

SAM

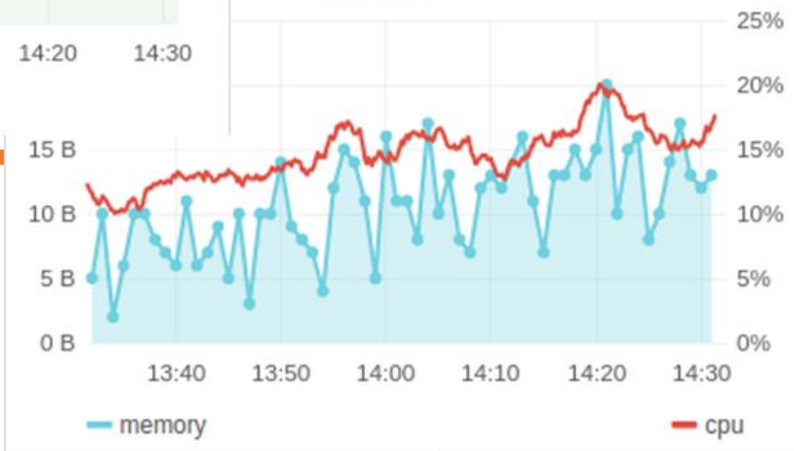
**Monitorizando InterSystems IRIS
con Grafana y Prometheus**

<https://github.com/es-comunidad-interSystems/webinar-sam.git>

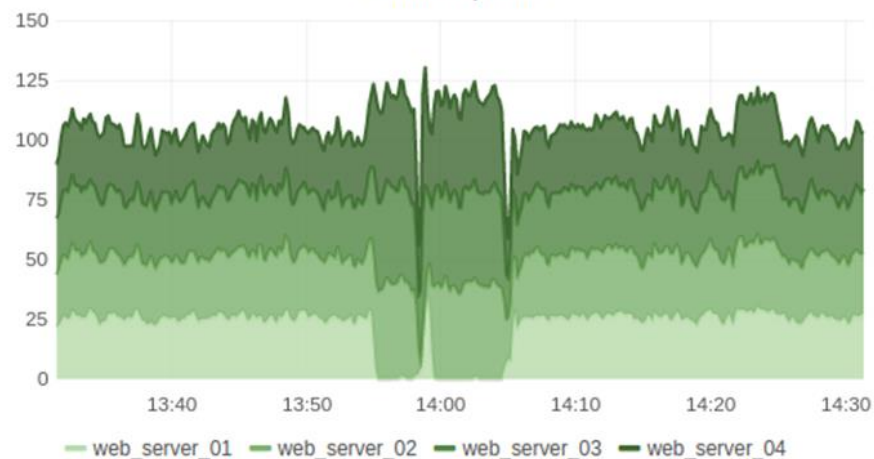
logins



Memory / CPU



server requests



Optimizando InterSystems IR
con Grafana y Prometheus



PIERRE-YVES DUQUESNOY

Senior Sales Engineer
de InterSystems Iberia

System Alerting & Monitoring (SAM)

¿Que es?

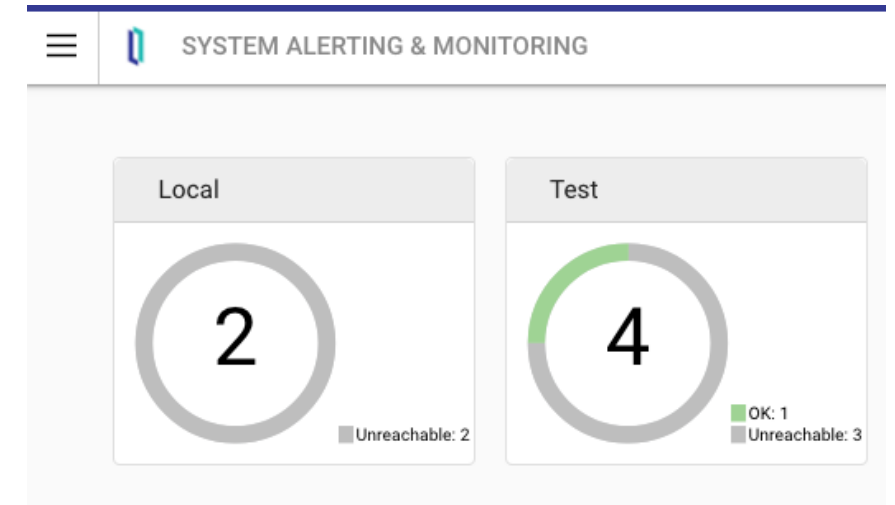
- una herramienta unificada de monitorización de clusters

¿Beneficios?

- Un único panel para todas las instancias de IRIS
- Agrupación de las instancias en Clusters
- Implementación simplificada
- Versión “Community” que se puede extender a Enterprise
- Metrics pre-construidas de IRIS: nada que instalar en las instancias a monitorizar (IRIS 2020.1+)

¿Como?

- Usando software Open Source
 - Prometheus – un proyecto CNCF
 - Grafana – El visualizado de metricas más utilizado
 - Agrupados en un docker-compose

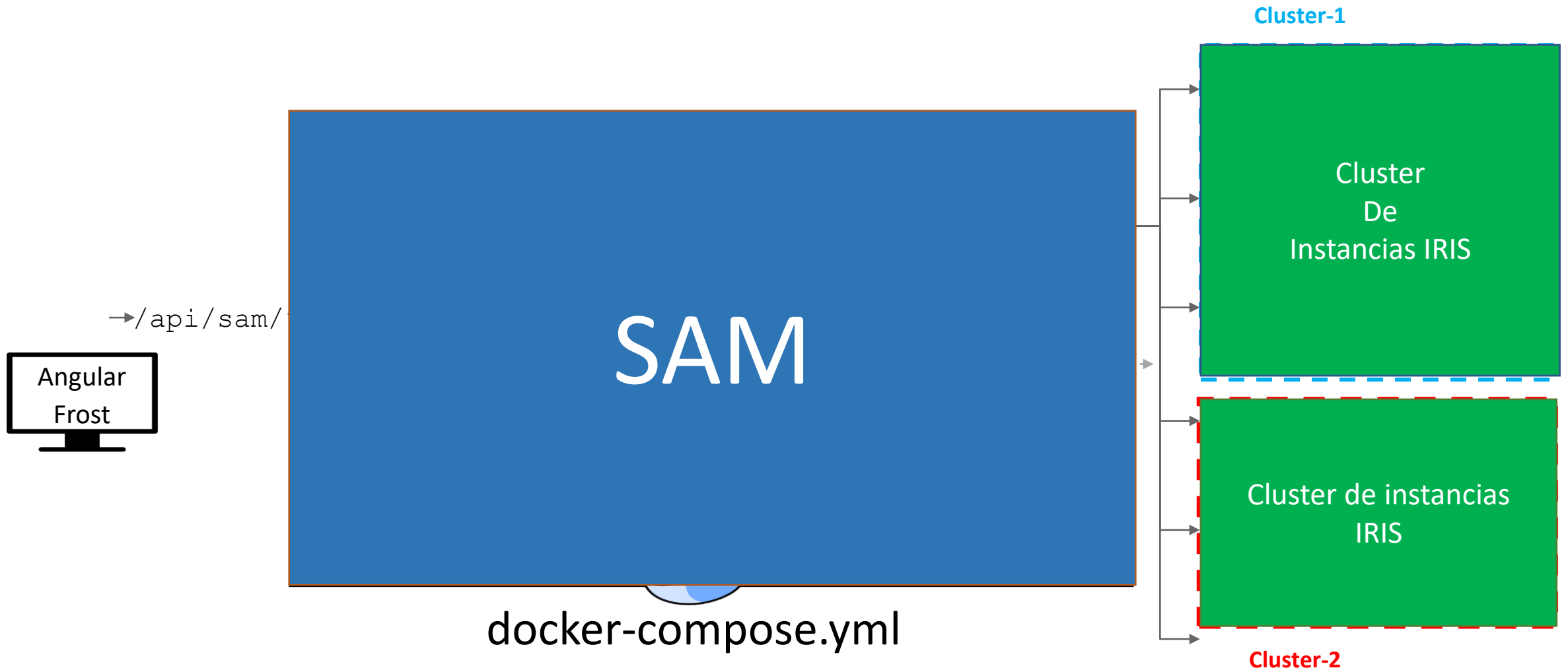


Que es System Alerting and Monitoring?

Una solución abierta para monitorizar InterSytem IRIS, proporcionando

- Un Interfaz Web
 - Definición de Clusters y Instancias
 - Monitorización del estado de las instancias
 - Visión de alertas
 - Definición de reglas de metricas
 - Visualización de las metricas
- Apoyado en componentes open source bien mantenidos y documentados
 - Alertmanager v0.20.0
 - Grafana v6.7.1
 - Nginx 1.17.9
 - Prometheus v2.17.1
 - SAM Manager 1.0
- Extensible
- Más funcionalidades previstas...

SAM –Vista General



SAM en el Servidor

- Incluido y habilitado en IRIS 2020.1+
 - Más funcionalidades en 2021+
- Construido encima de las herramientas existentes
 - System Monitor
 - Habilitado por defecto, Estado del Sistema “OK”, “WARNING”, “CRITICAL” +System Metrics (CPU, Lock Table...)
- Exportador para Prometheus
 - REST API
 - /api/monitor/metrics
 - /api/monitor/alerts

System Monitor

- Mantiene un único valor de salud del sistema
 - `$System.Monitor.State()`
 - Habilitado de fabrica
 - Indicadores de estado y uso de recursos (lck-table-%full, CPU-warning, ECP, etc.)
- Genera notificaciones y alertas
 - System Alerts (severity 2) Warnings (severity 1), OK (severity 0)
 - Reglas fijas para evaluar los valores y identificar desviaciones
 - Escribe en messages.log
 - Log Monitor (^MONITOR), Habilitado de fabrica
 - messages.log
 - alerts.log
- SYS.Monitor.SystemSensors
 - SYS.Monitor.DashboardSensors
 - SYS.Monitor.SAM.Sensors – Recoje sensores disponibles sin almacenarlos; Interogad por SAM Client
 - GetSensors() cada 30 seconds

SAM: Metricas de Servidor

- `/api/monitor/metrics`
 - Simple Clave/Valor
 - `iris_cpu_usage`, `iris_glo_ref_per_sec`, `iris_db_latency` ...
 - Permite definición de Metricas de aplicación Adicionales

SAM en el Cliente (Administración)

- Contenedores Docker arrancados con docker-compose
- SAM Manager
 - Instancia IRIS, con la Aplicación SAM principal
 - Mantiene Historial de Datos
- Nginx
 - Servidor Web, control de accesos
- Prometheus
 - BBDD TimeSeries, recogida de datos, visualización, integración
- Grafana
 - Visualization
- Alert Manager
 - Deduplication, Enrutamiento de alertas

Config Inicial

- Descargar el TAR sam-<version>.tar.gz del WRC -> Components or GitHub
 - <https://github.com/intersystems-community/sam>
 - Decomprimir con:
 - `tar zpxvf sam-<version>.tar.gz`
 - Arrancar: `./start.sh`
 - Verificar permisos RW en ./config y subdirectorios.
- <http://localhost:8080/api/sam/app/index.csp>
- Parar con: `./stop.sh`

Definir un Cluster

- Cluster: Grupo de Instancias a gestionar juntas
- Desde el portal: definir el cluster y añadir Instancias IRIS
 - Instancias IRIS 2020.1+ pre-configuradas para ser accesibles en la APIs: /api/monitor
 - Verificar que acceso sin Autenticar esta permitido
- Ajustar la configuración básica
 - Numero de días (1 to 30) de datos históricos a almacenar

Definir Alertas de Prometheus

- Reglas de Alertas de Prometheus para generar Alertas según condiciones

- Prometheus Query Language (PromQL)
- <https://prometheus.io/docs/prometheus/latest/querying/basics/>
- Sintaxis:

`metric_name{cluster="cluster_name",label(s)}>value`

- Permite Operadores logicos y numericos

```
iris_cpu_usage{cluster="test"}>90
```

```
(iris_db_size_mb{cluster="test",id="USER"}/iris_db_max_size_mb{cluster="test",id="USER"})*100>90
```

```
iris_ecp_conn{cluster="production"}<1 or iris_ecp_conn{cluster="production"}>20
```

```
iris_system_alerts_new{cluster="test"}>=1 and iris_system_monitor_health_state{cluster="test"}!=0
```

Gestionar Alertas en SAM

- Reglas de Alertas de Prometheus para generar Alertas según condiciones
 - Prometheus Query Language (PromQL)
 - <https://prometheus.io/docs/prometheus/latest/querying/basics/>
 - Sintaxis:
`metric_name{cluster="cluster_name",label(s)}>value`
 - Permite Operadores logicos y numericos

```
iris_cpu_usage{cluster="test"}>90
```

```
(iris_db_size_mb{cluster="test",id="USER"}/iris_db_max_size_mb{cluster="test",id="USER"})*100>90
```

```
iris_ecp_conn{cluster="production"}<1 or iris_ecp_conn{cluster="production"}>20
```

```
iris_system_alerts_new{cluster="test"}>=1 and iris_system_monitor_health_state{cluster="test"}!=0
```

Añadir una Métrica de Aplicación

- SubClass de %SYS.Monitor.SAM.Abstract
- Agrupar las métricas en un “PRODUCT”
- Implementar GetSensors()
 - SetSensor(Sensor , Value , Item As %String = "")
 - Do ..SetSensor(“TubesReceived”,+\$Get(^User.TubesReceivedD))
- Add Class to Configuration

```
%SYS>Do ##class(SYS.Monitor.SAM.Config).AddApplicationClass(ClassName,Namespace)
```

- **Añadir Permisos** a Aplicación Web /api/monitor
 - Ejecutar código y acceder a datos en “Namespace”
- Probar with <http://ServerIP:Port/csp/monitor/metrics>

Licenciamiento de SAM

SAM Community Edition

- 5 usuarios concurrentes
- 8 cores
- 10 GB de BBDD

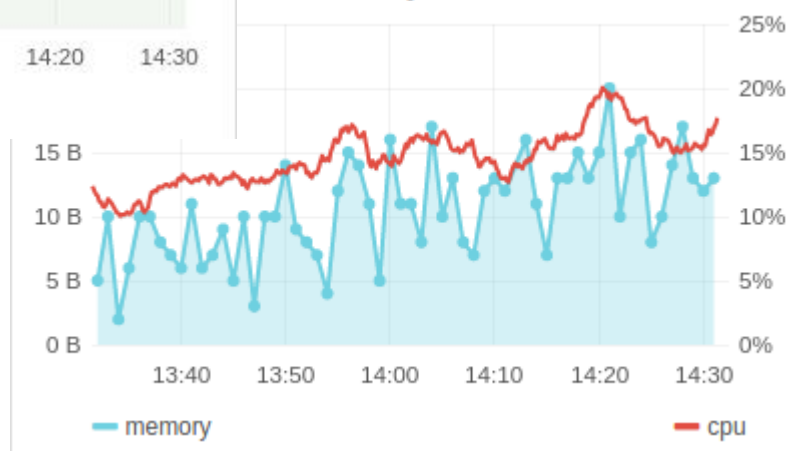
SAM Enterprise

- Usuarios y Cores según Licencia
- Permite InterSystems Mirroring
- (no Sharding included)

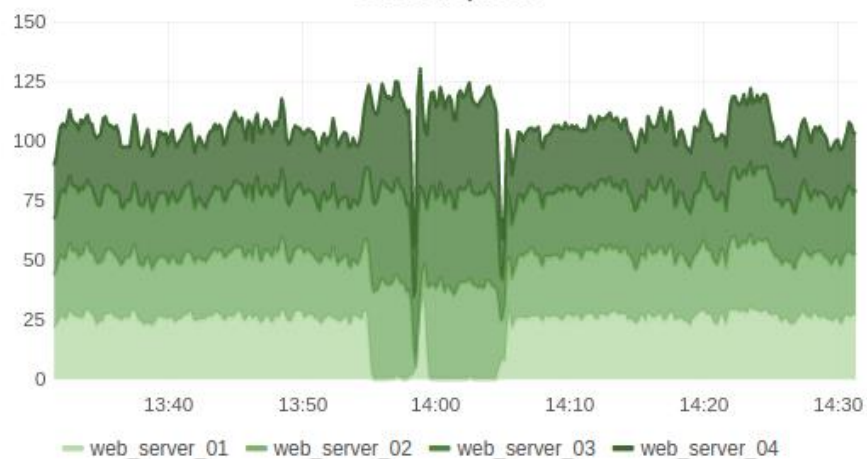
logins



Memory / CPU



server requests



DEMO