Signal Type	Description	Tools Involved
Metrics	Numeric data representing system state over time (e.g., CPU usage, request count)	Prometheus, Grafana Alloy, Grafana, and OpenTelemetry
Logs	Textual data that records discrete events (e.g., errors, messages, output logs)	Grafana Loki, Promtail, and OpenTelemetry
Traces	Detailed, end-to-end request journey through systems and services	OpenTelemetry, Grafana Alloy, and Grafana Tempo

Signal	Format	Description
Metric	Time-Series	Includes: Time Stamp, Metrc Name, and Metric Value, e.g., CPU usage at time x.
Logs	Json	Example: {“app”:”shoeHub”, “message”:” Connected to

		database”}
Traces	Binary	OpenTelemetry-specific binary format.

Core Components in Observability Stack

Prometheus

Feature	Description
Scrapes Metrics	Periodically pulls metrics from targets that expose a Prometheus-compatible API.
Store Time Series	Stores time-series data in memory + local disk
Alerting Rules	Evaluates user-defined conditions to trigger alerts

Prometheus Exporters

Exporter	Description	Platform	Signal Type
Node Exporter	Exposes Linux/macOS/Unix system metrics	Unix-like systems	Metrics

WMI Exporter	Exposes Windows performance metrics	Windows	Metrics

Prometheus PushGateway

Feature	Description	Signal Type
Metric Receiver	Accepts pushed metrics from short-lived jobs	Metrics
Stores Temporarily	Holds metrics until Prometheus scrapes them	Metrics

Grafana (Dashboards & Alerts)

Feature	Description	Signal Type
Dashboards	Visualize metrics/logs/traces using panels	Metrics, Logs, Traces
Alerting	Built-in alert engine for Grafana alerts	Metrics
Notification Policies	Route alerts via Slack, email, etc.	Alerts

Main Grafana Data Sources

Data Source	Description	Signal Type
Prometheus	Time-series metric source	Metrics
Loki	Label-based log query engine	Logs
Tempo (via Alloy)	Trace visualization	Traces

Deployment & Setup

Component	Platform(s) Supported	Installation Type
Prometheus	Windows, macOS, Ubuntu, Docker	Binary or Docker
Node Exporter	Linux, macOS	Binary / Systemd Service
WMI Exporter	Windows	MSI Installer
Grafana	Windows, macOS, Linux, Docker	Binary / Package Manager / Docker
Grafana Loki	Linux, Mac, and Docker	Binary / Docker

Promtail	Linux, Mac, and Docker	Binary / Docker
Grafana Alloy	Linux, Mac, and Docker	Binary / Docker

Ports and Connections

Prometheus

Purpose	Port	Notes
Web UI / API	`9090`	Access metrics, rules, targets, and query UI
Remote Write / Read	`9090`	Same endpoint as UI

Grafana

Purpose	Port	Notes
Web UI	`3000`	Default admin UI port
Alerts & Plugin APIs	`3000`	The same port is used for all services

Grafana Alloy (Open Telemetry Signal Collector)

Purpose	Port	Notes
OTLP HTTP Receiver	`4318`	For metrics, traces via HTTP
OTLP gRPC Receiver	`4317`	For metrics, traces via gRPC
Prometheus Receiver	`8888`	Prometheus scraping port (if enabled)
Health Check	`13133`	Optional health endpoint

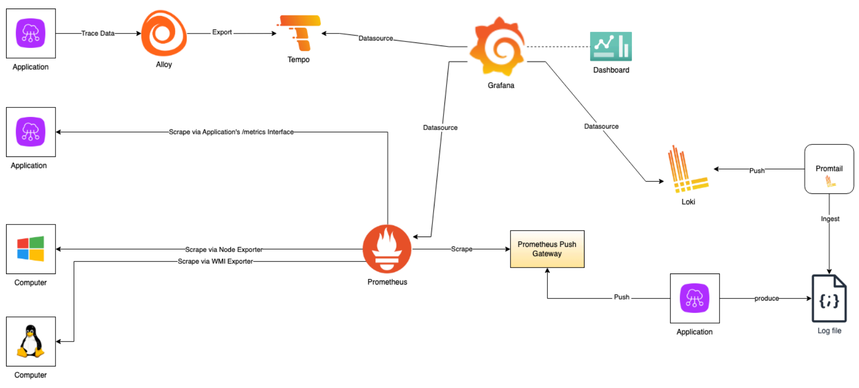
Grafana Loki (Logs)

Purpose	Port	Notes
HTTP API / Ingestion	`3100`	All Loki HTTP endpoints (push, query, config)

Grafana Tempo (OpenTelemetry Traces)

Purpose	Port	Notes

HTTP API	`3200`	For Tempo UI, metrics generator, etc.
gRPC	`4317`	OTLP gRPC endpoint
OTLP HTTP	`4318`	OTLP HTTP endpoint



System	Push or Pull	Data Type	To or From
Application	Neither	Metrics	Neither
Application	Push	Metrics	To Push Gateway
Application	Push	OTel Traces	Alloy
Application	Push	OTel Metrics	Alloy

Application	Push	OTel Logs	Alloy
Alloy	Push	OTel Signals	To Tempo
Computers	Neither	Infra. Signals	Neither
Prometheus	Pull	Metrics	To Applications and Computers
Prometheus Push Gateway	Push	Metrics	To Prometheus
Application	Push	Log Files	File System
Promtail	Pull	Log Files	File System
Promtail	Push	Log Files	Loki
Grafana	Pull	Metrics	Prometheus
Grafana	Pul	OTel Traces	Tempo
Grafana	Pull	Logs	Loki

Quick Setups

Either of the following approaches can set up the Grafana Stack, Prometheus, and Sample signal data quickly:

1. Grafana
2. Prometeus
3. Grafana Loki
4. Grafana Loki Promtail
5. Grafana Alloy
6. Grafana Tempo
7. ShoeHub for sample metrics

8. Payment Service and Order Service for OTel traces

Approach	Temporary or Permanent	Guide
Docker	Permanent	On Linux or Mac: - Clone https://github.com/aussieparef/grafana-udemy.git - Open Terminal - Go to “docker” directory. - Run docker-compose.sh file.
Killer Coda	Temporary	In a Browser: - Visit https://killercoda.com/aref-karimi/course/grafana - Sign in - Follow the instructions - Open Grafana or Prometheus. The environment lasts for 60 minutes.

Manual Setups

Setup Prometheus

On Mac, Windows, or Linux Client:

1. Download the binary file relevant to your OS from https://prometheus.io/download/

2. Install the binary file.
3. Search for and find "prometheus.yml" file.

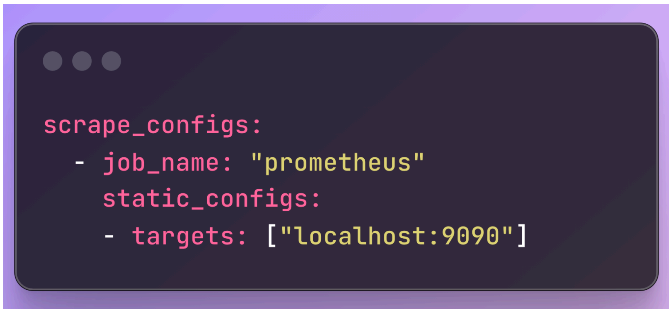
Or, if you use a Mac:

1. Download Homebrew from https://brew.sh
2. Run "brew install prometheus"
3. In Terminal, run "brew --prefix prometheus". This command will return the installation directory of Prometheus.
4. The returned directory is where the Prometheus configuration file (prometheus.yml) is.

Or, if you are on a Linux Server, e.g., Ubuntu:

1. In Terminal, run "sudo snap install prometheus --classic"
2. Check Prometheus is running: "sudo snap services"
3. The configuration file (prometheus.yml) must be at "/var/snap/prometheus/current/prometheus.yml"
4. If the configuration file is not in the above directory, run "sudo find / -name prometheus.yml 2>/dev/null"

- Open the prometheus.yml file in a text editor.
- Add targets (for scraping) under "static_configs":



```
scrape_configs:
  - job_name: "prometheus"
    static_configs:
    - targets: ["localhost:9090"]
```

Example prometheus.yml file is HERE.

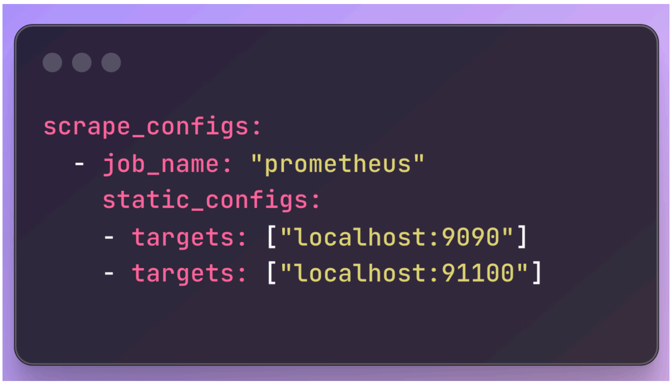
To generate metrics on Linux and Mac, you must install Node Exporter via: https://prometheus.io/download/#node_exporter

To generate metrics on Windows, you must install WMI Exporter via:
https://github.com/prometheus-community/windows_exporter/releases

Update the prometheus.yml file with the exporter's network address, then restart the Prometheus service.

The usual exporter's metrics endpoint is

http://localhost:9100/metrics



```
scrape_configs:
  - job_name: "prometheus"
    static_configs:
    - targets: ["localhost:9090"]
    - targets: ["localhost:91100"]
```

Setup Grafana

1. To install Grafana on Mac OS, follow the instructions here: https://grafana.com/docs/grafana/latest/setup-grafana/installation/mac/
2. To install Grafana on Windows, follow the steps here: https://grafana.com/docs/grafana/latest/setup-grafana/installation/windows/
3. To install Grafana on Linux, follow the steps here:
4. https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/

5. Find “grafana.ini” file.
6. Open “grafana.ini”.
7. Find [Security] section.
8. Change the admin’s password.
9. Restart Grafana or Grafana’s service.
    a. In Mac: `brew services restart grafana`
    b. In Linux: `systemctrl restart grafana`
10. Access Grafana on port 3000

Setup Loki

Loki can only be installed on Linux, e.g., Ubuntu, or via Docker containers, e.g., on Docker Desktop or Kubernetes.

There are two systems to install:

1. Loki
2. Promtail

Follow the instructions here to install Loki:

https://grafana.com/docs/loki/latest/setup/install/docker/#install-with-docker-compose

Configure Promtail to pick up the log file from your desired location. An example Promtail configuration file can be found here:

https://github.com/aussiearef/grafana-udemy/tree/main/loki

Promtail picks up the log files and pushes them to Loki!

```
Clients:

 - url: http://localhost:3100/loki/api/v1/push
```

Restart Promtail's service or container after changing its configuration file:

- Run "Docker ps"
- Take note of Promtail's container.
- Run "Docker ps restart <promtail's container name>"

Setup Alloy

Alloy can be installed on Mac, Windows, Linux, and as Docker on Docker Desktop and Kubernetes.

Installation guide:

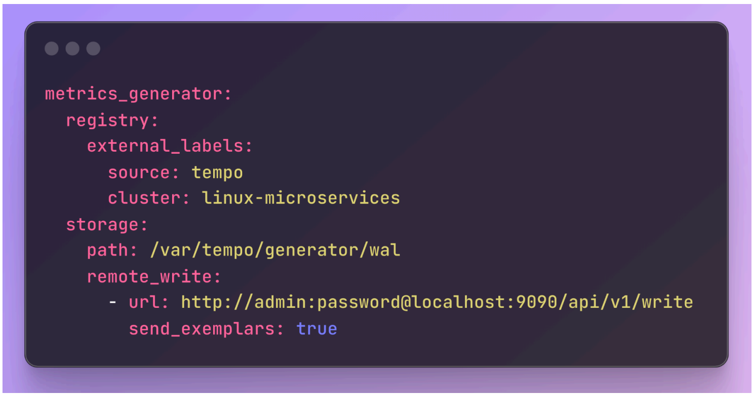
https://grafana.com/docs/alloy/latest/set-up/install/

- Find the config.alloy file.
- Develop the config.alloy file to receive, process, and export Open Telemetry (OTel) signals.
- Save the config.alloy file.
- Restart the Alloy service.

Example config. The alloy file is here:
https://github.com/aussiearef/grafana-udemy/blob/1c5eabdc7dc36334fc7cd0c3f4cddb4e1e2352fd/alloy/config-all-signals.alloy

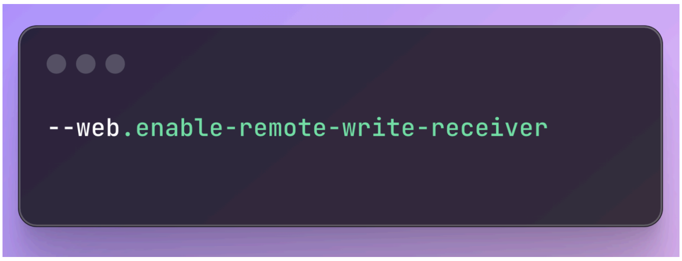
Setup Tempo

- Download the binary file relevant to your OS from here: https://github.com/grafana/tempo/releases
- Run the installation package.
- Find tempo.yml file by searching for it.
- Use this example config file to configure yours: https://github.com/aussiearef/grafana-udemy/blob/1c5eabdc7dc36334fc7cd0c3f4cddb4e1e2352fd/tempo/tempo.yml
- Tempo must push service graph data to Prometheus. Enter Prometheu’s address, with its username and password under “metrics_generator” section:



```yaml
metrics_generator:
  registry:
    external_labels:
      source: tempo
      cluster: linux-microservices
  storage:
    path: /var/tempo/generator/wal
    remote_write:
      - url: http://admin:password@localhost:9090/api/v1/write
        send_exemplars: true
```

You must enable “Service Graph” capability in Prometheus to see the service graph. When starting Prometheus’ executable file, you must pass this switch to it:



```
--web.enable-remote-write-receiver
```

Example:

prometheus --config.file=/etc/prometheus/prometheus.yml --web.enable-remote-write-receiver