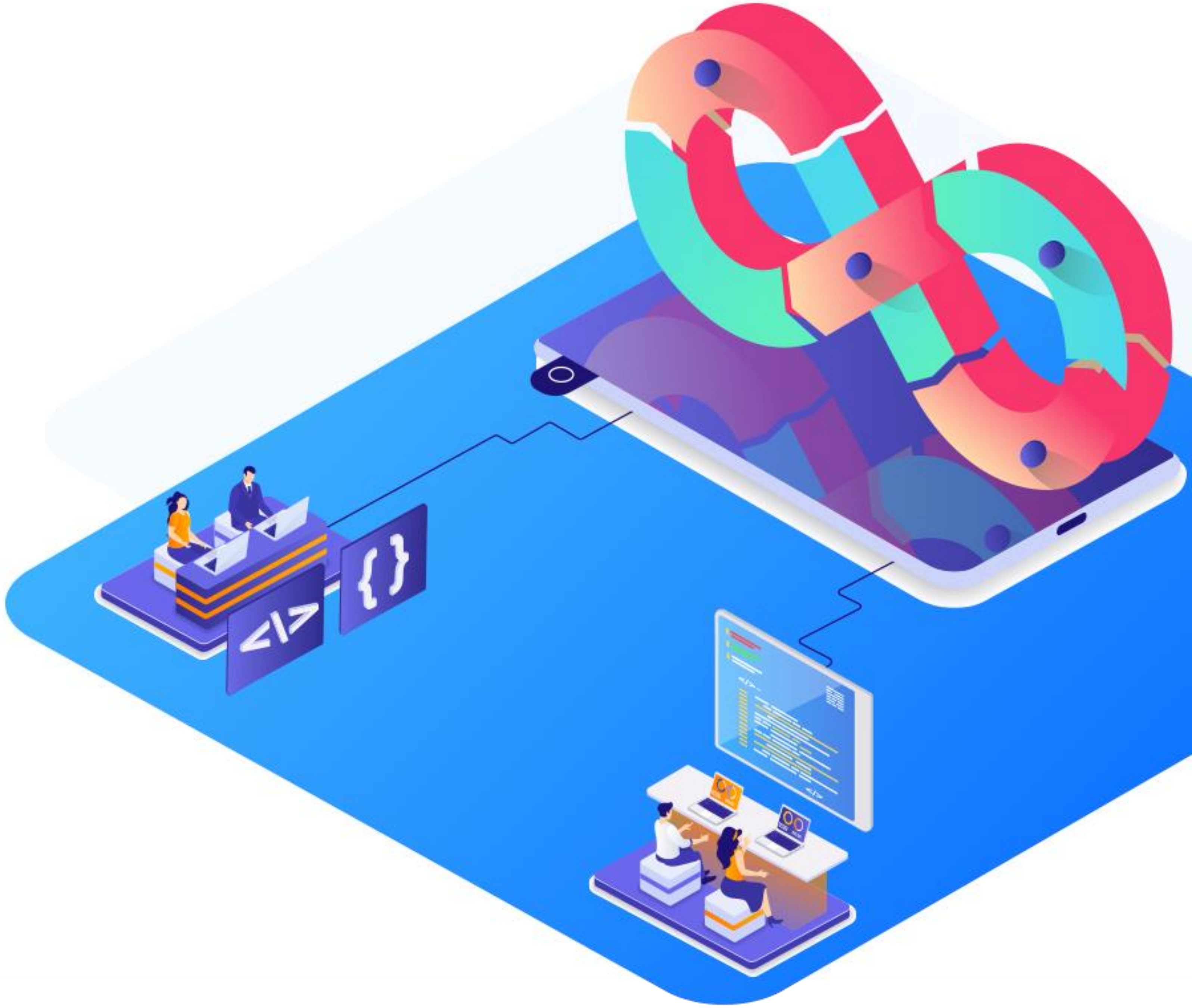
Monitoring and Logging in DevOps



Introduction to Monitoring and Prometheus

Learning Objectives

By the end of this lesson, you will be able to:

- Implement continuous monitoring in DevOps practices and record improvements for system performance and reliability
- Identify the right monitoring tool based on the application and the infrastructure to set up a robust monitoring system
- Configure basic infrastructure monitoring using Zabbix to track system performance and resource usage

Monitoring and Logging: Overview

What Is Monitoring and Logging?

Monitoring

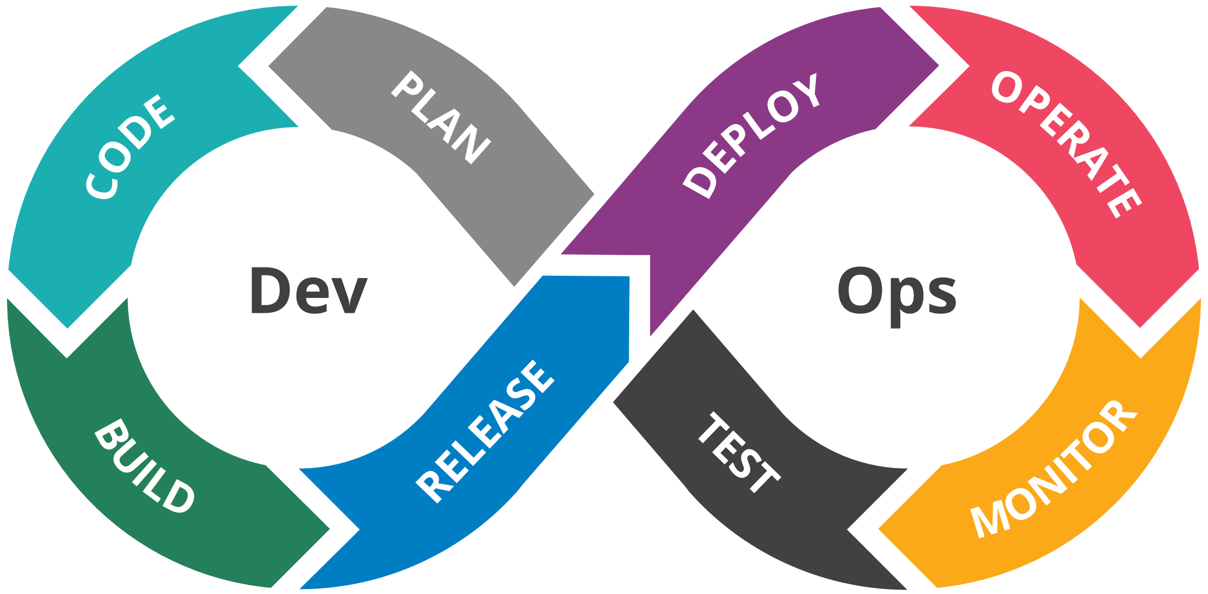
For tracking the performance, availability, and security of applications, systems, and infrastructure in real time

Logging

For recording and storing detailed information about system events, errors, and user activities within applications or infrastructure

Monitoring and Logging in DevOps

They are part of the monitor phase of the DevOps lifecycle. The infrastructure setup to perform monitoring and logging activities is called monitoring and logging system.



This system aims to continuously track system performance and detect issues in real time, ensuring the stability and reliability of the application in production.

Why Is Monitoring and Logging Used in DevOps?

Following are the key reasons depicting the importance of monitoring and logging:

Proactive issue identification and resolution

Analyze monitoring data in real-time to foresee anomalies and prevent incidents before they affect system performance or user experience

Performance optimization and resource utilization

Analyze resource consumption and performance data to identify bottlenecks, optimize configurations, and enhance overall system efficiency

Why Is Monitoring and Logging Used in DevOps?

Following are the key reasons depicting the importance of monitoring and logging:

Data-driven decision making

Utilize empirical data from continuous monitoring to make informed decisions, reducing reliance on guesswork and improving strategic planning

Continuous improvement and innovation

Leverage feedback from monitoring data to refine processes, adopt best practices, and foster a culture of innovation and continuous improvement

Why Is Monitoring and Logging Used in DevOps?

Following are the key reasons depicting the importance of monitoring and logging:

Compliance and security assurance

Monitor systems continuously to ensure compliance with regulations and promptly address security threats

Operational visibility and collaboration

Centralize monitoring data to provide comprehensive visibility, enhancing cross-team collaboration and informed decision-making

How Industries Utilize Monitoring and Logging

Healthcare

Healthcare organizations leverage monitoring and logging to ensure real-time access to critical medical records, data and services, maintaining system performance and compliance with regulatory standards.

IT Industries

IT companies use monitoring and logging to maintain consistent system performance. This enables rapid identification of infrastructure issues and ensures uptime in distributed environments.

How Industries Utilize Monitoring and Logging

Finance

Financial institutions implement monitoring and logging to track various online transactions for compliance with regulations and detect fraudulent activities, enhancing security and reducing risks.

Retail

Retailers utilize monitoring to manage traffic spikes and system performance during high-demand seasons, ensuring fast response times and preventing system failures during critical shopping periods.

How Industries Utilize Monitoring and Logging

Telecommunications

Telecom companies implement continuous monitoring to track network performance. This helps reduce service disruptions and ensures fast recovery times, improving overall service quality and customer satisfaction.

Business Use Cases

Real-time application monitoring

- **Scenario:** An e-commerce platform experiences sporadic slowdowns during high-traffic periods, impacting user experience.

- **Use Case:** Continuous monitoring enables real-time tracking of system performance metrics like CPU usage, memory consumption, and response time. This helps the platform detect slowdowns or performance bottlenecks early and automatically trigger resource scaling or optimizations to maintain smooth operations during peak periods like sales events.

Business Use Cases

Security threat detection and prevention

- **Scenario:** A financial institution needs to ensure its systems are protected from potential security breaches or fraud attempts.
- **Use Case:** Continuous logging allows the institution to collect and analyze system logs to identify unusual activity or security threats in real-time. This proactive monitoring helps detect anomalies such as unauthorized access or suspicious transactions, enabling swift incident response and mitigating potential risks before they escalate.

Business Use Cases

Compliance monitoring

- **Scenario:** A healthcare provider must comply with strict data protection regulations such as HIPAA. They need to ensure that sensitive patient data is always protected, and system activities are traceable.

- **Use Case:** Continuous logging tracks all access to patient data, generating real-time alerts for any unauthorized access attempts. This ensures compliance with data protection regulations and helps identify breaches before any sensitive data is compromised.

Business Use Cases

Resource optimization

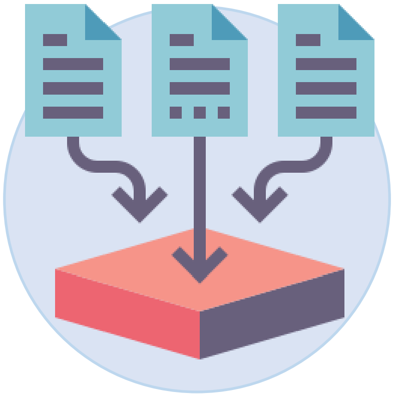
- **Scenario:** A cloud service provider experiences inefficient resource allocation, leading to high costs and system underperformance.

- **Use Case:** Continuous monitoring analyzes system utilization in real time, optimizing CPU, memory, and storage usage by dynamically adjusting resource allocations based on demand. This prevents over-provisioning, reduces costs, and ensures efficient system performance during peak loads.

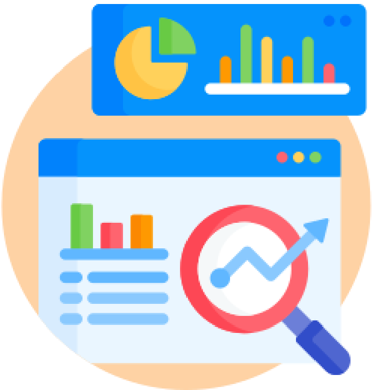
What Is Continuous Monitoring?

It is a skill that helps in tracking real time system performance using a monitoring system.

It includes the following key components:



Data collection



Data analysis

Data action

Components of Continuous Monitoring

Data collection

Collect data continuously from applications, servers, and networks for maintaining performance and reliability

Data analysis

Analyze data in real time for identifying issues and performance bottlenecks

Data action

Trigger proactive alerts to detect anomalies or threats in the application for timely feedback and response to the data outcomes

What Is Evaluation?

Evaluation (a part of the monitor phase under the DevOps lifecycle) aims to improve the overall efficiency of the monitoring and logging system by identifying improvements and measuring its reliability.

The result of the evaluation helps stakeholders understand if the system is working toward fulfilling the business requirements.

Core Concepts of Evaluation

These core concepts provide a structured approach for evaluating projects, systems, and outcomes, ensuring data-driven decisions and continuous improvement:

Periodic process

Conducts evaluations at specific intervals, such as the end of a project or during key decision points

Outcome-based assessment

Aims to measure the achievement of specific objectives and goals

Data-driven analysis

Uses quantitative and qualitative data to provide evidence-based conclusions

Difference between Monitoring and Evaluation

The following factors outline how monitoring is distinct from evaluation:

Aspect	Monitoring	Evaluation
Process	Measure indicators and metrics continuously	Analyze data periodically at milestones
Goal	Identify issues for prompt action and correction	Provide insights for decision-making and improvement
Timing	Conduct as part of daily activities	Conduct at specific intervals or milestones
Focus	Track current performance and issues	Assess overall effectiveness and outcomes
Scope	Focus on specific metrics and events	Consider multiple factors and dimensions
Purpose	Enable immediate action and correction	Support strategic planning and improvement

Quick Check

A global e-commerce company faces occasional system slowdowns during peak traffic, affecting user experience. They have implemented continuous monitoring to detect issues early and prevent impact. How does continuous monitoring enable proactive issue identification and resolution in this scenario?

A. By detecting anomalies early and allowing immediate resource adjustments to prevent incidents

B. By deferring system updates to limit potential disruptions

C. By reducing the frequency of monitoring to conserve system resources

D. By increasing manual system checks to avoid downtime

Monitoring Fundamentals

Components of a Monitoring System

The following are the components to monitor the performance and health of systems, applications, and infrastructure:

Monitoring agents or exporters	Collect and report metrics, logs, and data from systems and applications
Data collection, transport, and storage	Ensure secure transport and efficient storage of data using time-series databases
Data processing and enrichment	Parse, structure, and enrich data with tools like Logstash or Fluentd

Components of a Monitoring System

The following are the components to monitor the performance and health of systems, applications, and infrastructure:

Visualization and dashboarding

Visualize and analyze data using dashboards like Grafana or Kibana

Alerting, notification and access control

Generate alerts, notify teams, and manage access control and authentication for sensitive data

Continuous Monitoring Tools

In modern IT environments, these tools provide real-time insights, maintain system reliability, and enable quick issue resolution. Some popular tools include:

- Prometheus
- Grafana
- Elasticsearch, Logstash, Kibana (ELK)
- Datadog
- New Relic
- Dynatrace
- Splunk
- AppDynamics

Continuous Monitoring Tools

In modern IT environments, these tools provide real-time insights, maintain system reliability, and enable quick issue resolution. Some popular tools include:

Graylog

Pingdom

Nagios

Paessler PRTG

Zabbix

Continuous Monitoring Tools: Use Case

Prometheus

It is used for monitoring systems and services, especially in cloud-native environments with Kubernetes.

Grafana

It is used for visualizing metrics and logs with customizable dashboards across multiple data sources.

Continuous Monitoring Tools: Use Case

Elasticsearch, Logstash, Kibana (ELK)

It is used for log aggregation, search, and visualization in large-scale data analytics.

Datadog

It is used for monitoring infrastructure, application performance, and log management in cloud environments.

Continuous Monitoring Tools: Use Case

New Relic

It is used for monitoring application performance and infrastructure in distributed systems.

Dynatrace

It is used for monitoring applications, microservices, and cloud environments with AI-driven insights.

Continuous Monitoring Tools: Use Case

Splunk

It is used for log analysis, monitoring, and security information event management (SIEM).

AppDynamics

It is used for tracking application performance in complex IT environments.

Continuous Monitoring Tools: Use Case

Graylog

It is used for centralized log management and analysis in various IT infrastructures.

solarwinds pingdom

Pingdom

It is used for monitoring website and server uptime with real-time alerts.

Continuous Monitoring Tools: Use Case

Nagios

Nagios

It is used for monitoring IT infrastructure, including network, application, and system health.

PAESSLER PRTG NETWORK MONITOR

Paessler PRTG

It is used for monitoring networks, bandwidth, and system health.

Continuous Monitoring Tools: Use Case

ZABBIX

Zabbix

It is a widely used monitoring tool designed for tracking the health and performance of IT infrastructure, including networks, servers, virtual machines, and cloud services.

It enables automated alerting based on customizable triggers, helping organizations to proactively manage incidents and reduce downtime through timely notifications.

Continuous Monitoring Tools

Tool	Key features	Benefits
Prometheus	• Open-source, flexible integration • Time-series database • Built-in alerting	• Reduces downtime by early issue detection • Optimizes resource allocation • Lowers costs (open-source)
Grafana	• Customizable, real-time dashboards • Broad data source compatibility • Visualizations and alerts	• Enhances decision-making speed • Reduces incident response time • Improves operational efficiency
Elasticsearch, Logstash, Kibana (ELK)	• Centralized log management • Real-time data search and analysis • Open-source solution	• Lowers operational costs • Helps meet compliance needs • Improves incident response
Datadog	• Full-stack cloud monitoring • Built-in log management and APM • Real-time security insights	• Increases uptime • Enhances customer satisfaction • Reduces complexity by consolidating tools

Continuous Monitoring Tools

Tool	Key features	Benefits
New Relic	• Comprehensive APM • Full-stack observability • Easy system-wide monitoring	• Improves user experience • Enhances service reliability • Reduces troubleshooting time
Dynatrace	• AI-driven monitoring • Cloud infrastructure observability • Automation capabilities	• Reduces operational overhead through automation • Minimizes downtime • Supports scalability
Splunk	• Advanced big data analytics • Real-time log monitoring • Scalable architecture for large datasets	• Improves operational visibility • Enhances data-driven decision making • Reduces risk through proactive insights
AppDynamics	• Application performance tracking • Real-time business transaction insights • Detailed diagnostics	• Improves user experience • Increases business transaction efficiency • Supports business continuity

Continuous Monitoring Tools

Tool	Key features	Benefits
Graylog	• Centralized log management • Real-time log analysis • Log data indexing • Custom alerting	• Provides actionable insights. • Improves security event management. • Enables faster troubleshooting.
Pingdom	• Website and application performance monitoring • Real-time uptime alerts • Synthetic transaction testing	• Ensures website reliability • Enhances user experience • Minimizes downtime, increasing revenue
Nagios	• Infrastructure, service, and application monitoring • Custom alerting • Extensible with plugins	• Ensures system availability • Reduces downtime • Allows custom monitoring for diverse environments
Paessler PRTG	• Network and bandwidth monitoring • Distributed monitoring • Alerts via various channels	• Enhances network reliability • Identifies bottlenecks • Reduces operational costs

Continuous Monitoring Tools

Tool	Key features	Benefits
Zabbix	• Agent-based and agentless monitoring • Custom alerting • SNMP and IPMI monitoring • Visualization	• Ensures system health • Provides real-time performance data • Enables predictive maintenance

Assisted Practice

Setting up Basic Infrastructure Monitoring Using Zabbix

Duration: 10 Min.

Problem statement:

You have been assigned a task to set up basic infrastructure monitoring using the open-source monitoring tool Zabbix for tracking server performance, network devices, and application status across your IT environment.

Outcome:

By the end of this demo, you will be able to set up basic infrastructure monitoring using Zabbix to track server performance, network devices, and application status in your IT environment.

Note: Refer to the demo document for detailed steps:
01_Setting_Up_Basic_Infrastructure_Monitoring_Using_Zabbix

Assisted Practice: Guidelines

Steps to be followed:

1. Install Zabbix packages
2. Install the database
3. Create an initial database
4. Configure the database for the Zabbix server
5. Start Zabbix and Apache servers
6. Configure and explore the Zabbix Console

Types of Monitoring

It is categorized based on different aspects or components, such as:

- Application performance monitoring (APM)
- Infrastructure monitoring
- Network monitoring
- Log monitoring
- Synthetic monitoring
- Real user monitoring (RUM)

Application Performance Monitoring (APM)

- Monitors application performance, availability, and user experience
- Tracks metrics like response times, error rates, resource utilization, and user interactions
- **Tools**: New Relic, AppDynamics, Datadog APM

Infrastructure Monitoring

- Monitors the performance and health of servers, networks, and storage systems
- Tracks metrics such as CPU, memory, disk usage, network traffic, and uptime
- **Tools**: Prometheus, Nagios, Zabbix

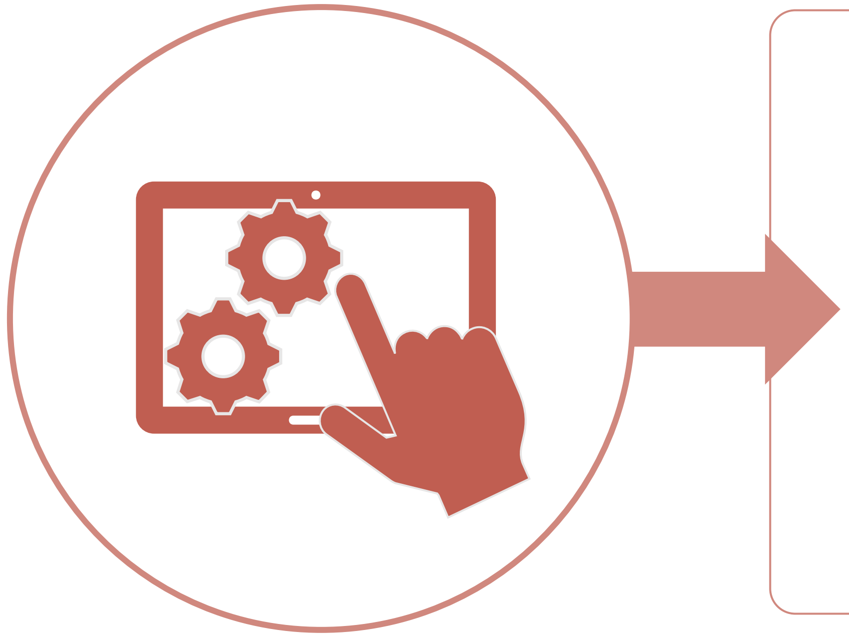
Network Monitoring

- Monitors the performance and availability of network infrastructure
- Tracks metrics like bandwidth utilization, packet loss, and latency
- **Tools**: SolarWinds Network Performance Monitor, Paessler PRTG, Nagios

Log Monitoring

- Collects and analyzes log data from applications, servers, and network devices
- Helps identify issues, troubleshoot problems, and conduct audits
- **Tools**: Graylog, Logstash, Splunk

Synthetic Monitoring



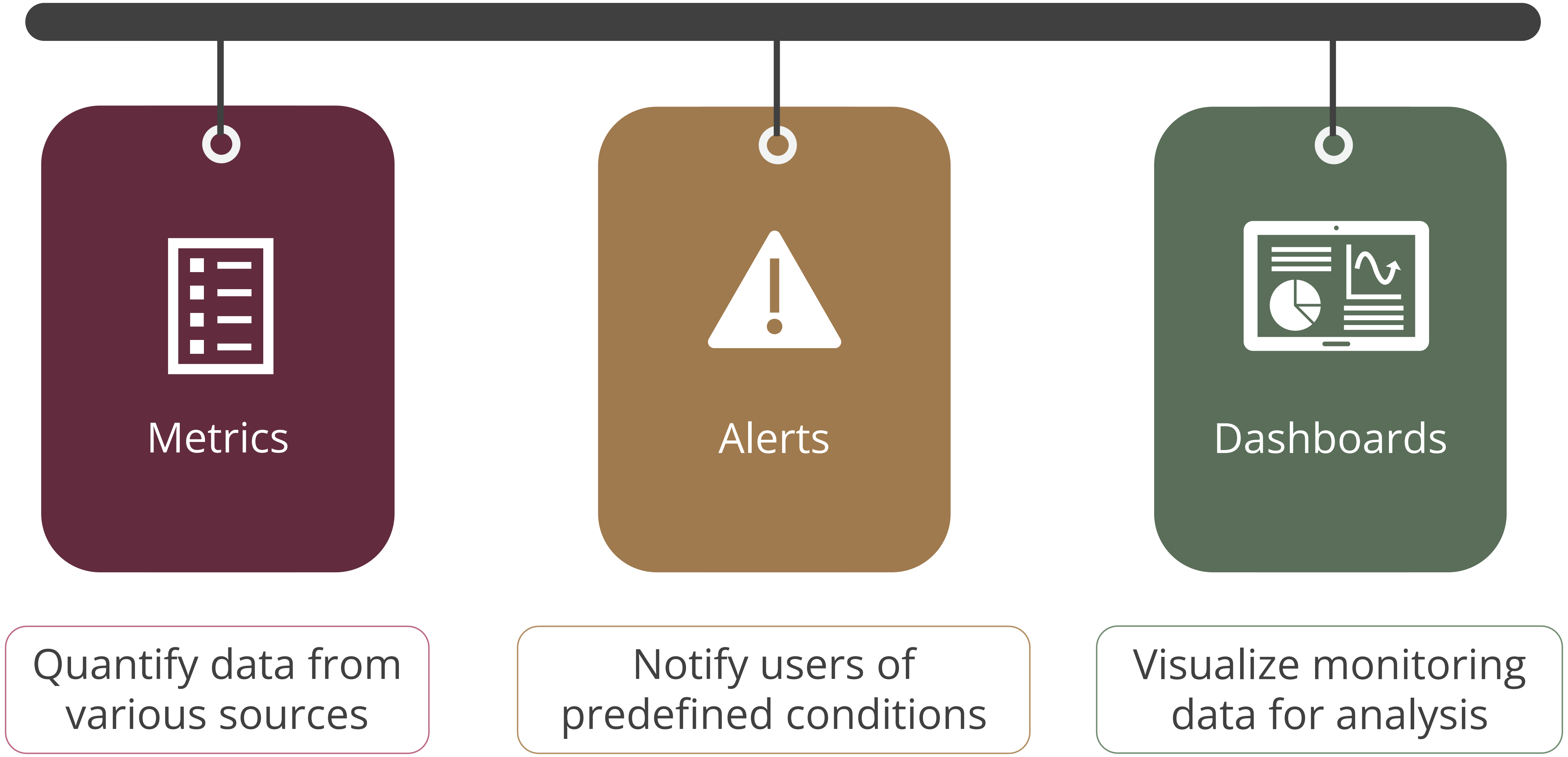
- Simulates user interactions and monitors application performance from various locations
- Helps identify issues before users are impacted
- **Tools**: Pingdom, Catchpoint, Dynatrace Synthetic Monitoring

Real User Monitoring (RUM)

- Monitors the performance and user experience from the perspective of actual end-users
- Tracks metrics like page load times, client-side errors, and user interactions
- **Tools**: New Relic Browser, Dynatrace Real User Monitoring, AppDynamics Browser Real User Monitoring

Terminologies: Metrics, Alerts, and Dashboards

They are the key elements of every monitoring system. Different tools represent these elements in different ways, and the synergy between them helps implement effective monitoring.



Metrics

Quantify data from various sources

Alerts

Notify users of predefined conditions

Dashboards

Visualize monitoring data for analysis

Metrics

Metrics

Collect quantifiable measurements from applications, systems, and infrastructure

Provide insights into performance, health, and behavior

Track CPU utilization, memory usage, error rates and other metrics

Store data in a time-series database for analysis and visualization

Types of Metrics in a Monitoring System

These are collected from diverse sources like operating systems, applications, network devices, databases, and monitoring agents to provide comprehensive insights:

System metrics

Application metrics

Infrastructure metrics

Container and orchestration metrics

Availability and uptime metrics

Business and user experience metrics

Use Cases for Metrics

System metrics

Monitoring the CPU and memory usage of a web server to ensure it can manage high traffic periods and prevent slowdowns or crashes.

Application metrics

Tracking an online store's checkout process to identify how frequently errors occur, allowing issues to be resolved before they affect too many customers.

Infrastructure metrics

Evaluating the health and performance of cloud servers to ensure they remain operational, providing uninterrupted service for users.

Use Cases for Metrics

Container and orchestration metrics

Tracking resource usage in a Kubernetes cluster to ensure smooth operation and prevent container failures.

Availability and uptime metrics

Monitoring a software service to maintain a 99.99% uptime, ensuring fewer disruptions and meeting customer expectations.

Business and user experience metrics

Analyzing how many users complete the checkout process on a retail website to identify potential issues and implement changes that boost sales.

Alerts

Trigger notifications when predefined conditions or thresholds are met

Act on rules or specific criteria defined within the alert system

Activate when metrics exceed set limits or specific events occur

Enable quick response by notifying the appropriate teams or individuals

Dashboards

Display monitoring data through charts, graphs, and visualizations

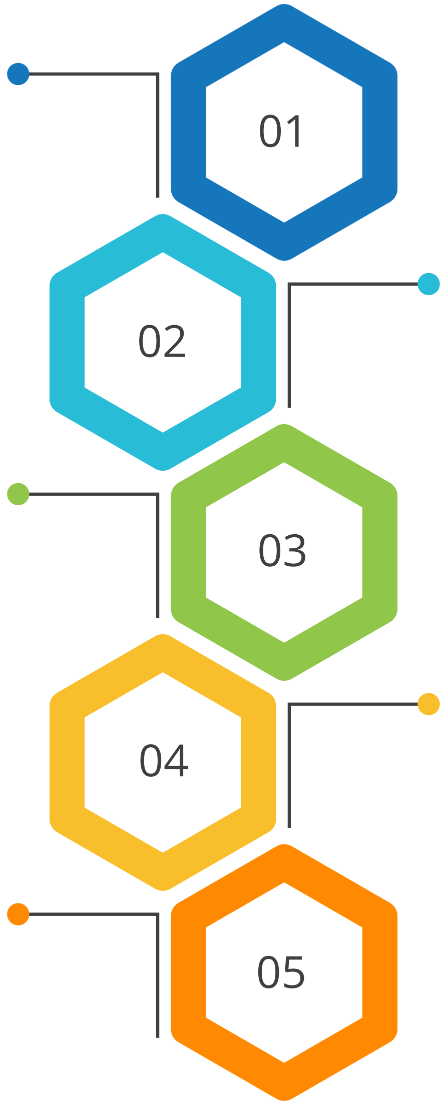
Offer a centralized view of system health, performance, and status

Customize to show relevant metrics, alerts, and information for different needs

Enable the identification of trends, patterns, and anomalies, and support collaboration

Metrics, Alerts, and Dashboards: Benefits

Effective utilization of these components within a monitoring system enables organizations to achieve the following key benefits:



Gain visibility into system performance and health

Detect and address issues proactively to minimize downtime

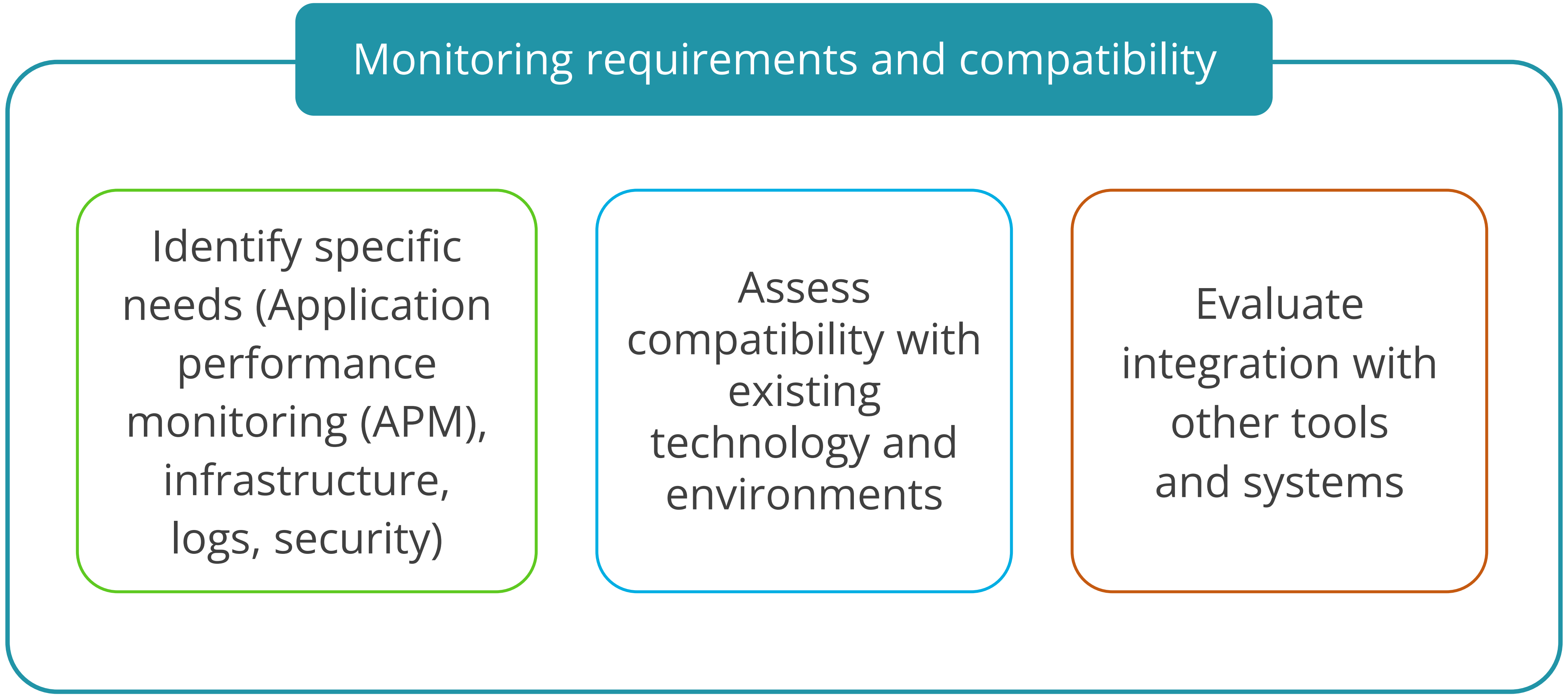
Gain insights from historical data to identify trends for optimization

Share insights to enhance collaboration and transparency

Make informed decisions based on actionable data

Choosing the Right Monitoring Tools

The following are key factors and things to be considered within each factor while selecting a monitoring tool for effective continuous monitoring:



Monitoring requirements and compatibility

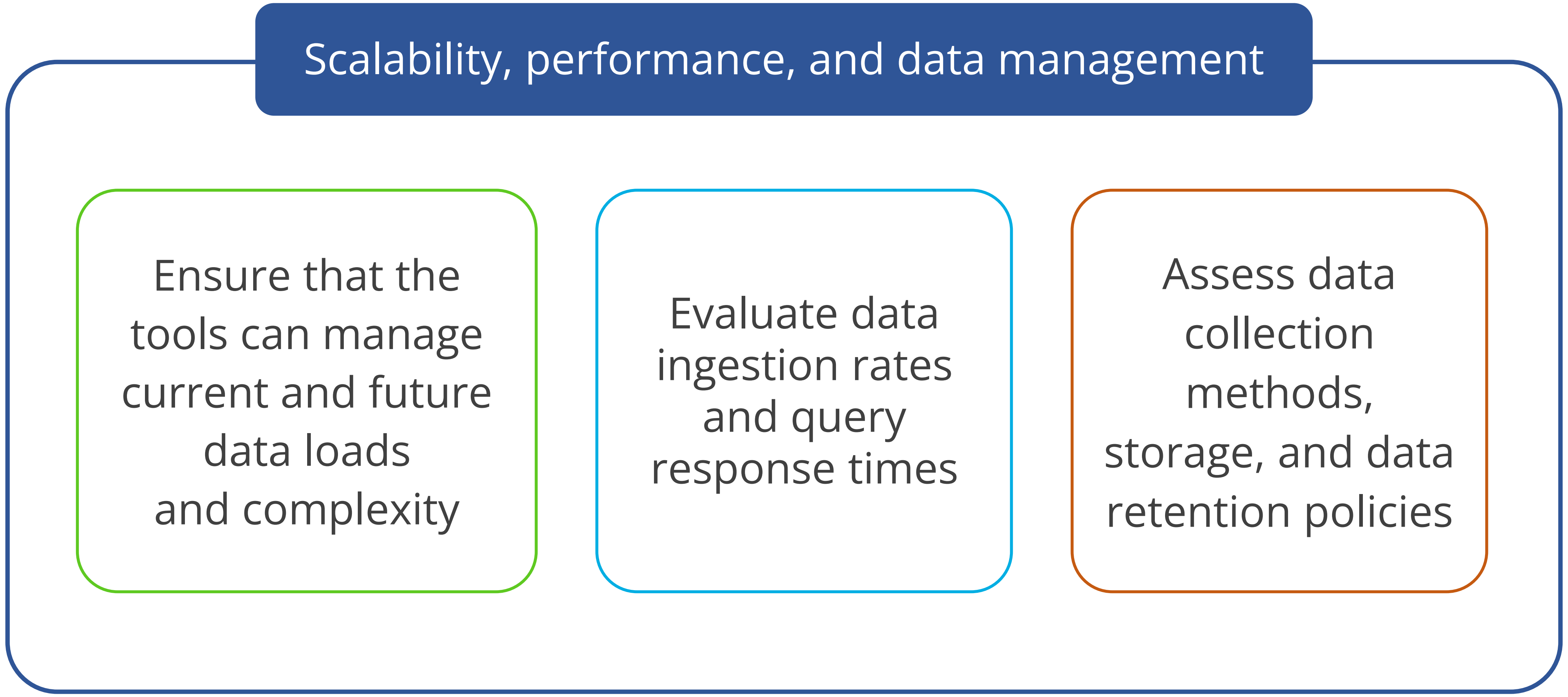
Identify specific needs (Application performance monitoring (APM), infrastructure, logs, security)

Assess compatibility with existing technology and environments

Evaluate integration with other tools and systems

Choosing the Right Monitoring Tools

The following are key factors in selecting monitoring tools for effective continuous monitoring:



Scalability, performance, and data management

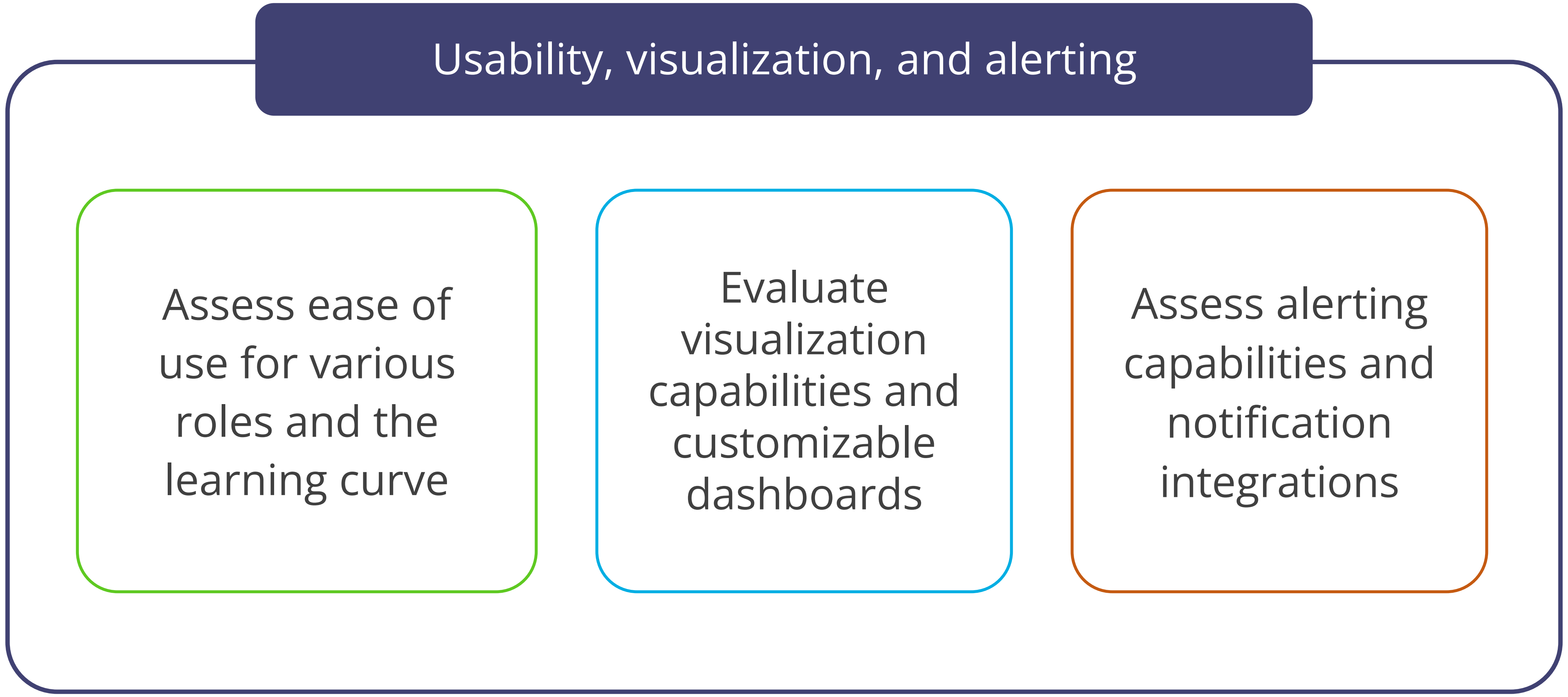
Ensure that the tools can manage current and future data loads and complexity

Evaluate data ingestion rates and query response times

Assess data collection methods, storage, and data retention policies

Choosing the Right Monitoring Tools

The following are key factors in selecting monitoring tools for effective continuous monitoring:



Usability, visualization, and alerting

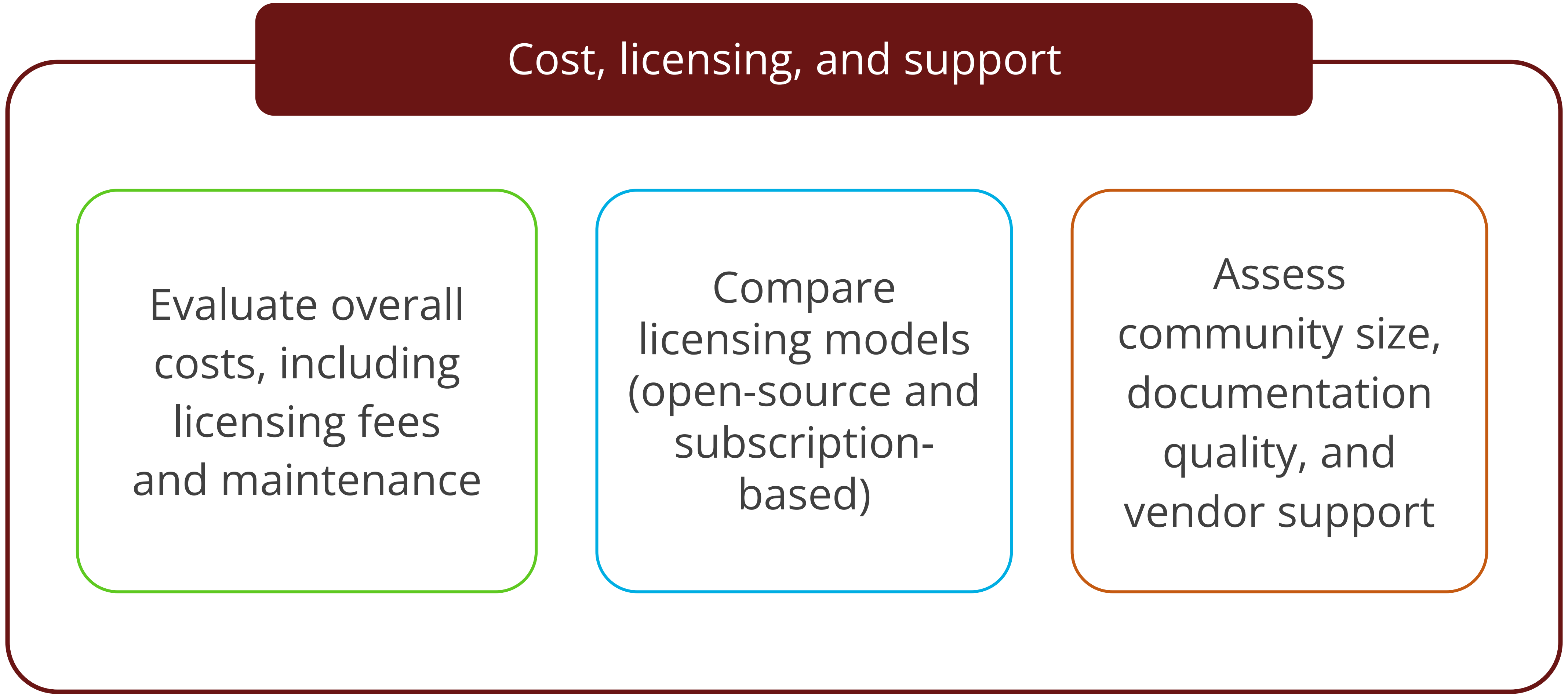
Assess ease of use for various roles and the learning curve

Evaluate visualization capabilities and customizable dashboards

Assess alerting capabilities and notification integrations

Choosing the Right Monitoring Tools

The following are key factors in selecting monitoring tools for effective continuous monitoring:



Cost, licensing, and support

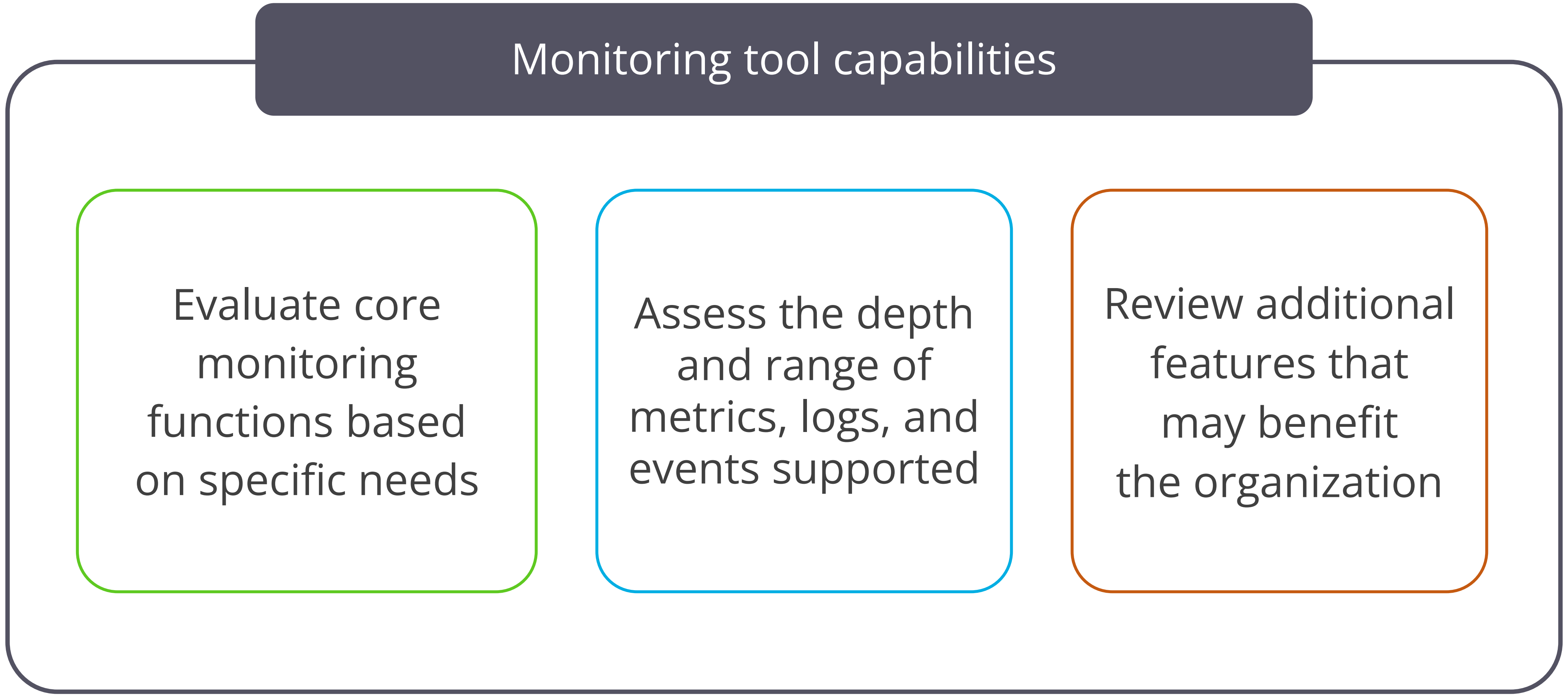
Evaluate overall costs, including licensing fees and maintenance

Compare licensing models (open-source and subscription-based)

Assess community size, documentation quality, and vendor support

Choosing the Right Monitoring Tools

The following are key factors in selecting monitoring tools for effective continuous monitoring:



Monitoring tool capabilities

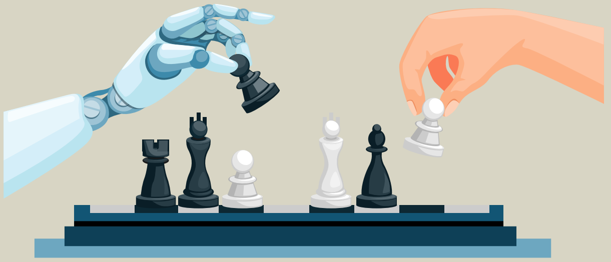
Evaluate core monitoring functions based on specific needs

Assess the depth and range of metrics, logs, and events supported

Review additional features that may benefit the organization

Use Case

Continuous Monitoring for Risk Management at JPMorgan Chase

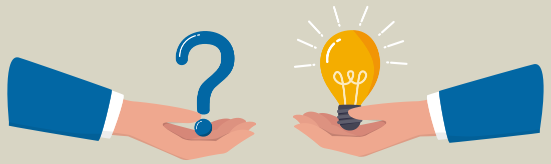


Problem

JPMorgan Chase, one of the largest banks in the United States, faces the challenge of identifying potential risks and issues in real time, particularly around money laundering and fraudulent activities.

Use Case

Continuous Monitoring for Risk Management at JPMorgan Chase



Solution

- The bank implemented a continuous monitoring system to track customer transactions and activities in real time.
- This system monitors vast amounts of data to detect suspicious patterns or anomalies related to fraud and money laundering.
- The system analyzes each transaction and customer behavior to flag potentially illegal activities early.

Use Case

Continuous Monitoring for Risk Management at JPMorgan Chase

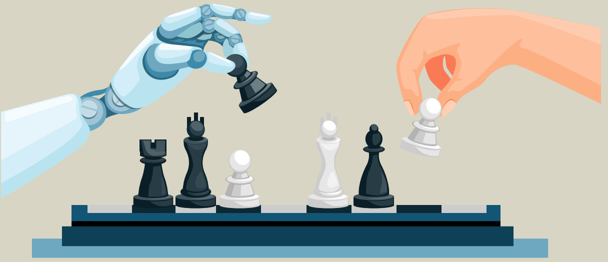
Outcome

By utilizing continuous monitoring, JPMorgan Chase has been able to proactively detect and mitigate risks before they escalate into major issues.

This has led to increased operational efficiency, reduced fraudulent activities, and significant savings in time and resources by preventing costly incidents.

Use Case

Continuous Monitoring for Security at Amazon

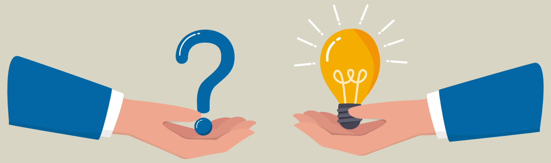


Problem

Amazon, the world’s largest online retailer, needs to ensure the security of its customers' data amidst growing cybersecurity threats and vulnerabilities.

Use Case

Continuous Monitoring for Security at Amazon



Solution

- The company implemented a continuous monitoring system that scans its network for potential security breaches and vulnerabilities in real time.
- This system detects anomalies and security risks as they arise, allowing Amazon to address potential threats before they escalate.

Use Case

Continuous Monitoring for Security at Amazon

Outcome

By using continuous monitoring, Amazon has stayed ahead of potential cyber threats, safeguarding customer data and maintaining trust.

This proactive approach has enhanced the company's ability to protect its network while ensuring uninterrupted service to its global customer base.

Quick Check

A DevOps team is responsible for maintaining a web application that serves a global user base. To ensure consistent availability and optimal performance, they want to proactively detect issues before real users experience them. The team decides to simulate user interactions with the web application from different geographical locations. Which type of monitoring should they use?

A. Real user monitoring (RUM)

B. Synthetic monitoring

C. Log monitoring

D. Infrastructure monitoring

Introduction to Prometheus

What Is Prometheus?

It is a widely used tool for monitoring and alerting. It is used for applications built with containers, microservices, or cloud technology.

It collects and saves data over time, like a series of snapshots. Each piece of data, known as a metric, is saved with the timestamp it was collected and includes labels for additional information.

Key Features of Prometheus

Prometheus offers capabilities to effectively monitor and manage systems that include:

Data model

Utilizes a flexible data model where metrics are identified by their name and labeled with key-value pairs

Query language

Uses Prometheus Query Language (PromQL) to explore data in various ways

Independent operation

Operates without relying on complex storage systems and functions autonomously with single server nodes

Key Features of Prometheus

Prometheus offers capabilities to effectively monitor and manage systems that include:

Data collection method

Utilizes a **pull** method over HTTP to gather data and also supports data pushed through a gateway

Target discovery

Uses automatic discovery or a fixed configuration to locate required monitoring targets

Visualization

Provides various options for creating graphs and dashboards, enabling effective data visualization

Terminologies in Prometheus

Key terms that define how data is collected, organized, and analyzed in Prometheus include:

Metrics

Describes a numerical measurement of a monitored system or application

Time series

Represents a stream of data points for a specific metric over time

Labels

Provides key-value pairs attached to a time series for additional context

Targets

Identifies monitored endpoints that expose metrics for Prometheus to scrape

Terminologies in Prometheus

Key terms that define how data is collected, organized, and analyzed in Prometheus include:

Job

Groups related targets that serve the same purpose

Scrape

Refers to the process where Prometheus collects metrics by making HTTP requests to the metrics endpoints

PromQL

Provides a powerful query language to filter, aggregate, and perform mathematical operations on collected metrics

Alerts

Defines rules that trigger notifications when certain conditions based on metric values are met

Terminologies in Prometheus

Key terms that define how data is collected, organized, and analyzed in Prometheus include:

Exporters

Gathers metrics from various services and makes them available in a format Prometheus can read

Pushgateway

Allows metrics from batch jobs to be temporarily stored so they can be scraped by Prometheus later

Federation

Aggregates and queries metrics from multiple servers, enabling monitoring of distributed or large scale environments

Recording rules

Generates new metrics based on existing ones using pre-defined queries, which are stored to make future queries faster and more efficient

Prometheus vs. Zabbix

Prometheus and Zabbix are both robust monitoring tools, however Prometheus stands out in modern, dynamic infrastructures due to its cloud-native capabilities and advanced metric handling.

Prometheus	Zabbix
Is designed for seamless integration with cloud-native environments	Is designed for traditional, static infrastructure setups
Specializes in real-time, high-precision time-series monitoring	Focuses on general-purpose infrastructure monitoring
Scales easily using a pull-based model	Scales with complexity using an agent-based model

Prometheus vs. Zabbix

Prometheus and Zabbix are both robust monitoring tools, however Prometheus stands out in modern, dynamic infrastructures due to its cloud-native capabilities and advanced metric handling.

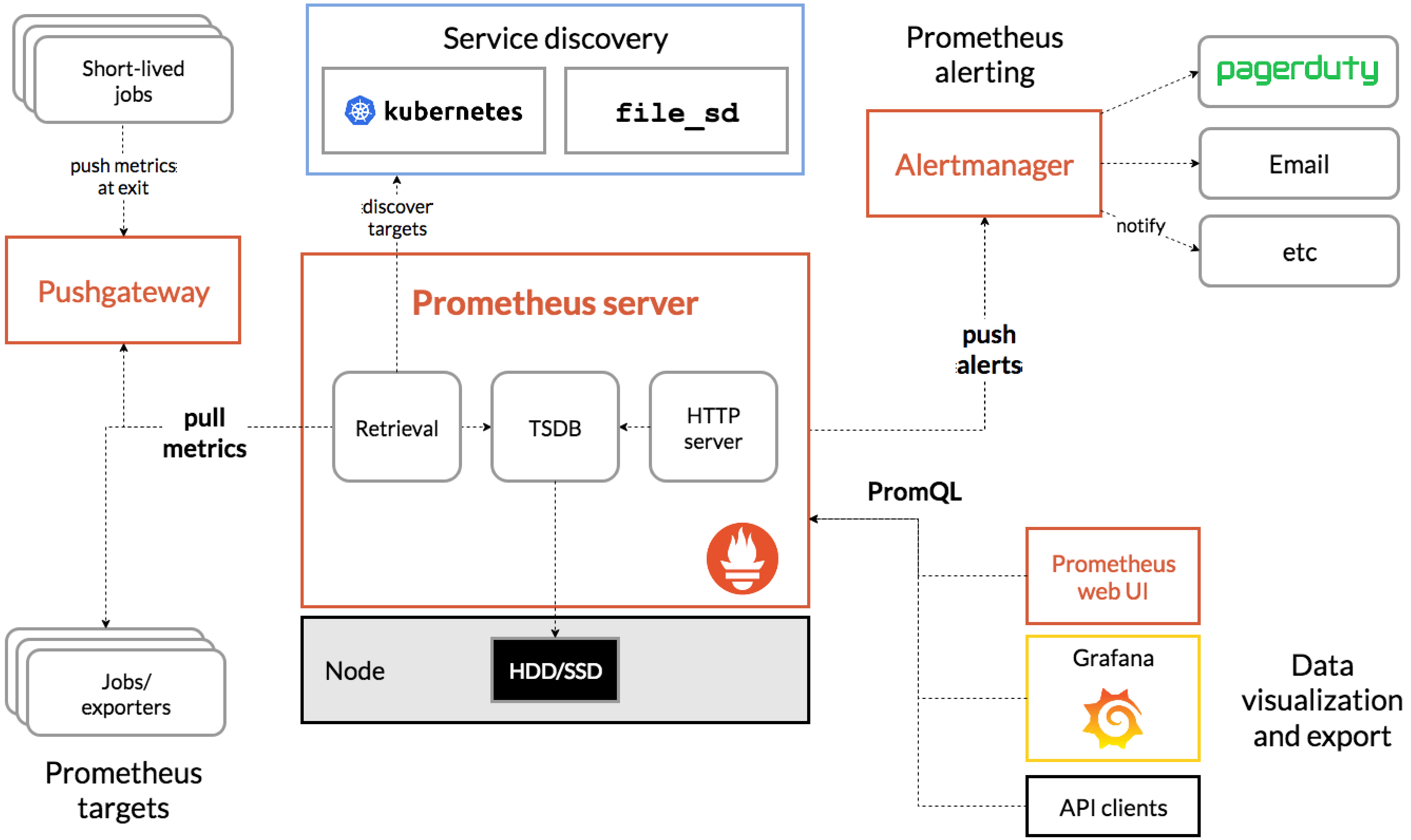
Prometheus	Zabbix
Provides flexible querying with PromQL for metrics	Relies on templates with less flexible querying
Excels at monitoring containers and microservices	Supports containers but requires more configuration

NOTE

Zabbix is another widely-used tool for monitoring infrastructure, but it provides limited features. Use Prometheus as it offers advanced capabilities for handling time-series data and powerful querying through PromQL.

Prometheus Architecture

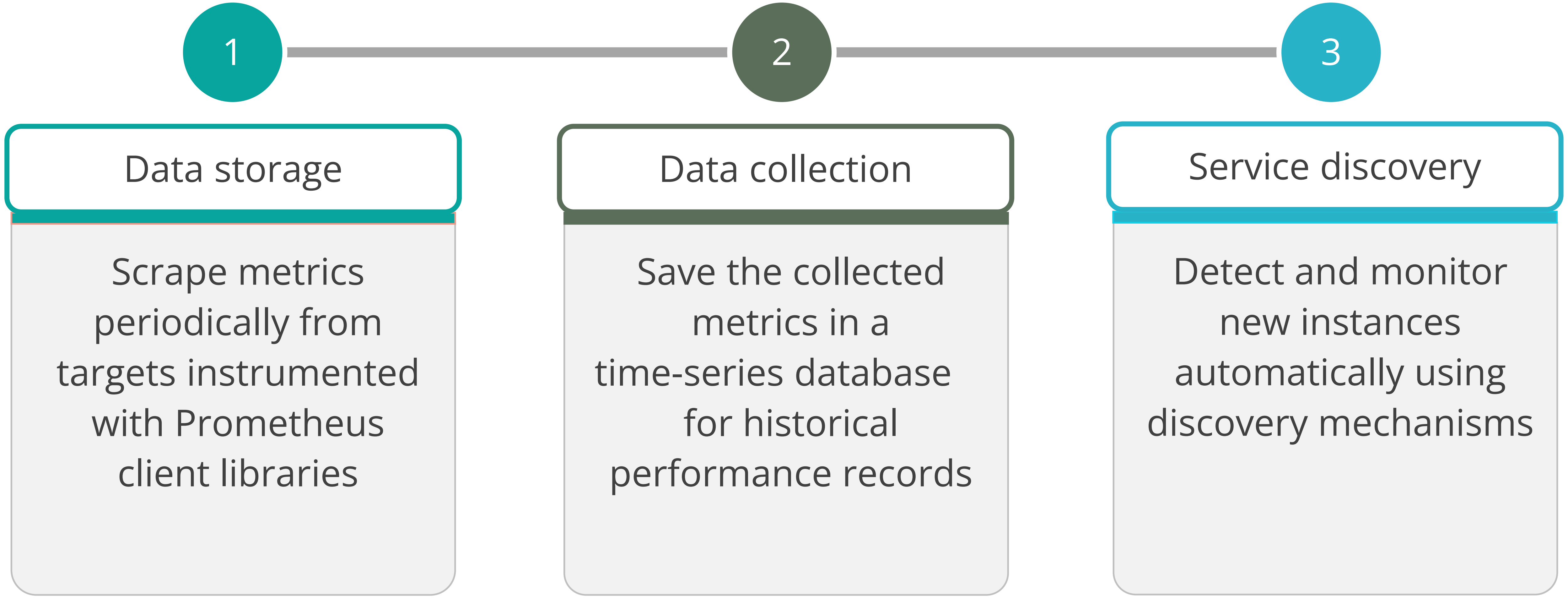
The below architecture shows how Prometheus is designed to collect, store, and query metrics from various sources:



Source: https://prometheus.io/docs/introduction/overview/

Working of Prometheus

Prometheus gathers and stores metrics data, using automated mechanisms to detect and monitor new instances. The following is a simplified breakdown of its operation:



1

Data storage

Scrape metrics periodically from targets instrumented with Prometheus client libraries

2

Data collection

Save the collected metrics in a time-series database for historical performance records

3

Service discovery

Detect and monitor new instances automatically using discovery mechanisms

Prometheus Architecture

Jobs or exporters

Collects metrics from monitored systems at regular intervals for scraping

Pushgateway

Accepts metrics from short-lived jobs and pushes them to the Prometheus server for aggregation

Service discovery

Discovers monitoring targets using Kubernetes or file-based discovery for dynamic target identification and configuration

Prometheus server

Retrieves metrics from targets, stores them in a time-series database (TSDB), and provides a query interface via PromQL

Storage (node)

Saves TSDB data on local storage for reliable data retention

Prometheus Architecture

Alertmanager

Triggers alerts based on metrics and sends notifications through channels for timely incident response

Prometheus Web UI

Visualizes and queries metrics using the Web UI for easy data analysis

Grafana

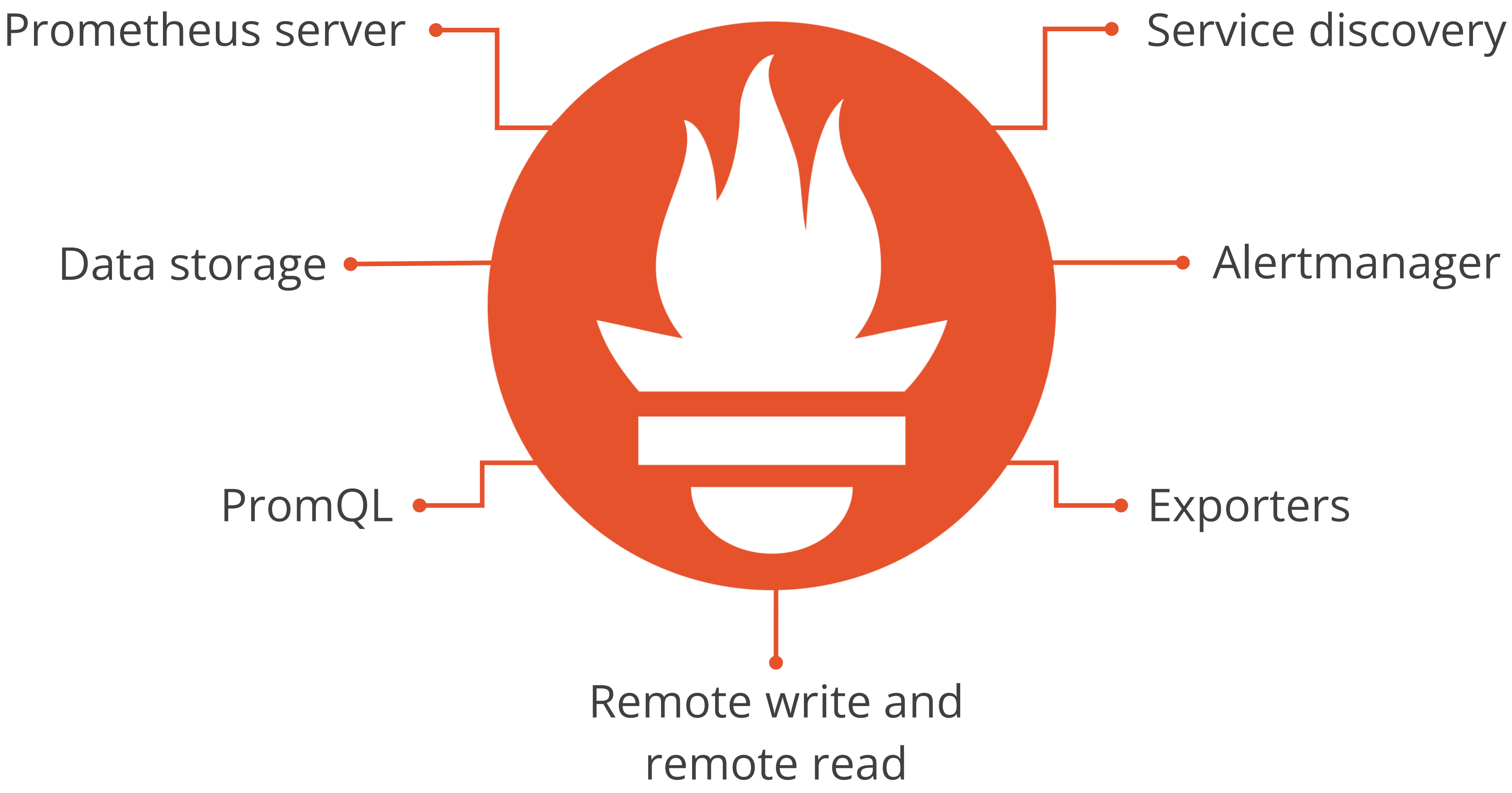
Creates advanced dashboards and queries metrics from Prometheus for enhanced visualization

API clients

Interacts with the Prometheus API for querying and programmatic access

Components of Prometheus

Prometheus uses a pull-based architecture, where the server scrapes metrics from targets. It consists of the following components:



Prometheus server

Data storage

PromQL

Remote write and remote read

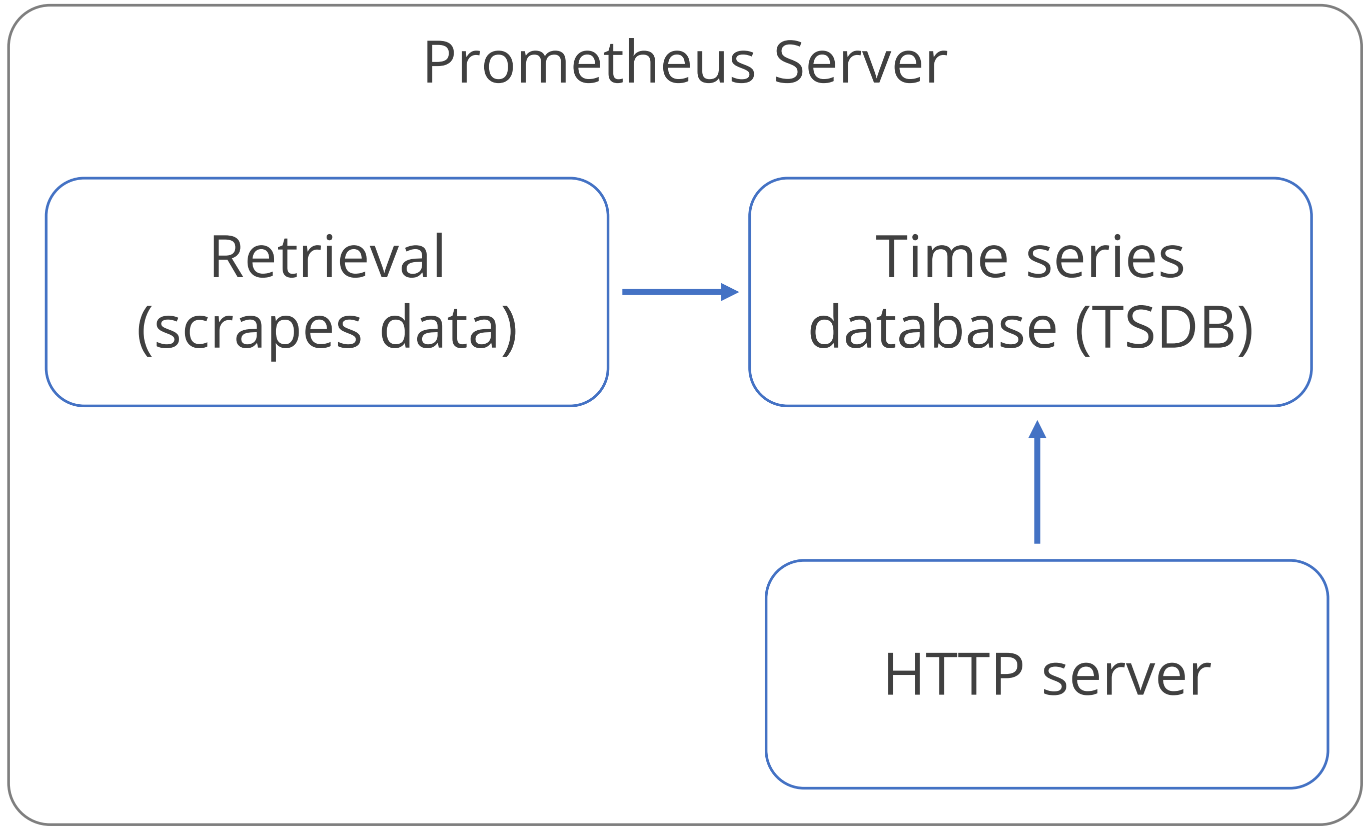
Service discovery

Alertmanager

Exporters

Prometheus Server

It acts as the central component responsible for scraping, storing, and providing access to the metrics data. The diagram below shows the core components:



- **Retrieval:** Scrapes or pulls metrics from the configured targets
- **TSDB:** Stores all the scraped metrics as time series data
- **HTTP server:** Allows users to query the stored metrics using PromQL

Data Storage

Prometheus provides flexible data storage options for the vast amount of metrics data that it collects. The following are some of the available options depending on specific needs:

Time Series Database (TSDB)

Local storage (HDD and SSD)

Remote storage (Thanos, Cortex)

Exporters

They work as third-party applications or libraries that expose metrics from various systems and services in a Prometheus readable format. Some of the popular exporters include:

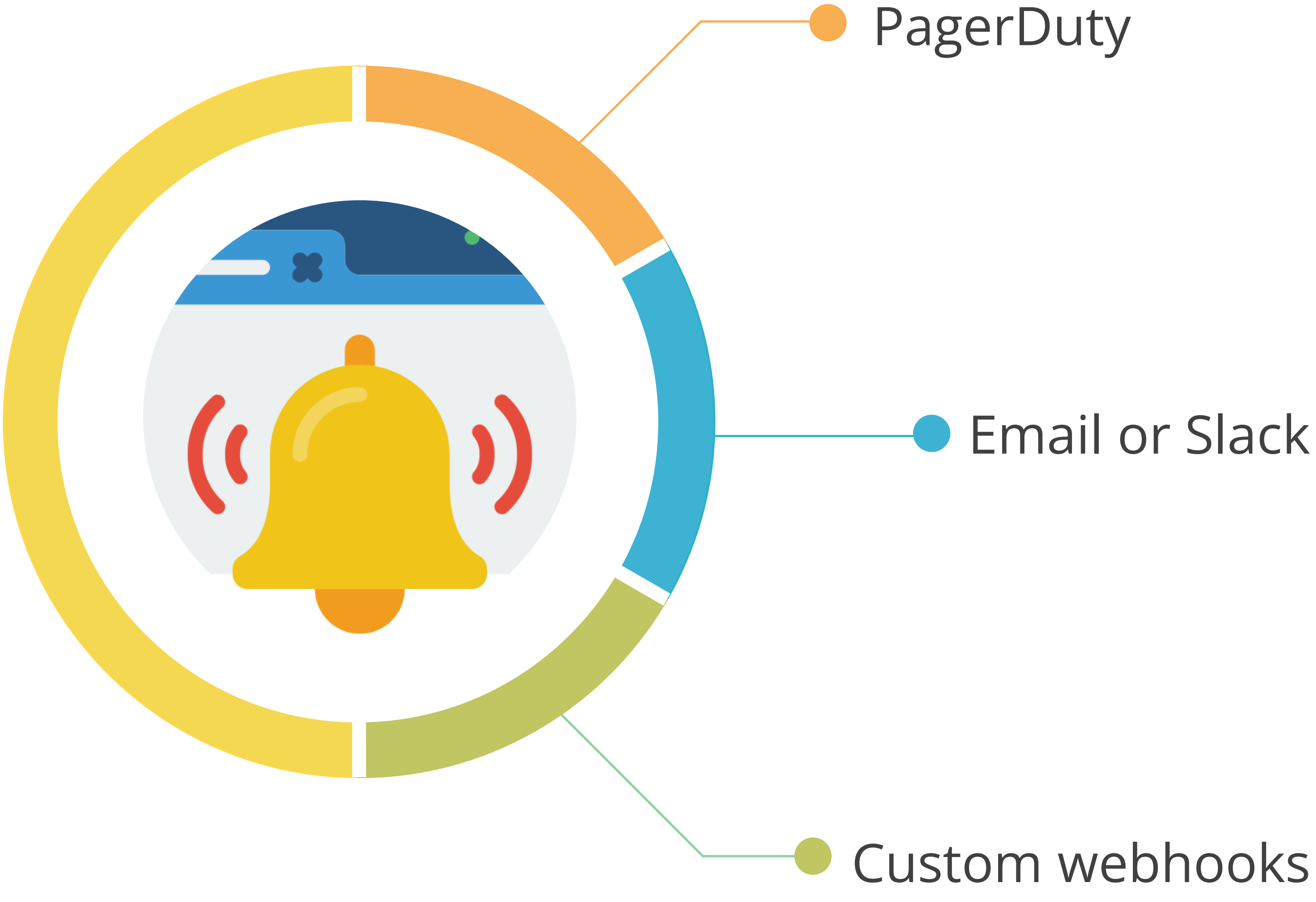
Node exporter
(for system metrics)

cAdvisor
(for container metrics)

MySQL exporter
(for MySQL metrics)

Prometheus Alerting

Alertmanager manages alert routing and notification delivery by receiving alerts from Prometheus and sending them through the following channels:



- PagerDuty
- Email or Slack
- Custom webhooks

Prometheus Query Language (PromQL)

It is a query language designed to extract, manipulate, and analyze metrics data within Prometheus. Some of its main features are:

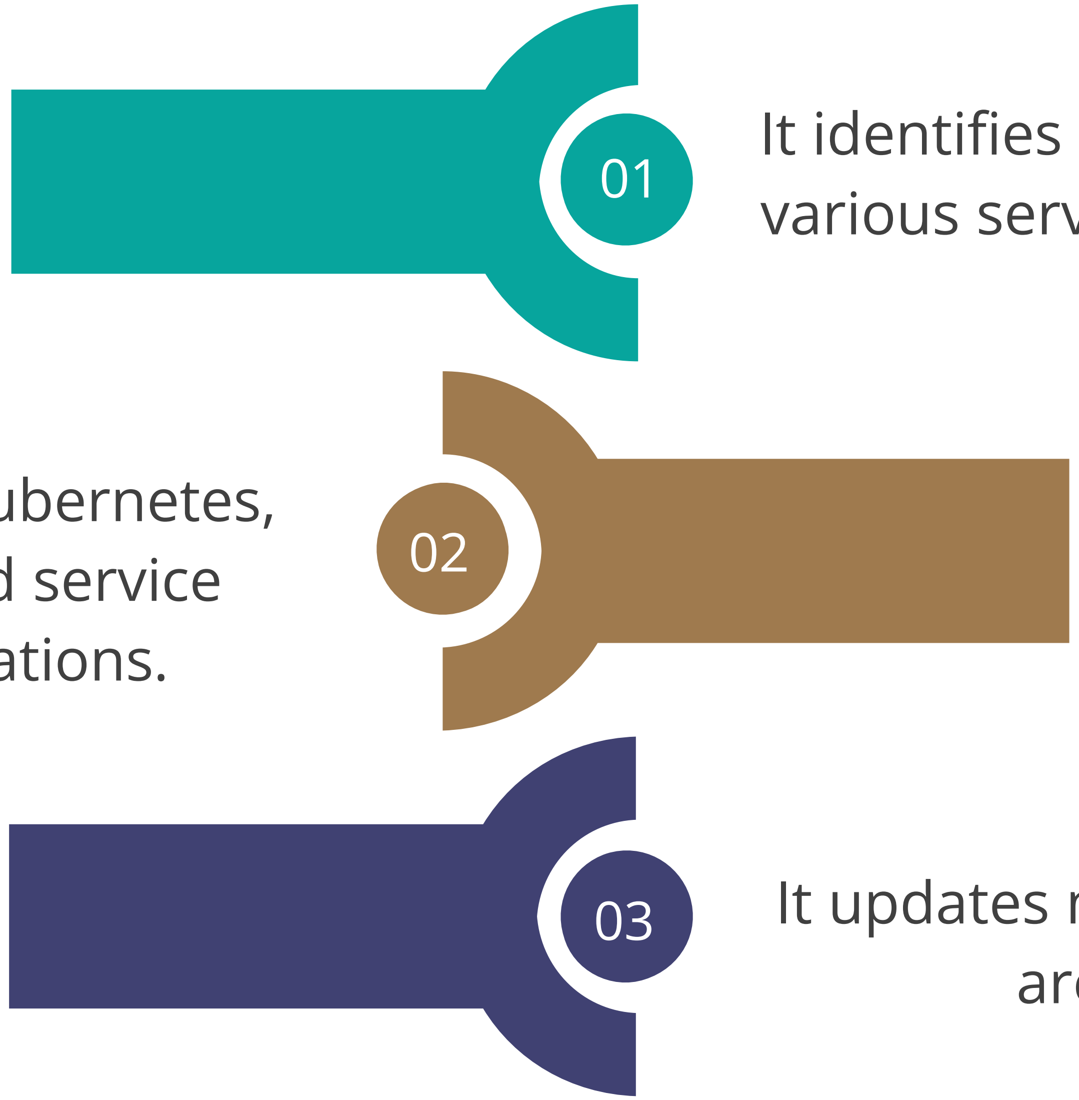
It filters and groups metrics based on labels, enabling precise queries and customized monitoring views.

It allows users to visualize the time series data stored in Prometheus.

It executes queries directly in the Prometheus web UI and integrates with visualization tools like Grafana.

Service Discovery

It is a mechanism that dynamically identifies and monitors the infrastructure, ensuring that the monitoring setup adapts as the environment evolves. Some of its important functions include:



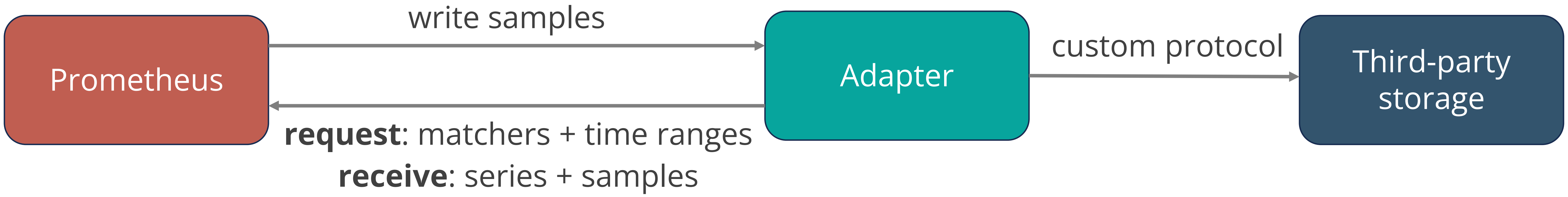
01 It identifies and monitors targets using various service discovery mechanisms.

02 It integrates with systems like Kubernetes, Amazon ECS or uses file-based service discovery for static configurations.

03 It updates monitoring targets as services are added or removed.

Remote Write and Remote Read

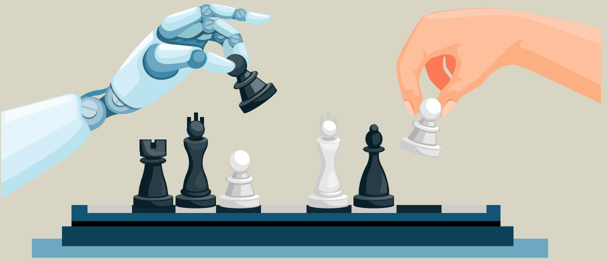
These functionalities allow Prometheus to integrate with third-party storage systems using an adapter.


Prometheus
write samples
Adapter
custom protocol
Third-party storage
request: matchers + time ranges
receive: series + samples

The adapter acts as the bridge that allows Prometheus to seamlessly write data to remote storage and later retrieve it, ensuring scalability, high availability, and long-term data retention.

Case Study

Enhanced Web Application Monitoring with Prometheus



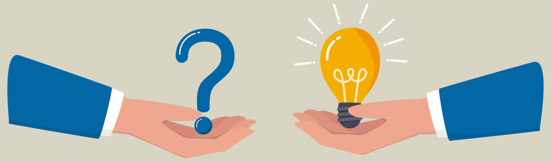
Problem

Company ABC has a web application that operates on multiple servers (instances). Recently, users have reported slow response times and occasional errors when using the application.

The challenge lies in identifying which specific server instance is causing these issues and understanding the overall performance of the application.

Case Study

Enhanced Web Application Monitoring with Prometheus



Solution

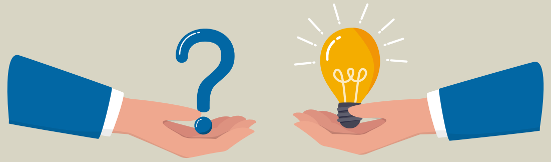
- Implement Prometheus as the monitoring tool
- Treat each server instance within the web application as an individual ‘instance’ under a common ‘job’ named ‘web_application’

Prometheus collects key metrics, such as:

- **http_requests_total:** Tracks the total number of HTTP requests received by each server
- **http_response_time_seconds:** Records the response time for each HTTP request

Case Study

Enhanced Web Application Monitoring with Prometheus



Solution

The metrics are accompanied by labels:

- **instance:** Identifies the server instance providing the metric
- **status_code:** Represents the HTTP status codes (for example, 200 for success, 500 for errors)

Analysis from Prometheus:

- Filter metrics by the ‘instance’ label to identify performance issues on specific servers
- Combine metrics across all instances within the ‘web_application’ job to gain a comprehensive view of overall application performance

Case Study

Enhanced Web Application Monitoring with Prometheus

Outcome

By organizing, querying, and analyzing the metrics, performance bottlenecks and problematic server instances were identified.

For example, if one server instance was handling more requests or showing higher response times, corrective actions such as adjusting load balancing or scaling server resources were performed.

Quick Check

A development team is using Prometheus to monitor their distributed applications. They want to aggregate metrics from multiple Prometheus servers to get a unified view of their system's performance. Which Prometheus feature should they use?

A. Scrape

B. Labels

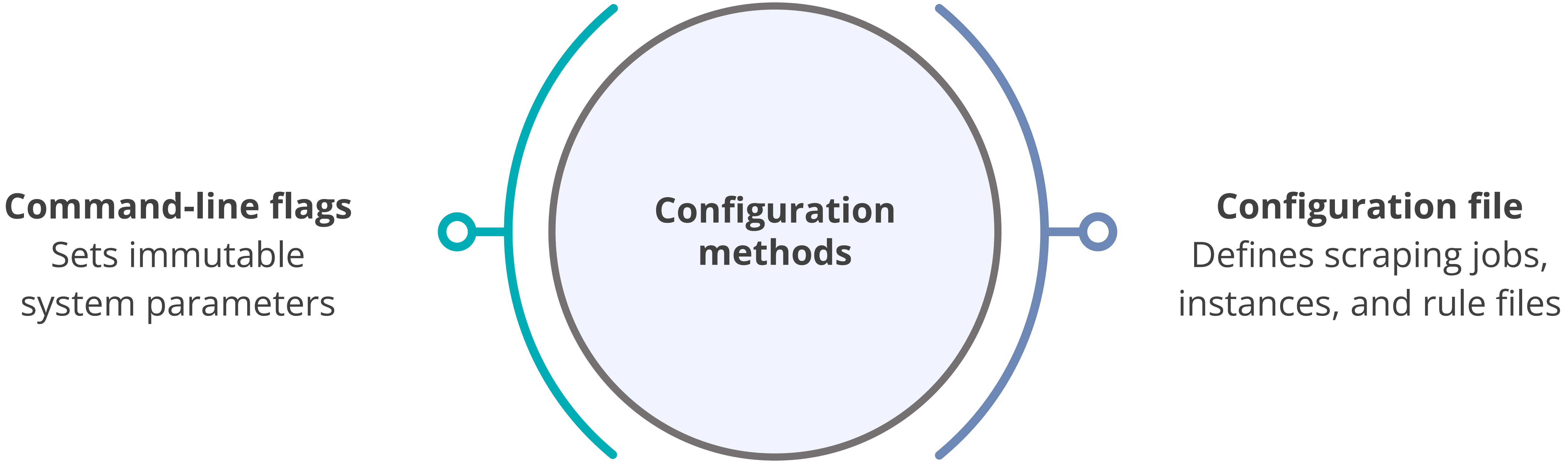
C. Federation

D. Job

Getting Started with Prometheus

Prometheus Configuration

It refers to the overall setup and parameters that define how Prometheus operates. This includes the following methods:



Configuration methods

Command-line flags
Sets immutable system parameters

Configuration file
Defines scraping jobs, instances, and rule files

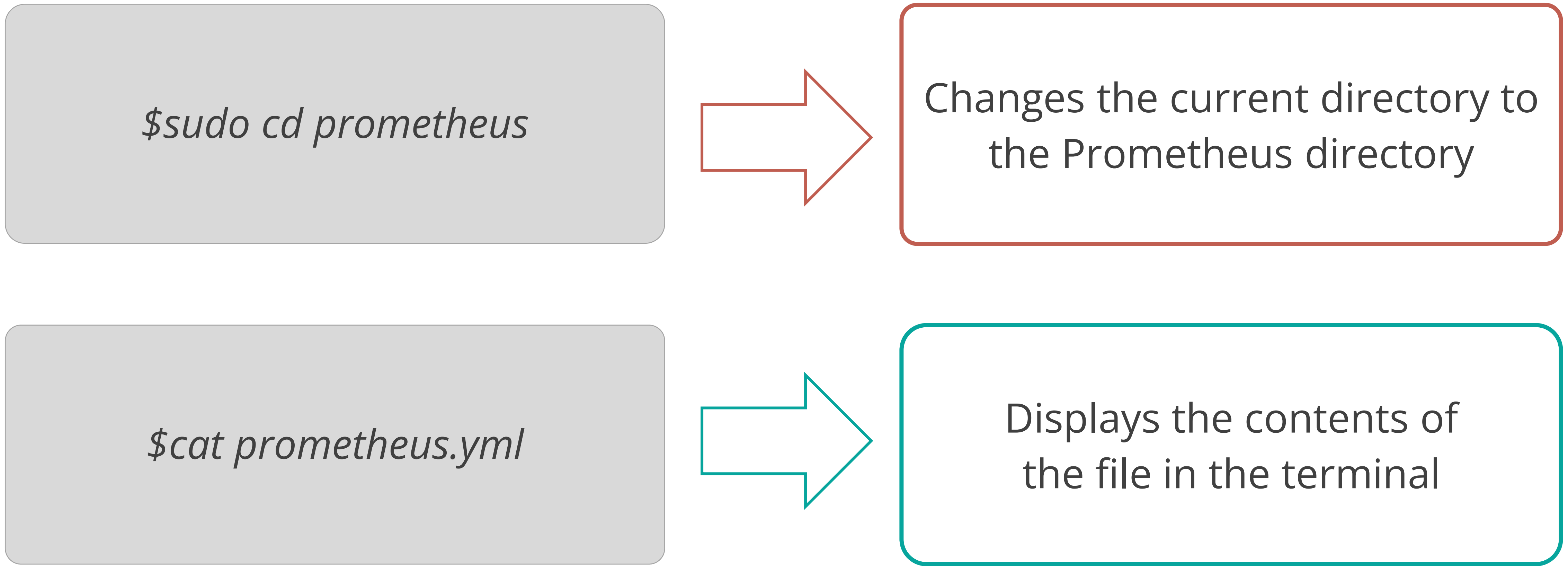
NOTE

YAML stands for YAML Ain't Markup Language. It is used for configuration files. YML and YAML refer to the same language, and **.yaml** is the standard file extension.

Prometheus Configuration File

It is a YML file used to define the configurations and parameters for Prometheus, including data sources, scrape intervals, and alerting rules.

To view and access the configuration file, run the following commands:



```
$sudo cd prometheus
```

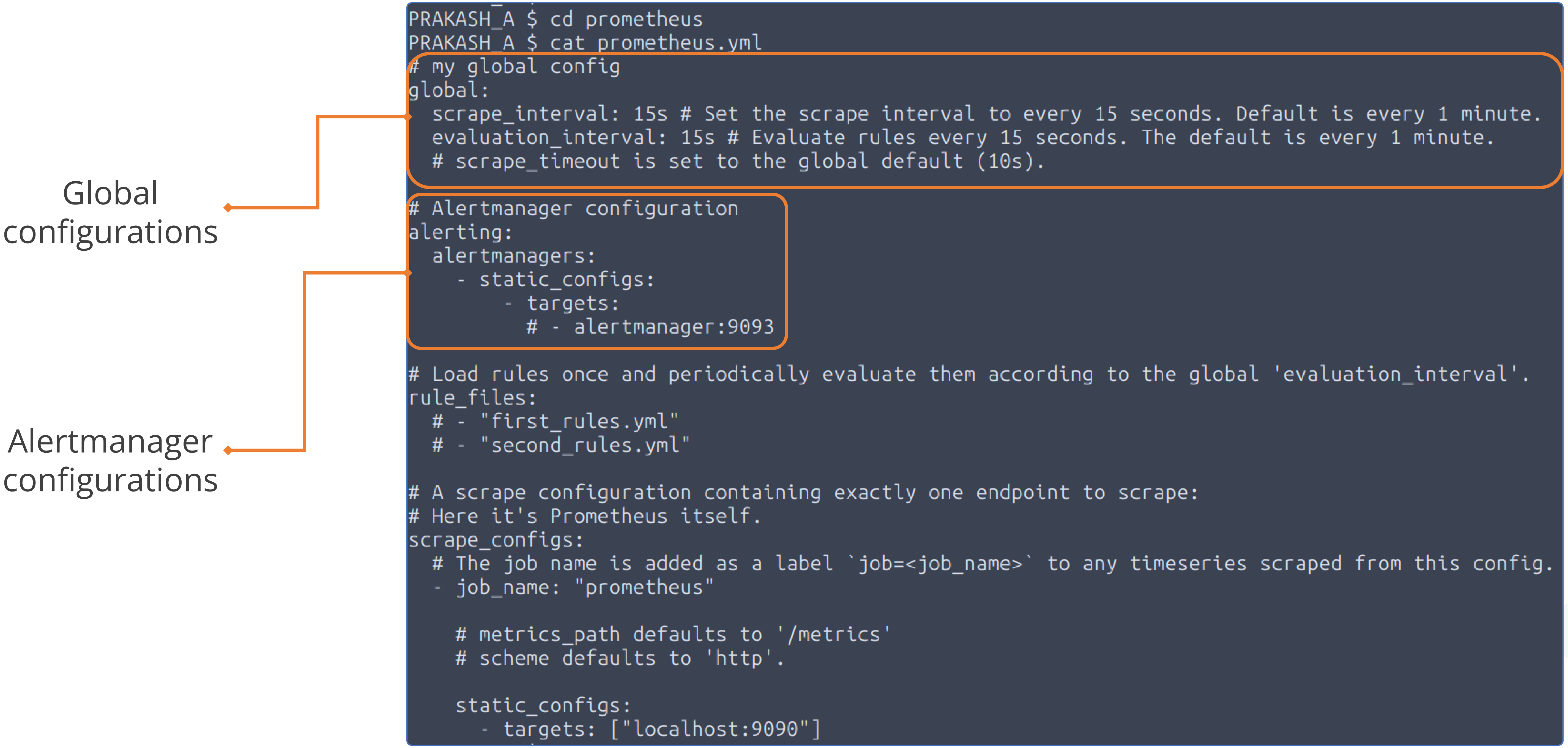
Changes the current directory to the Prometheus directory

```
$cat prometheus.yml
```

Displays the contents of the file in the terminal

Prometheus Configuration File

A standard **prometheus.yml** configuration file includes the following configurations:



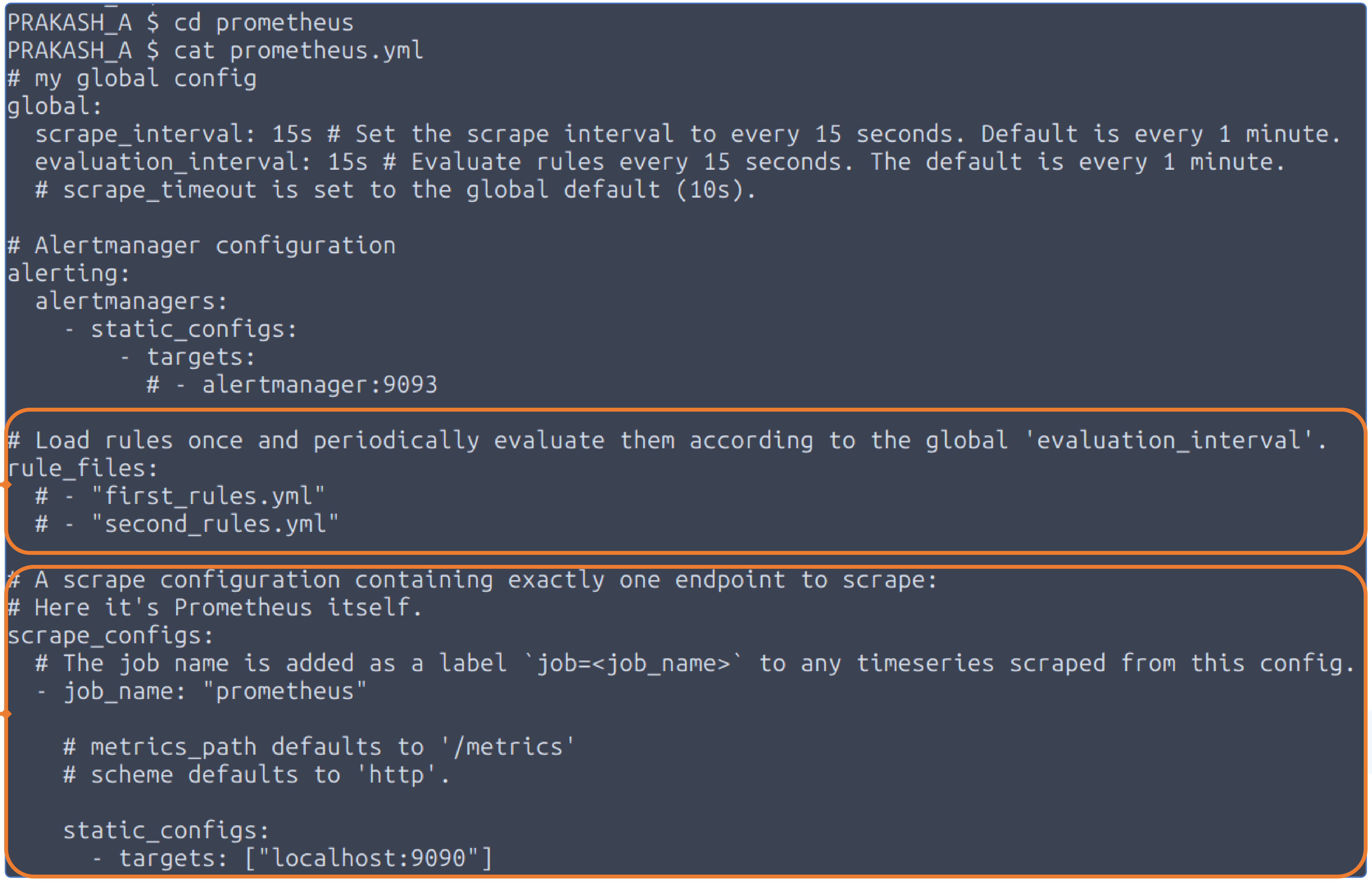
Global configurations

Alertmanager configurations

```
PRAKASH_A $ cd prometheus
PRAKASH_A $ cat prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
```

Prometheus Configuration File

A standard **prometheus.yml** configuration file includes the following configurations:



```
PRAKASH_A $ cd prometheus
PRAKASH_A $ cat prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
```

Rule files

Scrape configurations

Basic UI Elements of Prometheus Server

It involves identifying the components and functionalities of the Prometheus monitoring tool's user interface. This includes the following key elements:

Alert
Manages and reviews active alerts

Graph
Visualizes and analyzes metrics over time

Status
Checks runtime details, TSDB status, configuration, rules, targets, and service discovery

Help
Accesses documentation and supports resources

Assisted Practice

Configuring Prometheus to Scrape and Visualize the Metrics

Duration: 10 Min.

Problem statement:

You have been assigned a task to configure Prometheus to scrape and visualize metrics for monitoring system performance through its web interface for improved observability and real-time insights.

Outcome:

By the end of this demo, you will be able to configure Prometheus to scrape and visualize metrics for monitoring system performance through its web interface.

Note: Refer to the demo document for detailed steps:
02_Setting_up_a_Local_Prometheus_Instance_and_Exploring_Its_Web_Interface

Assisted Practice: Guidelines

Steps to be followed:

1. Start Prometheus binary
2. Explore Prometheus UI

Quick Check

As a DevOps engineer, you are responsible for setting up Prometheus to monitor a web server. You need to configure how frequently Prometheus scrapes metrics from the server. In which configuration file and section can you specify the scrape interval, and what is the purpose of this setting?

A. In the alertmanager.yml file; to define how frequently alerts are sent

B. In the global configuration section of the prometheus.yml file; to set the scrape interval for collecting metrics from targets

C. In the rules.yml file; to specify how often Prometheus should evaluate rules

D. In the webserver.yml file; to adjust web server logging frequency

Key Takeaways

- Continuous monitoring is crucial for maintaining system performance and ensuring continuous feedback.
- Key components of a monitoring system include agents, data collection, storage, processing, visualization, and alerting.
- Metrics, alerts, and dashboards are essential for tracking performance, identifying issues, and visualizing data to support decision-making.
- Prometheus is effective in time-series data collection and integrates well with various exporters and visualization tools.
- Prometheus UI enables users to execute PromQL queries, visualize metrics, and monitor system performance effectively.

Running Prometheus as a Docker Container for Monitoring

Duration: 25 Min.

Project agenda: To demonstrate the setup and running of Prometheus as a Docker container for monitoring, enabling efficient tracking and management of system metrics

Description: You are a developer in a software company responsible for configuring and deploying Prometheus within a Docker container for real-time monitoring of system metrics. You begin by setting up the Docker environment, pulling the Prometheus image, and ensuring it is correctly configured for your infrastructure. After running the container, you verify that Prometheus efficiently collects and displays metrics.

Running Prometheus as a Docker Container for Monitoring

Duration: 25 Min.

Perform the following:

1. Pull and set up Docker environment
2. Create and edit Prometheus configuration
3. Start the Prometheus container
4. Access the Prometheus web interface

Expected deliverables: A fully configured Prometheus Docker container setup to scrape and visualize system metrics, accessible through a web interface for real-time monitoring and infrastructure management

LESSON-END PROJECT

Thank You