# Running Both Kafka Producer Types in the Same Stack

This guide explains how to run both the \*\*kafka-python\*\* and \*\*confluent-kafka\*\* producers against the same Kafka + Prometheus + Grafana stack, and what changes are needed to Prometheus and Grafana to support both.

---

## 1. Prometheus Changes

### If Prometheus runs on the host

Add static scrape configs using `localhost` targets:

```yaml

scrape\_configs:

- job\_name: 'kafka-producers-kpython'

static\_configs:

- targets: ['localhost:9108', 'localhost:9109']

labels:

impl: 'kpython'

- job\_name: 'kafka-producers-ckafka'

static\_configs:

- targets: ['localhost:9118', 'localhost:9119']

labels:

impl: 'ckafka'

```

### If Prometheus runs in Docker and producers run on the host

First, in `docker-compose.yml` for the Prometheus service:

```yaml

services:

prometheus:

extra\_hosts:

- "host.docker.internal:host-gateway"

```

Then, use these targets:

```yaml

scrape\_configs:

- job\_name: 'kafka-producers-kpython'

static\_configs:

- targets: ['host.docker.internal:9108', 'host.docker.internal:9109']

labels: { impl: 'kpython' }

- job\_name: 'kafka-producers-ckafka'

static\_configs:

- targets: ['host.docker.internal:9118', 'host.docker.internal:9119']

labels: { impl: 'ckafka' }

```

(Alternative: put producers in the same Docker network and use container names.)

### Reload Prometheus without restart

If Prometheus started with `--web.enable-lifecycle`:

```bash

curl -X POST http://<prom-host>:9090/-/reload

```

Otherwise, send SIGHUP to the Prometheus PID or restart the container.

---

## 2. Grafana Updates

1. \*\*Add dashboard variables\*\* (Settings → Variables):

- `topic`: `label\_values(kafka\_producer\_message\_rate, topic)`

- `impl`: `label\_values(kafka\_producer\_message\_rate, impl)`

- (optional) `instance`: `label\_values(kafka\_producer\_message\_rate, instance)`

2. \*\*Update panel queries\*\* to include the new `impl` label:

- Per-producer rate:

```

kafka\_producer\_message\_rate{topic="$topic", impl=~"$impl"}

```

- Aggregate per topic:

```

sum by (topic, impl) (kafka\_producer\_message\_rate{topic="$topic", impl=~"$impl"})

```

- Error rate:

```

kafka\_producer\_error\_rate{topic="$topic", impl=~"$impl"}

```

3. \*\*Optional comparison panels\*\*:

- Compare implementations:

```

sum by (impl) (kafka\_producer\_message\_rate{topic="$topic"})

```

4. \*\*Variable filters\*\* allow showing one, both, or all implementations.

---

## 3. Checklist

- Producers export on different ports (e.g., kafka-python: 9108/9109, confluent-kafka: 9118/9119)

- Prometheus shows both target groups as UP

- Grafana variables populate correctly

- Panels can filter or compare by `impl`

---

## Why This Matters

By separating `impl` labels in Prometheus, students can run:

- Only kafka-python

- Only confluent-kafka

- Both at once

They can then compare performance metrics side-by-side in Grafana, which is crucial for tuning and understanding trade-offs between client libraries.