Introducción a Terraform y Salt Project

Unidad 06. Caso práctico 01

Autor: Sergi García

Actualizado Noviembre 2025

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
UNIÓN EUROPEA
Fondo Social Europeo
El FSE invierte en tu futuro
GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura, Universitats i Ocupació
CFR CEFIRE
FORMACIÓ PROFESSIONAL ENSENYANCES ARTÍSTIQUES I ESPORTIVES
FPCV
Formació Professional Comunitat Valenciana

Licencia

Nomenclatura

A lo largo de este tema se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📖 **Importante**

❕ **Atención**

💬 **Interesante**

ÍNDICE

Unidad 06 - Caso práctico 01

🔥 Proyecto: WordPress con monitorización y persistencia usando Terraform + Docker + Prometheus + Grafana

📘 1. Explicación: ¿Qué haremos en este caso práctico?

En este caso práctico construiremos, paso a paso, una infraestructura completa de **servicios web y monitorización**, utilizando **Terraform** como herramienta principal de *Infraestructura como Código (IaC)*.

El objetivo es aprender a desplegar un entorno realista —similar al que se usaría en un entorno de desarrollo o pruebas profesional— combinando **contenedores Docker** con herramientas de observabilidad ampliamente utilizadas en el mundo DevOps.

Concretamente, **Terraform definirá y gestionará** todo lo necesario para levantar un pequeño ecosistema de servicios interconectados:

Todo esto se desplegará **localmente** sobre Docker, pero manteniendo una estructura modular, escalable y reutilizable, exactamente igual a como se haría en un entorno cloud (AWS, Azure, GCP o Kubernetes).

🎯 2. Objetivo

Desplegar una infraestructura **100 % declarativa** con Terraform que levante un entorno local compuesto por:

- 🗃️ **MySQL**, actuando como base de datos del sistema.
- 🌐 **WordPress**, que usará esa base de datos para servir una aplicación web completa.
- ⚙️ **Node Exporter**, encargado de exponer métricas del sistema (uso de CPU, memoria, red, etc.).
- 📊 **Prometheus**, que recopilará y almacenará esas métricas de manera continua.
- 📈 **Grafana**, que se conectará a Prometheus para visualizar la información en paneles dinámicos y personalizables.

Todo conectado mediante una red Docker privada, con volúmenes persistentes y monitorización.

🧱 3. Estructura del proyecto

UD06-CasoPractico01/
├── main.tf
├── variables.tf
├── outputs.tf
├── prometheus.yml
└── dashboards/
└── wordpress-basic.json

🚀 4. Ejecución, Monitorización y Dashboard en Grafana

🚀 Ejecución paso a paso

Abre una terminal en el directorio del proyecto y ejecuta los siguientes comandos:

```
terraform init
```

```
alumno@alumno-virtualbox:~/Desktop/UD06-Caso01$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of kreuzwerker/docker from the dependency lock file
- Using previously-installed kreuzwerker/docker v3.6.2

Terraform has been successfully initialized!
```

```
terraform apply
```

```
alumno@alumno-virtualbox:~/Desktop/UD06-Caso01$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.grafana will be created
  + resource "docker_container" "grafana" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = [
          + "GF_SECURITY_ADMIN_PASSWORD=admin",
          + "GF_SECURITY_ADMIN_USER=admin",
        ]
```

Con estos comandos Terraform:

1. Descargará los providers.
2. Creará la red, volúmenes y contenedores.
3. Montará el archivo prometheus.yml.
4. Mostrará los outputs al finalizar.

Si todo va bien, deberías ver en consola algo así:

```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

wordpress_url = "http://localhost:8080"
prometheus_url = "http://localhost:9090"
grafana_url = "http://localhost:3000"
```

Comprueba los contenedores:

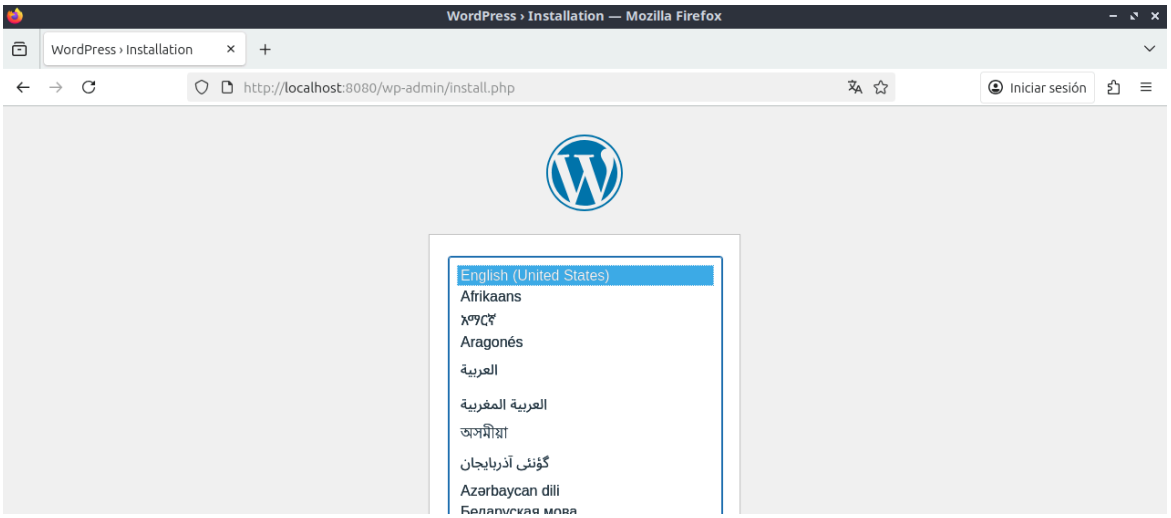
```
docker ps
```

```
alumno@alumno-virtualbox:~/Desktop/UD06-Caso01$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
          NAMES
a510209abeec   bac4f177a0d5   "/run.sh"                6 minutes ago    Up 6 minutes    0.0.0.0:3000->3000/tcp
          grafana
10c7a35c1c3a   a683da769912   "/bin/prometheus --c…"   6 minutes ago    Up 6 minutes    0.0.0.0:9090->9090/tcp
          prometheus
a2233fd5af11   7332768c717f   "docker-entrypoint.s…"   9 minutes ago    Up 9 minutes    0.0.0.0:8080->80/tcp
          wordpress
aac5ac9aa079   5107333e08a8   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:3306->3306/tcp, 33
060/tcp   mysql_db
474a56b48669   696e69e899e0   "/bin/node_exporter"     16 minutes ago   Up 15 minutes   0.0.0.0:9100->9100/tcp
          node_exporter
alumno@alumno-virtualbox:~/Desktop/UD06-Caso01$
```

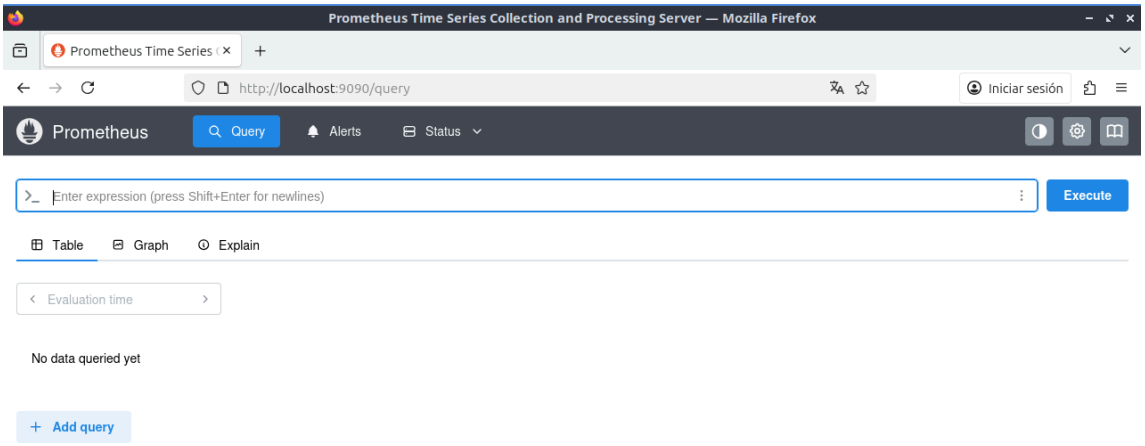
🌐Accede a los servicios creados

A continuación, mostramos el acceso vía navegador a los servicios que hemos creado en este caso práctico.

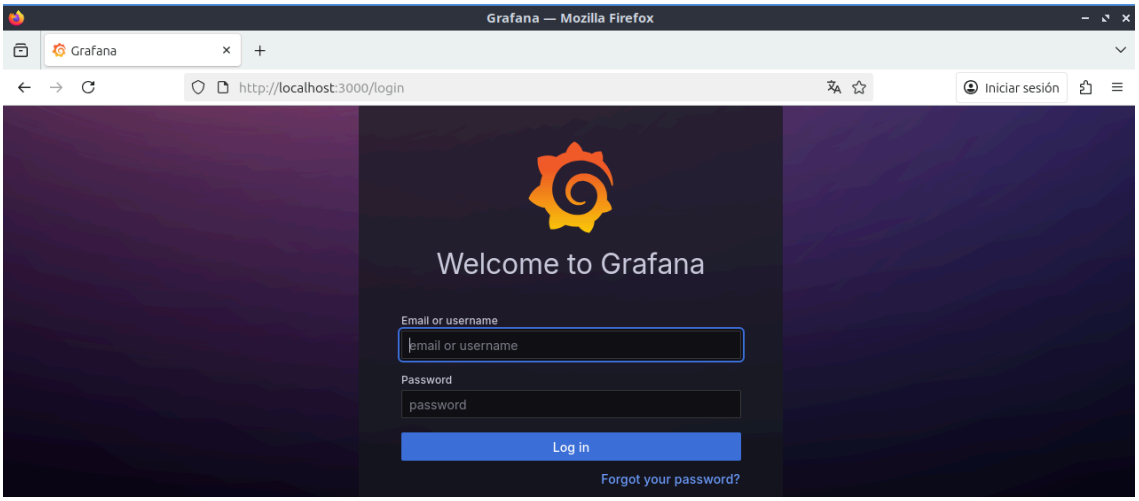
WordPress http://localhost:8080



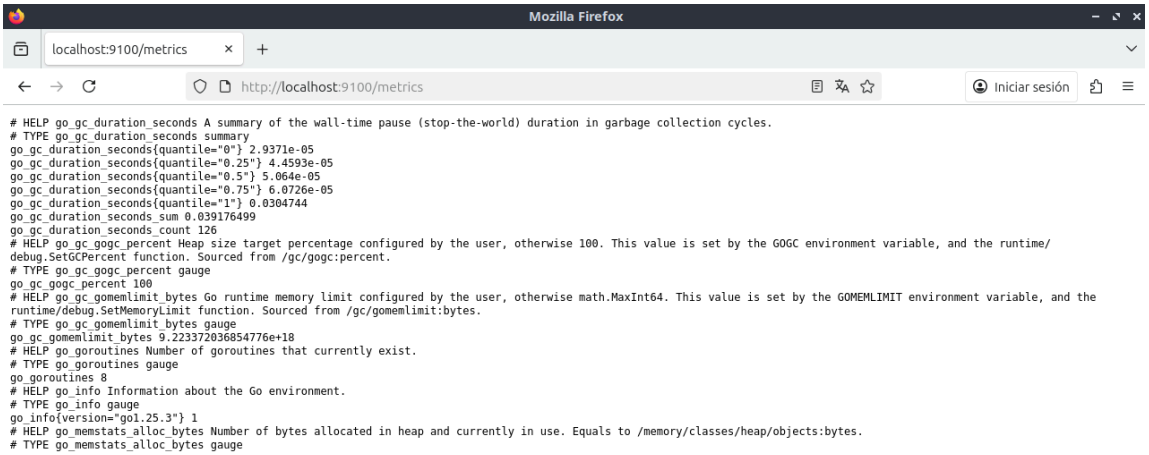
Prometheus http://localhost:9090



Grafana http://localhost:3000 usuario: admin password: admin



Node Exporter http://localhost:9100/metrics

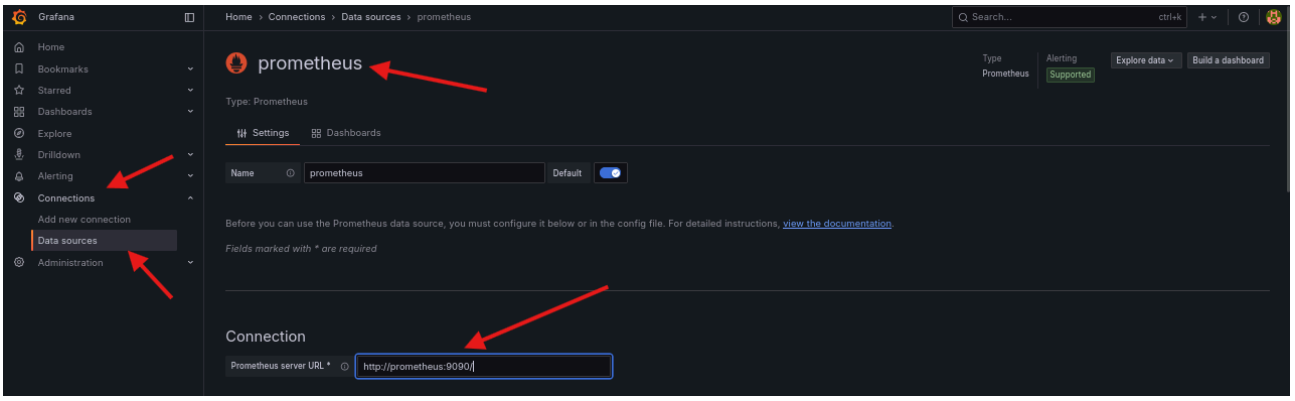


📊 5. CONFIGURAR PROMETHEUS EN GRAFANA

Una vez dentro de Grafana, entra con:

```
Usuario: admin
Contraseña: admin
```

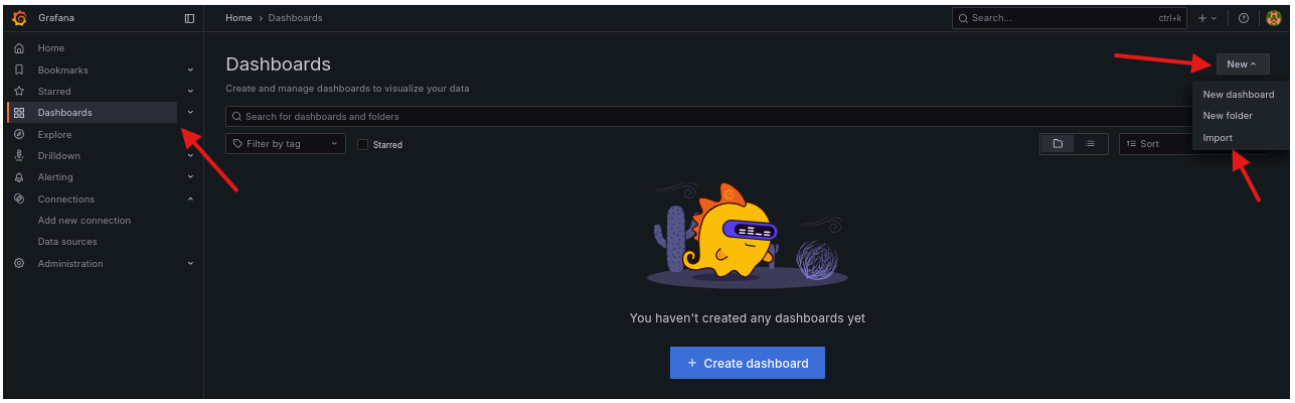
1. Esta contraseña puedes cambiarla al iniciar sesión.
2. Ve a **Connections → Data Sources → Add data source → Prometheus**
   a. En el campo **URL**, escribe: http://prometheus:9090
   b. Grafana y Prometheus comparten la red Docker interna, por eso usamos el nombre del contenedor, **y por eso NO USAMOS localhost**).
3. Clic en **Save & Test** → debería mostrar ✅ "Data source is working".



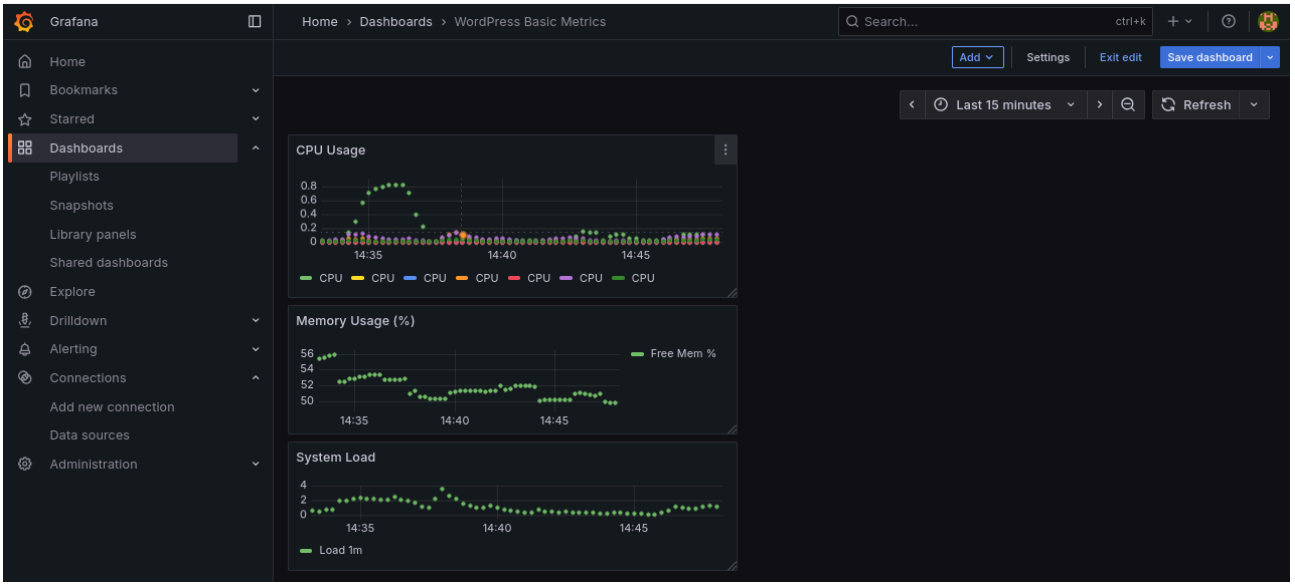
🅰️Opción A: Crear un dashboard básico (Importado de JSON)

Una vez conectada la fuente, puedes importar directamente un dashboard simple a partir de un fichero JSON. Utilizaremos para este ejemplo el archivo JSON que adjuntamos en el caso práctico.

Importa este dashboard en Grafana → **Dashboards → Import → Upload JSON file → Selecciona el archivo**. Verás tus métricas en tiempo real ✅



Tras esto, tendremos acceso a nuestro dashboard, similar a este:



dashboards/wordpress-basic.json

```
{
  "title": "WordPress Basic Metrics",
  "panels": [
    {
      "type": "graph",
      "title": "CPU Usage",
      "targets": [
        { "expr": "rate(node_cpu_seconds_total{mode!='idle'}[1m])", "legendFormat":
"CPU" }
      ]
    },
    {
      "type": "graph",
      "title": "Memory Usage (%)",
      "targets": [
        { "expr": "node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes *
100", "legendFormat": "Free Mem %" }
      ]
    },
    {
      "type": "graph",
      "title": "System Load",
      "targets": [
        { "expr": "node_load1", "legendFormat": "Load 1m" }
      ]
    }
  ]
}
```

🅱️ Opción B: Crear un Dashboard básico (Manual)

Puedes crear un Dashboard básico en Grafana equivalente al del caso anterior, con las siguientes consultas PromQL:

Panel 1 - Uso de CPU

```
rate(node_cpu_seconds_total{mode!="idle"}[1m])
```

Panel 2 - Memoria disponible

```
node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100
```

Panel 3 - Carga del sistema

```
node_load1
```

Tras ello, guarda el dashboard con nombre: WordPress System Overview

🧠 6. Cómo funciona el flujo

1. **WordPress** se ejecuta en un contenedor que usa MySQL para sus datos.
2. **Node Exporter** expone métricas del sistema del host (CPU, RAM, etc.).
3. **Prometheus** consulta (scrapea) periódicamente esas métricas.
4. **Grafana** se conecta a Prometheus y visualiza los datos mediante paneles.
5. **Terraform** define y mantiene toda esta infraestructura de forma declarativa.

💣 7. Destrucción segura

Cuando termines, destruye toda la infraestructura:

```
terraform destroy
```

Los **volúmenes persistentes** conservarán los datos de WordPress y MySQL:

```
docker volume ls
```

Para eliminarlos manualmente:

```
docker volume rm mysql_data_volume wordpress_data_volume
```

🧩 8. Detalle de los ficheros

📄 main.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = ">= 2.20.0"
    }
  }
}
```

```
provider "docker" {}

########################
# 🕸️ RED DOCKER PRIVADA
########################
resource "docker_network" "wp_net" {
  name = "wordpress_monitoring_net"
}

########################
# 💾 VOLUMENES PERSISTENTES
########################
resource "docker_volume" "mysql_data" {
  name = "mysql_data_volume"
}

resource "docker_volume" "wordpress_data" {
  name = "wordpress_data_volume"
}

########################
# 🗃️ MYSQL DATABASE
########################
resource "docker_image" "mysql" {
  name = "mysql:5.7"
}

resource "docker_container" "mysql" {
  name  = "mysql_db"
  image = docker_image.mysql.image_id

  networks_advanced {
    name = docker_network.wp_net.name
  }

  env = [
    "MYSQL_ROOT_PASSWORD=${var.mysql_root_password}",
    "MYSQL_DATABASE=${var.mysql_database}",
    "MYSQL_USER=${var.mysql_user}",
    "MYSQL_PASSWORD=${var.mysql_password}"
  ]

  mounts {
    target = "/var/lib/mysql"
    source = docker_volume.mysql_data.name
    type   = "volume"
  }

  ports {
    internal = 3306
    external = 3306
  }
}
```

```
########################
# 🌐 WORDPRESS APP
########################
resource "docker_image" "wordpress" {
  name = "wordpress:latest"
}

resource "docker_container" "wordpress" {
  name  = "wordpress"
  image = docker_image.wordpress.image_id

  networks_advanced {
    name = docker_network.wp_net.name
  }

  env = [
    "WORDPRESS_DB_HOST=${docker_container.mysql.name}:3306",
    "WORDPRESS_DB_NAME=${var.mysql_database}",
    "WORDPRESS_DB_USER=${var.mysql_user}",
    "WORDPRESS_DB_PASSWORD=${var.mysql_password}"
  ]

  mounts {
    target = "/var/www/html"
    source = docker_volume.wordpress_data.name
    type   = "volume"
  }

  ports {
    internal = 80
    external = var.wordpress_port
  }

  depends_on = [docker_container.mysql]
}

########################
# ⚙️ NODE EXPORTER
########################
resource "docker_image" "node_exporter" {
  name = "prom/node-exporter:latest"
}

resource "docker_container" "node_exporter" {
  name  = "node_exporter"
  image = docker_image.node_exporter.image_id

  networks_advanced {
    name = docker_network.wp_net.name
  }

  ports {
```

```
    internal = 9100
    external = var.node_exporter_port
  }
}

########################
# 📊 PROMETHEUS
########################
resource "docker_image" "prometheus" {
  name = "prom/prometheus:latest"
}

resource "docker_container" "prometheus" {
  name  = "prometheus"
  image = docker_image.prometheus.image_id

  networks_advanced {
    name = docker_network.wp_net.name
  }

  mounts {
    type   = "bind"
    source = abspath("${path.module}/prometheus.yml")
    target = "/etc/prometheus/prometheus.yml"
  }

  ports {
    internal = 9090
    external = var.prometheus_port
  }

  depends_on = [docker_container.node_exporter]
}

########################
# 📈 GRAFANA
########################
resource "docker_image" "grafana" {
  name = "grafana/grafana:latest"
}

resource "docker_container" "grafana" {
  name  = "grafana"
  image = docker_image.grafana.image_id

  networks_advanced {
    name = docker_network.wp_net.name
  }

  env = [
    "GF_SECURITY_ADMIN_USER=${var.grafana_user}",
    "GF_SECURITY_ADMIN_PASSWORD=${var.grafana_password}"
  ]
```

```
  ports {
    internal = 3000
    external = var.grafana_port
  }

  depends_on = [docker_container.prometheus]
}
```

variables.tf

```
variable "mysql_root_password" {
  description = "Contraseña root de MySQL"
  default     = "rootpass"
}

variable "mysql_database" {
  description = "Base de datos de WordPress"
  default     = "wpdb"
}

variable "mysql_user" {
  description = "Usuario MySQL"
  default     = "wpuser"
}

variable "mysql_password" {
  description = "Contraseña MySQL"
  default     = "wppass"
}

variable "wordpress_port" {
  description = "Puerto de WordPress"
  default     = 8080
}

variable "prometheus_port" {
  description = "Puerto de Prometheus"
  default     = 9090
}

variable "grafana_port" {
  description = "Puerto de Grafana"
  default     = 3000
}

variable "node_exporter_port" {
  description = "Puerto de Node Exporter"
  default     = 9100
}
```

```
variable "grafana_user" {
  description = "Usuario admin Grafana"
  default     = "admin"
}

variable "grafana_password" {
  description = "Contraseña admin Grafana"
  default     = "admin"
}
```

prometheus.yml

```
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: "node_exporter"
    static_configs:
      - targets: ["node_exporter:9100"]

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

outputs.tf

```
output "wordpress_url" {
  value       = "http://localhost:${var.wordpress_port}"
  description = "Acceso a WordPress"
}

output "prometheus_url" {
  value       = "http://localhost:${var.prometheus_port}"
  description = "Panel Prometheus"
}

output "grafana_url" {
  value       = "http://localhost:${var.grafana_port}"
  description = "Panel Grafana"
}
```