Grafana, Alert-Manager and Prometheus Manager

Vidal Peña, Arturo [1] Pérez Costoya, Fernando [2]

[1]Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Madrid

[2]Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Madrid

En colaboración con Accenture Technologies S.L.

Junio 2022

¿Qué es Prometheus?

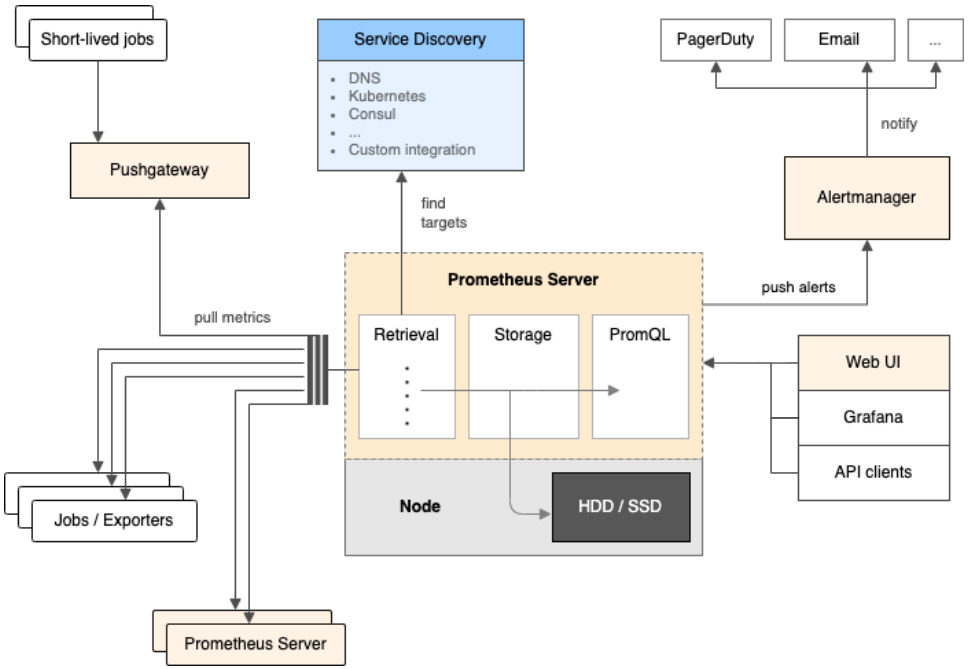


Figure 1: Arquitectura de funcionamiento de Prometheus

¿Qué se necesita para monitorizar con Prometheus?

- Servidor de prometheus
- Targets con etiquetas, agrupados por proyectos
- Reglas de alertado

¿Cómo se monitoriza un nuevo target?

Forma estática:

```
- job: node exporter
  static_configs:
  - targets:
    - "localhost:9091"
      labels:
      jobname: node
      os: linux
```

Forma dinámica:

```
- job: node exporter
  file_sd_configs:
  - files:
    - "targets/*.json"
    refresh_interval: 2m
```

```
{
  "targets": [
    "localhost:9091"
  ],
  "labels": {
    "jobname": "node",
    "os": "linux"
  }
}
```

Motivaciones: Problemas con la monitorización actual

1. Proceso lento y repetitivo
2. Con etiquetas diferentes, hay que separar los targets
3. No hay automatizaciones oficiales
4. Alta probabilidad de introducir errores humanos
5. Con ficheros grandes, los errores son difíciles de detectar

Objetivos: ¿Qué se pretende conseguir con la app?

- Automatizar el proceso de añadir targets
- Aumentar la rapidez y eficacia del proceso
- Disminuir el número de errores introducidos

¿Cómo funciona Ansible en este proyecto?

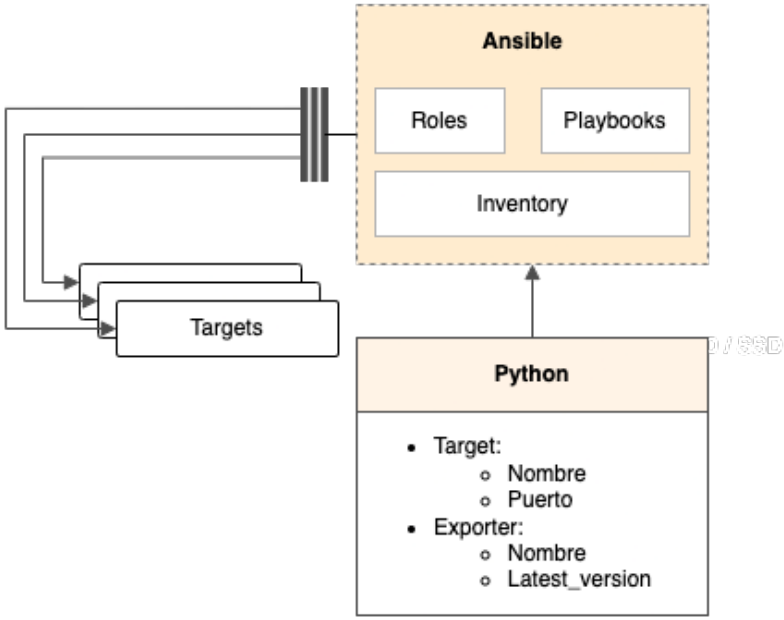


Figure 2: Diagrama ansible-node_exporter

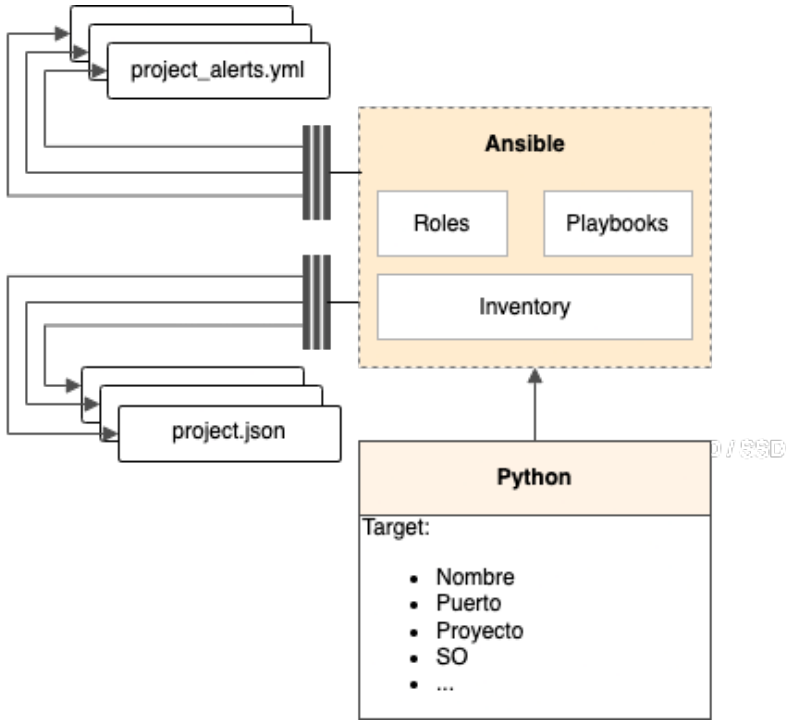


Figure 3: Diagrama ansible-ficheros_prometheus

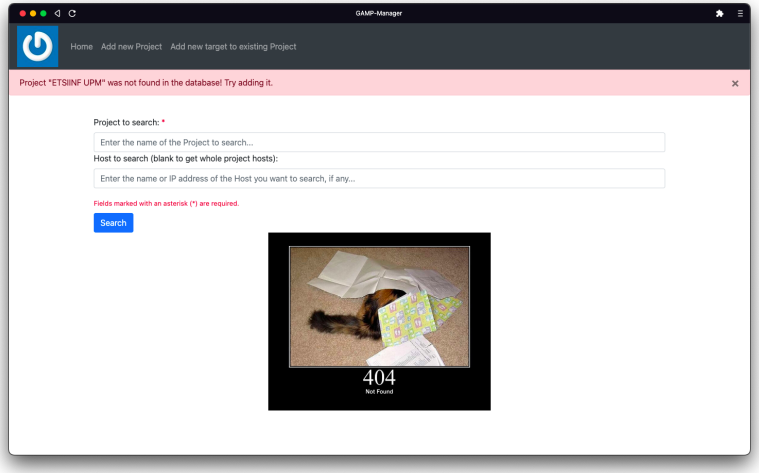
API en Python

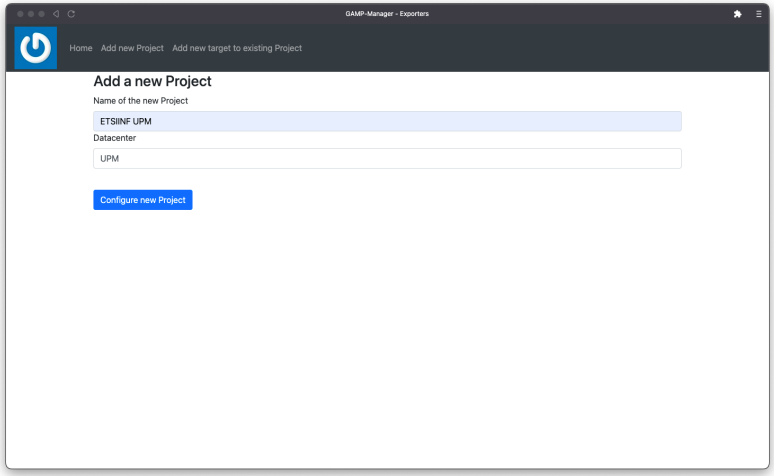
- Establece el backend
- Genera endpoints para conectar con frontend
- Ejecuta ansible para instalar node exporter en los targets
- Ejecuta ansible para añadir target a Prometheus
- Crea el exporter de versiones en el target
- Añade un fichero de targets y de reglas de alertado para cada proyecto

Interfaz

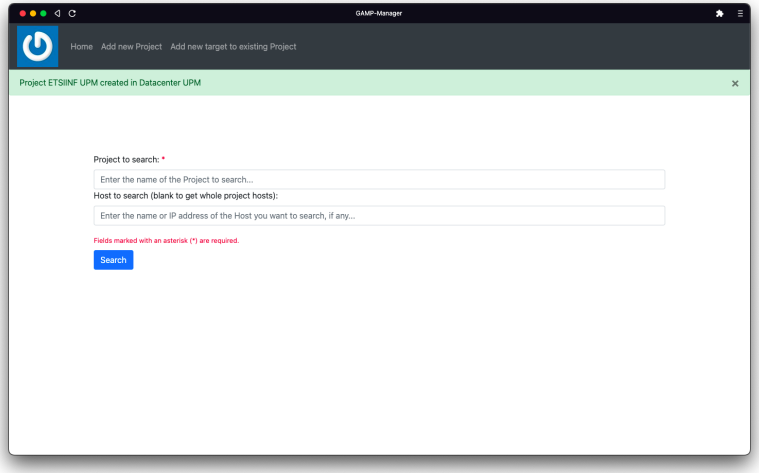
- Hecha con ElectronJS
- Simple y práctica
- Permite:
  1. Crear proyectos
  2. Añadir targets
  3. Buscar targets por projecto

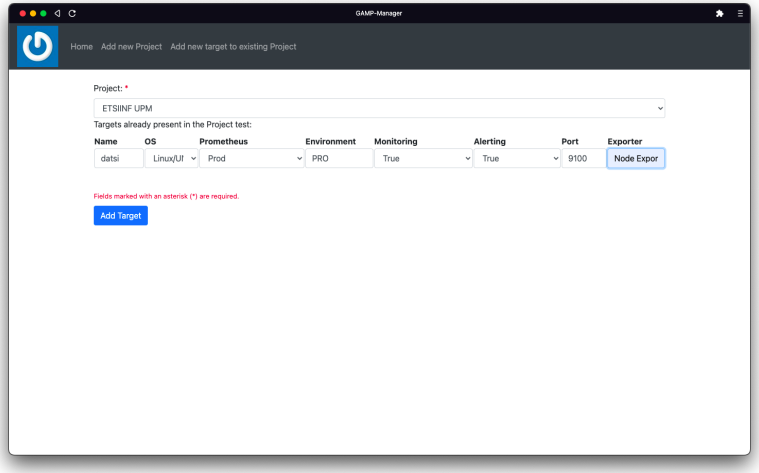
Resultados: Funcionamiento de la aplicación



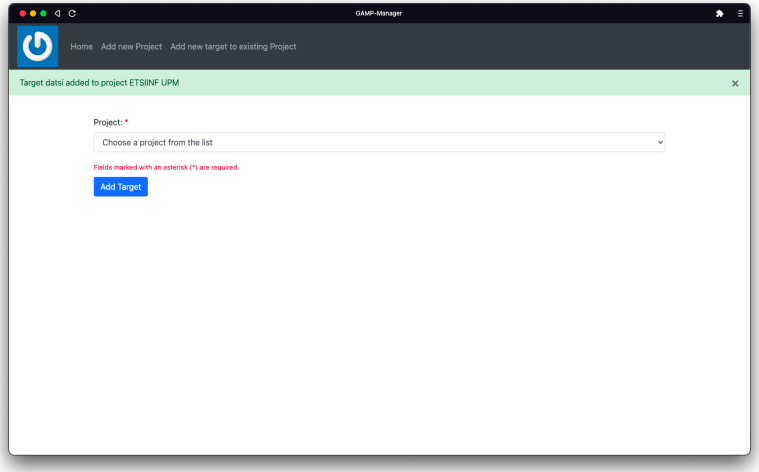


Resultados: Funcionamiento de la aplicación





Resultados: Funcionamiento de la aplicación




```
$ tree .
.
|-- prometheus.yml
|-- rules
|   |-- cbgi_record_rules.yml
|   |-- standard.yml
|-- targets

2 directories, 3 files
```

```
$ tree .
.
|-- prometheus.yml
|-- rules
|   |-- ETSIINF_UPM_alerts.yml
|   |-- cbgi_record_rules.yml
|   |-- standard.yml
|-- targets
    |-- ETSIINF_UPM.json

2 directories, 5 files
```

Trabajo futuro

- Trabajar con más exporters
- Añadir más servidores de Prometheus, Alertmanager y Grafana
- Monitorización en la nube
- Creación de dashboards en Grafana mediante la app

Conclusiones: App vs tradicional

Objetivos cumplidos:

- ✓ La opción de introducir errores se ve reducida a la insertada por el usuario
- ✓ Los targets se añaden automáticamente
- ✓ La velocidad en el proceso aumenta considerablemente