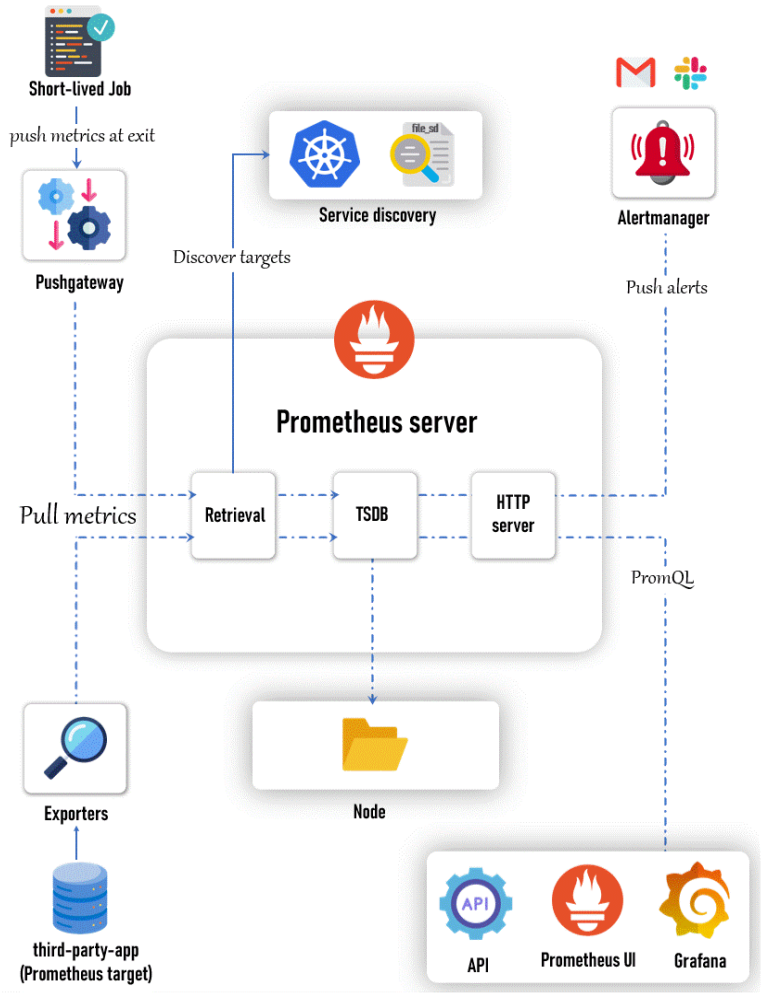
Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud.

It is known for its robust data model, powerful query language (PromQL), and the ability to generate alerts based on the collected time-series data.It can be configured and set up on both bare-metal servers and container environments, such as Kubernetes.

Architecture

The architecture of Prometheus is designed to be highly flexible, scalable, and modular.It consists of several core components, each responsible for a specific aspect of the monitoring process.



🔥 Prometheus Server

Prometheus server is the core of the monitoring system. It is responsible for scraping metrics from various configured targets, storing them in its time-series database (TSDB), and serving queries through its HTTP API.

Components:

- **Retrieval:** This module handles the scraping of metrics from endpoints, which are discovered either through static configurations or dynamic service discovery methods.
- **TSDB (Time Series Database):** The data scraped from targets is stored in the TSDB, which is designed to handle high volumes of time-series data efficiently.
- **HTTP Server:** This provides an API for querying data using PromQL, retrieving metadata, and interacting with other components of the Prometheus ecosystem.
- **Storage:** The scraped data is stored on local disk (HDD/SSD) in a format optimized for time-series data.

🌐 Service Discovery

Service discovery automatically identifies and manages the list of scrape targets (i.e., services or applications) that Prometheus monitors.

This is crucial in dynamic environments like Kubernetes where services are constantly being created and destroyed.

Components:

- **Kubernetes:** In Kubernetes environments, Prometheus can automatically discover services, pods, and nodes using Kubernetes API, ensuring it monitors the most up-to-date list of targets.
- File SD (Service Discovery): Prometheus can also read static target configurations from files, allowing for flexibility in environments where dynamic service discovery is not used.

📤 Pushgateway

The Pushgateway is used to expose metrics from short-lived jobs or applications that cannot be scraped directly by Prometheus.

These jobs push their metrics to the Pushgateway, which then makes them available for Prometheus to scrape(pull).

Use Case:

It's particularly useful for batch jobs or tasks that have a limited lifespan and would otherwise not have their metrics collected.

🚨 Alertmanager

The Alertmanager is responsible for managing alerts generated by the Prometheus server.

It takes care of deduplicating, grouping, and routing alerts to the appropriate notification channels such as PagerDuty, email, or Slack.

🧲 Exporters

Exporters are small applications that collect metrics from various third-party systems and expose them in a format Prometheus can scrape. They are essential for monitoring systems that do not natively support Prometheus.

Types of Exporters:

Common exporters include the Node Exporter (for hardware metrics), the MySQL Exporter (for database metrics), and various other application-specific exporters.

🖥️ Prometheus Web UI

The Prometheus Web UI allows users to explore the collected metrics data, run ad-hoc PromQL queries, and visualize the results directly within Prometheus.

📊 Grafana

Grafana is a powerful dashboard and visualization tool that integrates with Prometheus to provide rich, customizable visualizations of the metrics data.

🔌 API Clients

API clients interact with Prometheus through its HTTP API to fetch data, query metrics, and integrate Prometheus with other systems or custom applications.

🛠️ Installation & Configurations

1. Create EKS Cluster
2. Install kube-prometheus-stack
3. Deploy the chart into a new namespace "monitoring"
4. Verify the Installation
5. Clean UP

Step 1.Create EKS Cluster

Prerequisites

- **Download and Install AWS Cli - Please Refer this link.**
- **Setup and configure AWS CLI using the `aws configure` command.**
- **Install and configure eksctl using the steps mentioned here.**
- **Install and configure kubectl as mentioned here.**

```
eksctl create cluster --name=observability \
                --region=us-east-1 \
                --zones=us-east-1a,us-east-1b \
                --without-nodegroup
eksctl utils associate-iam-oidc-provider \
   --region us-east-1 \
   --cluster observability \
   --approve
```

```
eksctl create nodegroup --cluster=observability \
                        --region=us-east-1 \
                        --name=observability-ng-private \
                        --node-type=t3.medium \
                        --nodes-min=2 \
                        --nodes-max=3 \
                        --node-volume-size=20 \
                        --managed \
                        --asg-access \
                        --external-dns-access \
                        --full-ecr-access \
                        --appmesh-access \
                        --alb-ingress-access \
                        --node-private-networking


# Update ./kube/config file

aws eks update-kubeconfig --name observability
```

Step 2: Install kube-prometheus-stack

```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

helm repo update
```

Step 3: Deploy the chart into a new namespace "monitoring"

```
kubectl create ns monitoring
```

```
helm install monitoring prometheus-community/kube-prometheus-stack \

-n monitoring \

-f ./custom_kube_prometheus_stack.yml
```

Step 4: Verify the Installation

```
kubectl get all -n monitoring
```

- Prometheus UI:

```
kubectl port-forward service/prometheus-operated -n monitoring 9090:9090
```

NOTE: If you are using an EC2 Instance or Cloud VM, you need to pass `--address 0.0.0.0` to the above command. Then you can access the UI on instance-ip:port

- Grafana UI: password is `prom-operator`

```
kubectl port-forward service/monitoring-grafana -n monitoring 8080:80
```

- Alertmanager UI:

```
kubectl port-forward service/alertmanager-operated -n monitoring 9093:9093
```

Step 5: Clean UP

- Uninstall helm chart: `helm uninstall monitoring --namespace monitoring`
- Delete namespace: `kubectl delete ns monitoring`
- Delete Cluster & everything else: `eksctl delete cluster --name observability`