Monitoring Stack

Prometheus, Alert Manager, Grafana

Why We need Monitoring

1. Real-Time Monitoring of System Health

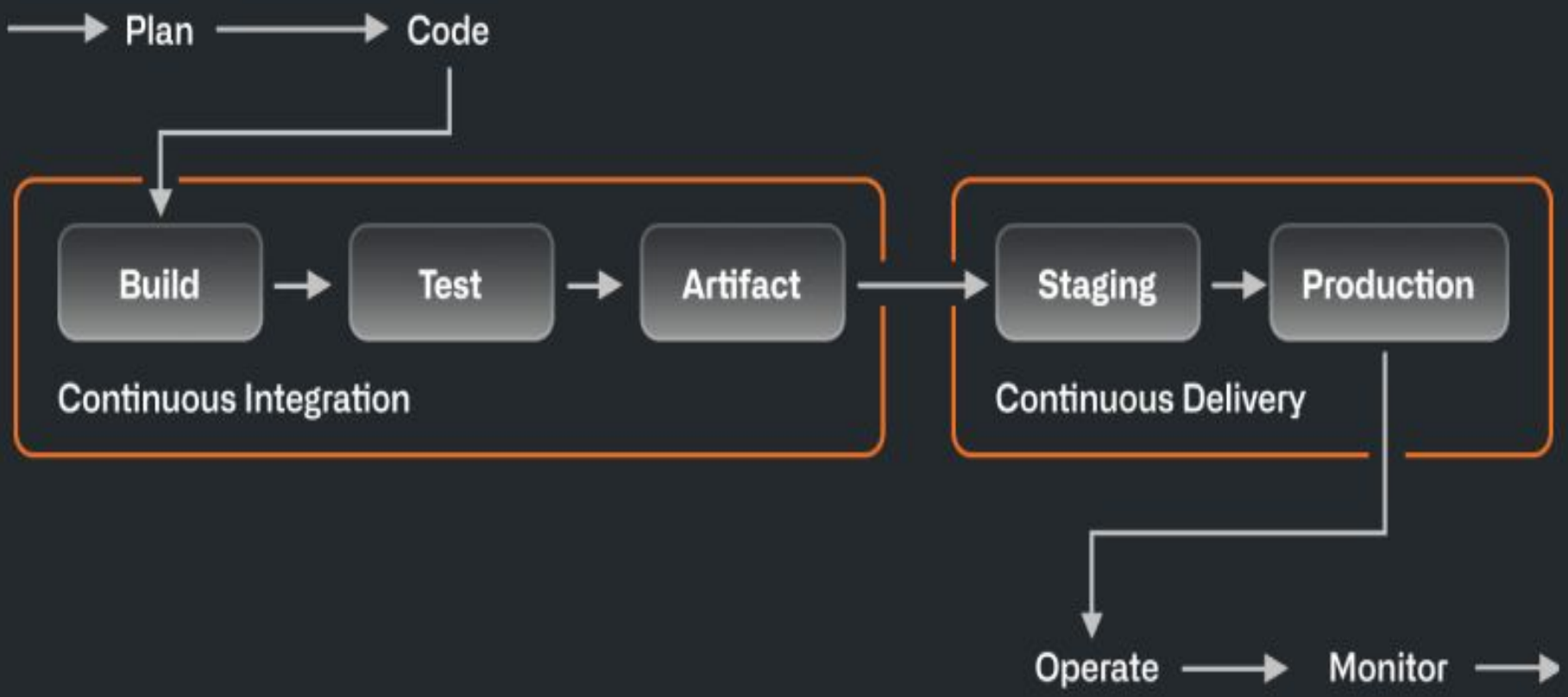
- **Why**: In today's complex, distributed systems, ensuring the health of servers, containers, applications, and other components is critical to avoid downtime.
- **Benefit**: Prometheus provides real-time monitoring, allowing teams to track metrics continuously and understand how systems are performing at any given moment.

Automated Alerting on Critical Issues

- **Why**: Constantly monitoring dashboards manually is impractical. Automated alerts notify teams when something goes wrong.
- **Benefit**: Alertmanager works with Prometheus to automatically trigger alerts for predefined conditions, such as high CPU usage or down services, so teams can quickly respond to issues.

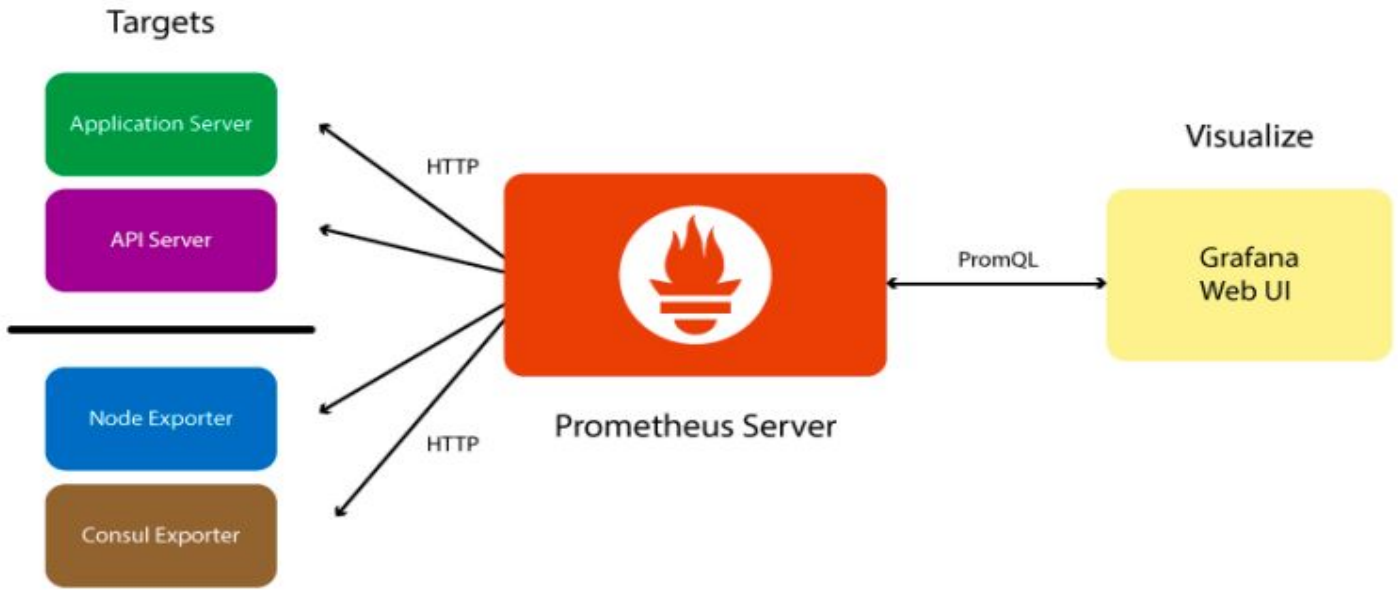
Enhanced System Reliability and Faster Incident Response

- **Why**: Monitoring alone isn't enough; teams also need to identify and address issues quickly to reduce downtime.
- **Benefit**: With timely alerts and real-time insights from dashboards, teams can diagnose and resolve issues faster, resulting in higher system availability and reliability.


Plan
Code
Build
Test
Artifact
Continuous Integration
Staging
Production
Continuous Delivery
Operate
Monitor

- **Prometheus**: Data collection and monitoring
- **Alertmanager**: Alert routing and management
- **Grafana**: Visualization and dashboarding

Architecture Overview-



Prometheus-

Prometheus main features A multidimensional data model with time series data identified by metric name and key/value pairs

A flexible query language to leverage this dimensionality

Time series collection happens via a pull model over HTTP

Targets are discovered via service discovery or static configuration

Data Collection: Uses exporters to scrape metrics from various systems.

Data Flow: Exporters -> Prometheus -> Storage -> Alerts/Queries

Prometheus monitoring

In Prometheus terms, the main monitoring service is referred to as the Prometheus Server and the services Prometheus monitors are called targets. A target can be a host, a network equipment or a specific service.

Typically, the Prometheus server/daemon collects metrics from targets by making HTTP[s] requests to the Prometheus exporters. In Prometheus terms, that process is called scarping.

Prometheus exporters

An Exporter is a piece of software that fetches metrics from a given system and exports them in a format that the Prometheus server can understand.

There are a number of libraries that can be used to write custom exporters but there are also many existing FOSS exporters, written and maintained by the Prometheus team, third party vendors and community members.

Prometheus AlertManager

The Prometheus Alertmanager handles alerts sent by client applications such as the Prometheus server.

It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie.

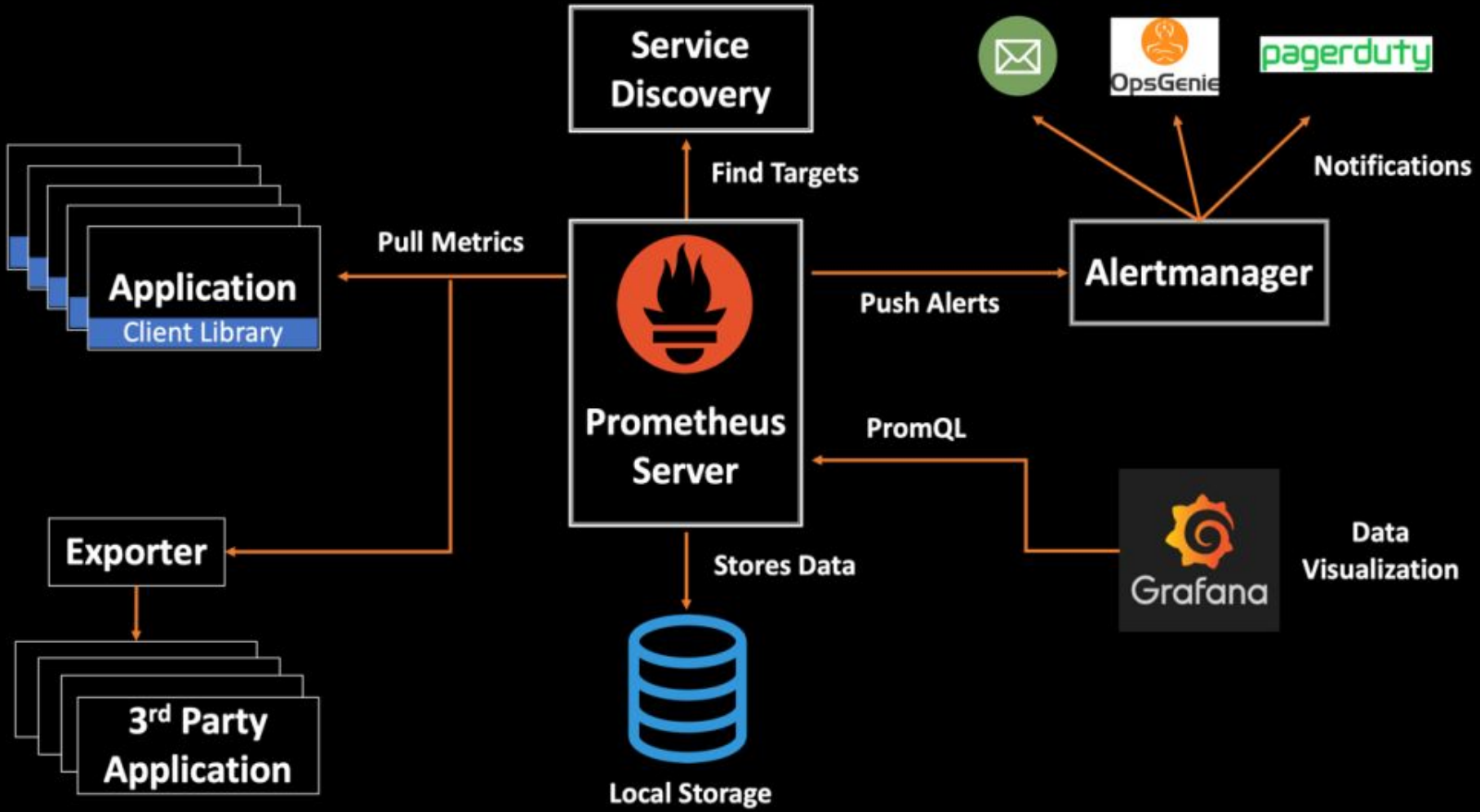
AlertManager is also capable of silencing and inhibition of alerts. In this context, inhibition means suppressing notifications for certain alerts if certain other alerts are already firing.

Grafana reporting and data visualisation

Grafana is an open source platform for analytics and monitoring.

It allows you to query, visualise, alert on and understand your metrics.

Grafana can connect to many different data sources including, though not limited to: MySQL, ElasticSearch and of course, Prometheus:)


Service Discovery
Find Targets
Notifications
OpsGenie
pagerduty
Application
Client Library
Pull Metrics
Prometheus Server
Push Alerts
Alertmanager
PromQL
Exporter
Stores Data
Grafana
Data Visualization
3rd Party Application
Local Storage

AWS Prometheus Cluster Creation

Repo name-

https://github.com/ankit20000/PROM-AMP

Node Exporter installation

wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz

tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz

sudo mv node_exporter-1.6.1.linux-amd64/node_exporter /usr/local/bin/

sudo vim /etc/systemd/system/node_exporter.service

copy and paste-

Node Exporter installation

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target
StartLimitIntervalSec=500
StartLimitBurst=5
[Service]
#User=node_exporter
#Group=node_exporter
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/node_exporter \
   --collector.logind
[Install]
WantedBy=multi-user.target
```

4-systemctl daemon-reload
5-ss
6-sudo systemctl start node_exporter
7-sudo systemctl status node_exporter

Prometheus installation

```
wget
https://github.com/prometheus/prometheus/releases/download
/v2.45.0/prometheus-2.45.0.linux-amd64.tar.gz
 tar -xvf prometheus-2.45.0.linux-amd64.tar.gz
  sudo mkdir -p /data /etc/prometheus
  cd prometheus-2.45.0.linux-amd64/
   sudo mv prometheus promtool /usr/local/bin/
   sudo mv consoles/ console_libraries/ /etc/prometheus/
   sudo mv prometheus.yml /etc/prometheus/prometheus.yml
  sudo vim /etc/systemd/system/prometheus.service
```

Prometheus installation

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/data \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries \
  --web.listen-address=0.0.0.0:9090 \
  --web.enable-lifecycle

[Install]
WantedBy=multi-user.target
```

Prometheus installation

```
sudo systemctl enable prometheus
sudo systemctl start prometheus
sudo systemctl status prometheus
cd /etc/prometheus/
nano prometheus.yml
```

copy and paste-

```yaml
global:
  scrape_interval: 15s
  external_labels:
    monitor: 'prometheus'
scrape_configs:
  # 🔹 Scrape Prometheus server itself
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  # 🔹 Scrape EC2 instances with tag: prometheus=yes
  - job_name: 'ec2-node-exporter'
    ec2_sd_configs:
      - region: "ap-south-1"
        filters:
          - name: tag:prometheus
            values:
              - yes
    relabel_configs:
      # Use EC2 private IP and append :9100
      - source_labels: [__meta_ec2_private_ip]
        regex: (.*)
        target_label: __address__
        replacement: ${1}:9100
      # Set instance label to the IP:port for use in alerts
      - source_labels: [__address__]
        target_label: instance
remote_write:
  - url: https://aps-workspaces.ap-south-1.amazonaws.com/workspaces/ws-1bfe4e50-4961-41b1-9a09-04701ef4e7e0/api/v1/remote_write
    queue_config:
      max_samples_per_send: 1000
      max_shards: 200
      capacity: 2500
    sigv4:
      region: ap-south-1
```