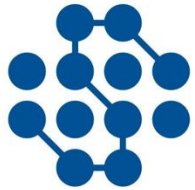
DATA SATURDAYS #64 PARMA 2024

HowTo Monitor&Metrics Microsoft SQL on-premise by TimeSeries DB

Sponsor & Org



DATA SKILLS
UNDERSTANDING THE WORLD

Bi Factory

DATA KNOWLEDGE ADVISOR

Work Path

IT OPERATIONS (ITOps)

Enter your sub headline here

Windows PowerShell

WindowServer.it

Community

Connect

giulianolatini

giulianolatini

giulianolatini

Questo talk è AI-free?

“ *Nessun Large Language Model o Intelligenza Artificiale Generativa sono stati maltrattati per lo sviluppo di questa presentazione.*

Hanno contribuito rispondendo a domande umane.

Le risposte lette ed analizzate in modo critico dall’umano, contribuiscono alla presentazione.

***Io ho selezionato** quelle di **qualità** secondo parametri orgogliosamente individuali e **verificando** (per quanto umanamente possibile) la loro **veridicità**!*

”

Take care of …

Microsoft® SQL Server®

Metriche, queste sconosciute?

“Le metriche IT sono misurazioni quantificabili utilizzate per valutare vari aspetti dei sistemi, processi e prestazioni IT all'interno di un'organizzazione. Queste misurazioni aiutano i team e le organizzazioni a comprendere le prestazioni IT, l'erogazione del valore e l'efficacia operativa.

Metriche di Prestazione
Monitorano le operazioni di prodotti e servizi, inclusi il tempo di attività dell'infrastruttura e l'affidabilità del sistema.

Thank’s to Perplexity *”*

```
PS C:\Windows\System32> Get-ChildItem | Measure-Object

Count             : 4852
Average           :
Sum               :
Maximum           :
Minimum           :
StandardDeviation :
Property          :
```

```
PS C:\Windows\System32> Get-ChildItem -Filter *.exe | Measure-Object

Count             : 655
Average           :
Sum               :
Maximum           :
Minimum           :
StandardDeviation :
Property          :
```


Controller Class
All
Namespace
live
Ingress
All
Ingress Pod
All
Config Reloads
Overview
Requests (period)
68.8 K
% Succe...
97.0%
Conns (...
314
Reqs (2...
768
% Success (2m)
92.6%
HTTP 1/2xx (2m)
496
HTTP 3xx (2m)
200
HTTP 4xx (2m)
72
HTTP 5xx (2m)
0
HTTP Requests / Ingress
1.25 K
1 K
750
500
250
0
22:30
23:00
23:30
00:00
00:30
01:00
HTTP Status Codes
1.50 K
1 K
500
0
22:30
23:00
23:30
00:00
00:30
01:00
HTTP 200
HTTP 204
HTTP 206
HTTP 301
HTTP 302
HTTP 304
HTTP 400
HTTP 403
HTTP 404
HTTP 408
HTTP 499
HTTP 500
HTTP 502
HTTP 503
HTTP 504
Total HTTP Requests
3 K
2.50 K
2 K
1.50 K
1 K
Latency
Latency (Average Percentiles)
50 s
40 s
30 s
20 s
10 s
0 s
23:00
00:00
01:00
min
max
avg
0.99
4.3 s
42.3 s
6.6 s
0.95
2.4 s
4.5 s
2.8 s
0.9
2.1 s
2.7 s
2.3 s
0.5
357.2 ms
631.6 ms
436.9 ms
0.75
1.0 s
1.8 s
1.4 s
Average
3.2 s
5.1 s
4.1 s
Latency Heatmap
1 min
20 s
5 s
1 s
250 ms
50 ms
10 ms
22:30
23:00
23:30
00:00
00:30
01:00
0
1,000
2,000
2,869
Connections
New Connections Opened (Controller / Ingress Pod)
700
600
500
400
300
200
100
0
Total Connections Open (Controller / Ingress Pod)
320
315
310
305
300

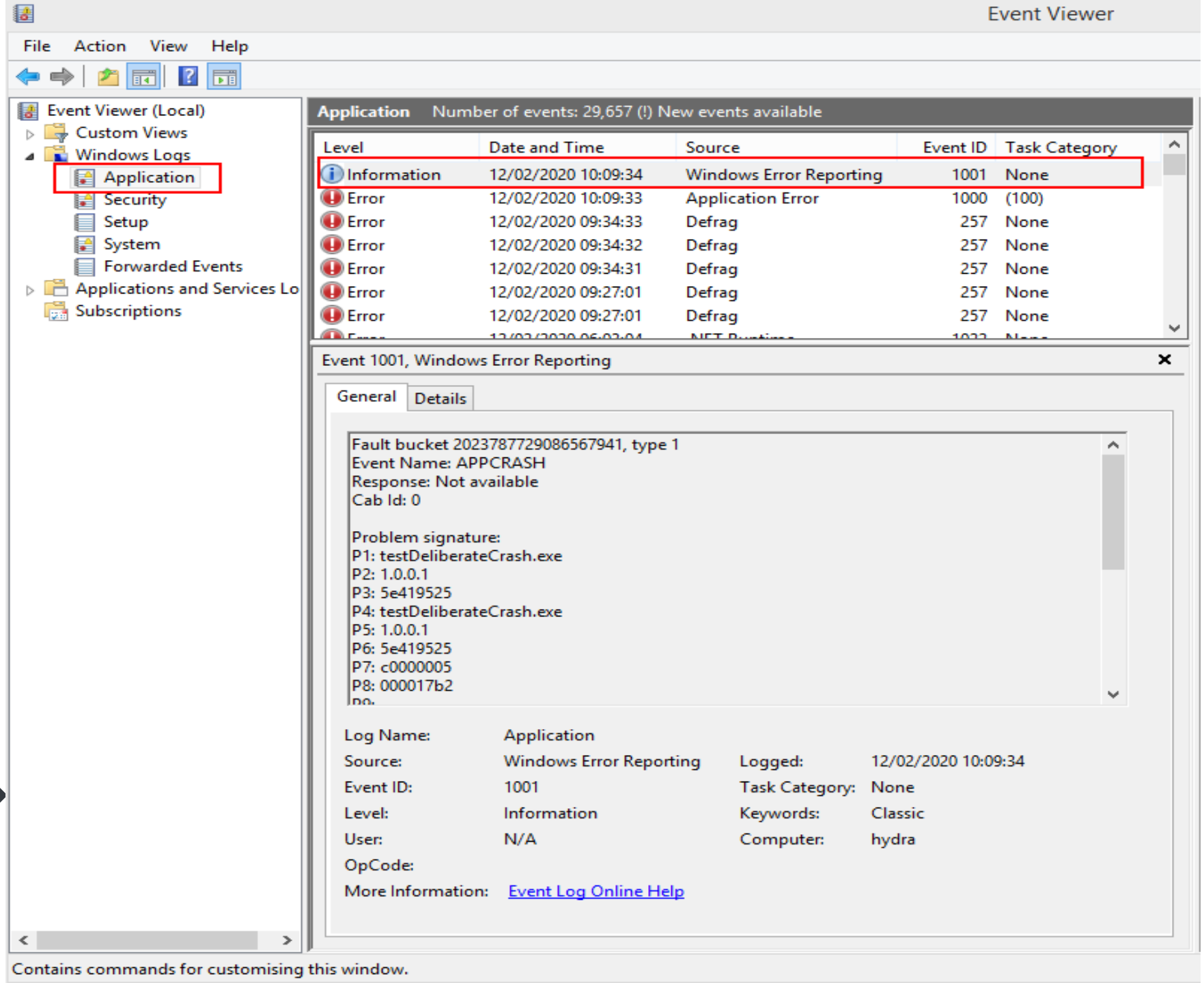
Log, questo sconosciuto?

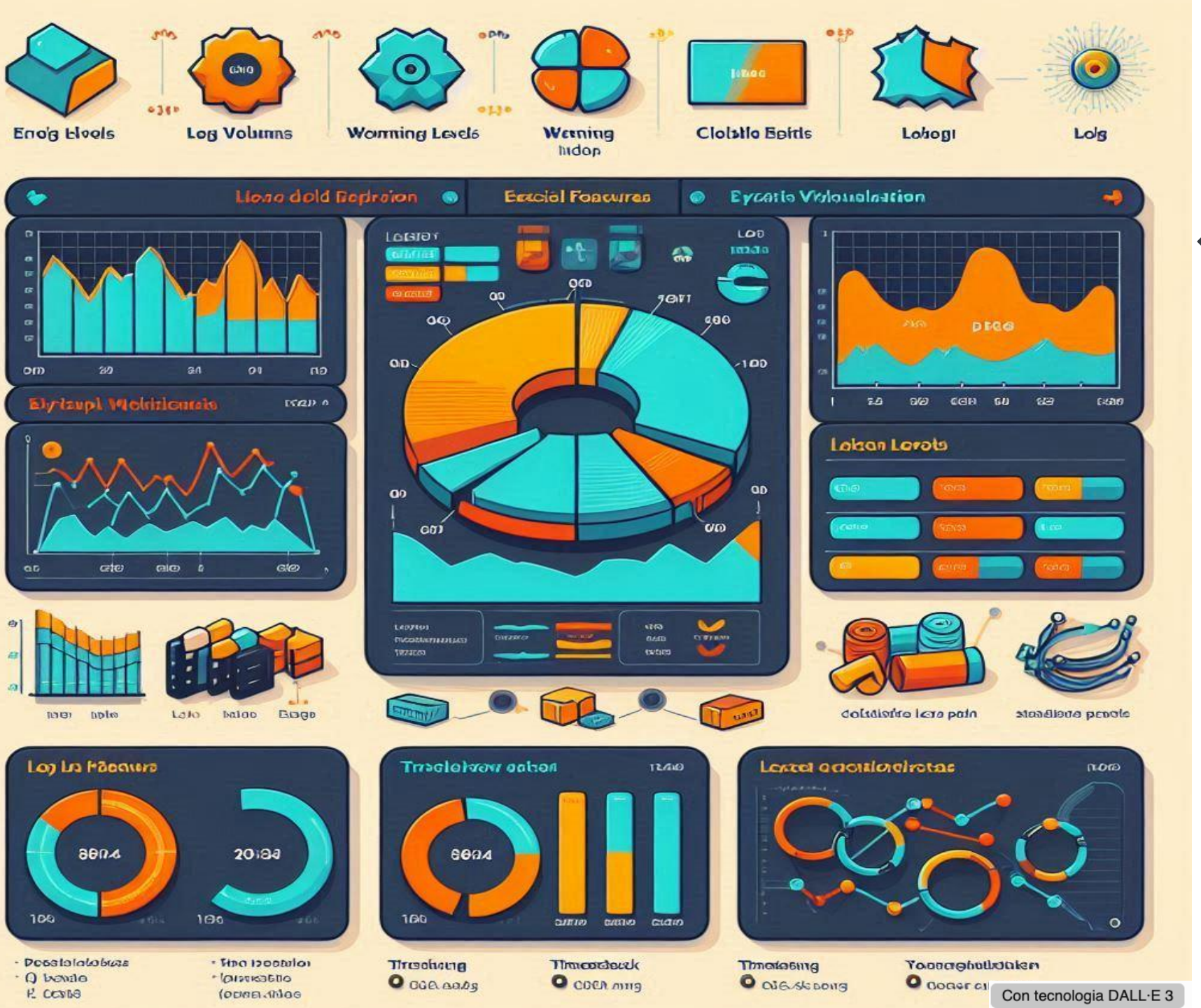
“ Il log è la registrazione cronologica delle azioni ed eventi che si verificano all'interno di un sistema informatico. Si tratta di un file in cui un software registra in modo sequenziale e automatico informazioni cruciali relative alle attività svolte da utenti, amministratori o processi automatizzati.

Log Applicativi
Documentano gli eventi che si verificano all'interno delle applicazioni software

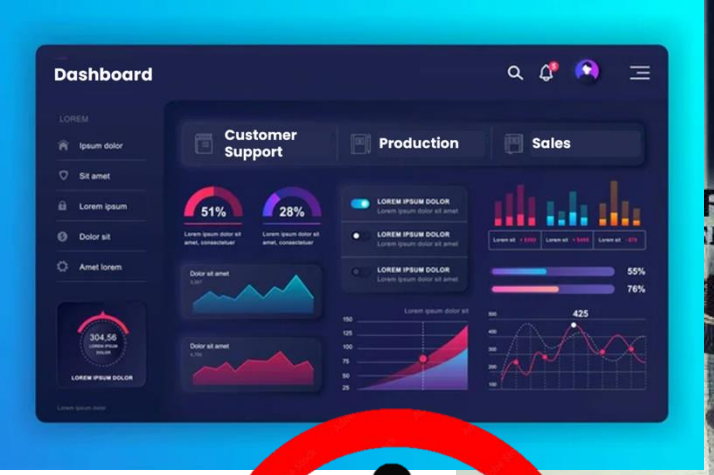
Log di Sistema
Registrano eventi a livello del sistema operativo, come il caricamento dei driver delle periferiche o il collegamento di dispositivi USB

Thank’s to Perplexity *”*

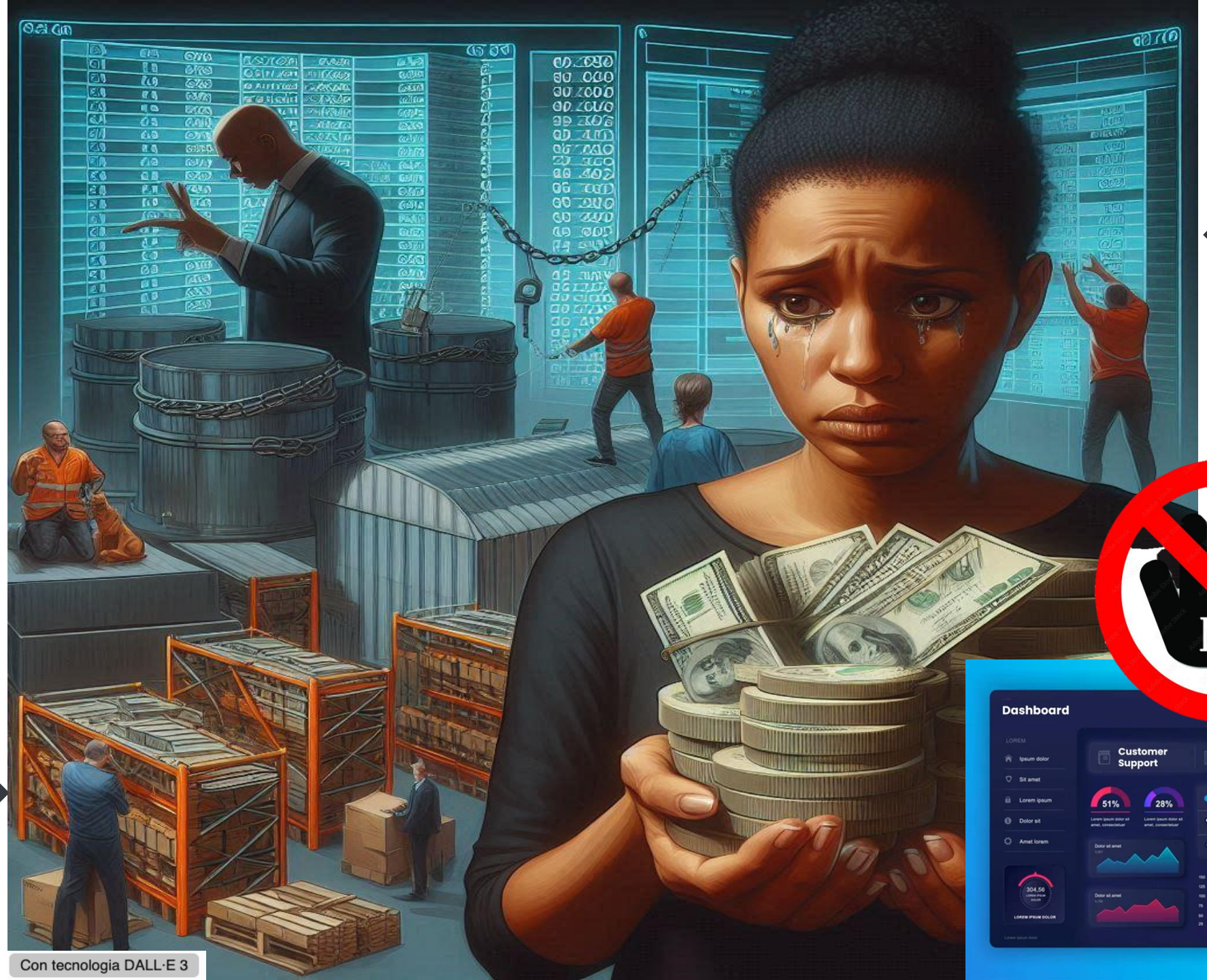

Event Viewer
File Action View Help
Event Viewer (Local)
Custom Views
Windows Logs
Application
Security
Setup
System
Forwarded Events
Applications and Services Lo
Subscriptions
Application Number of events: 29,657 (!) New events available
Level Date and Time Source Event ID Task Category
Information 12/02/2020 10:09:34 Windows Error Reporting 1001 None
Error 12/02/2020 10:09:33 Application Error 1000 (100)
Error 12/02/2020 09:34:33 Defrag 257 None
Error 12/02/2020 09:34:32 Defrag 257 None
Error 12/02/2020 09:34:31 Defrag 257 None
Error 12/02/2020 09:27:01 Defrag 257 None
Error 12/02/2020 09:27:01 Defrag 257 None
Event 1001, Windows Error Reporting
General Details
Fault bucket 2023787729086567941, type 1
Event Name: APPCRASH
Response: Not available
Cab Id: 0
Problem signature:
P1: testDeliberateCrash.exe
P2: 1.0.0.1
P3: 5e419525
P4: testDeliberateCrash.exe
P5: 1.0.0.1
P6: 5e419525
P7: c0000005
P8: 000017b2
Log Name: Application
Source: Windows Error Reporting
Logged: 12/02/2020 10:09:34
Event ID: 1001
Task Category: None
Level: Information
Keywords: Classic
User: N/A
Computer: hydra
OpCode:
More Information: Event Log Online Help
Contains commands for customising this window.



Con tecnologia DALL·E 3



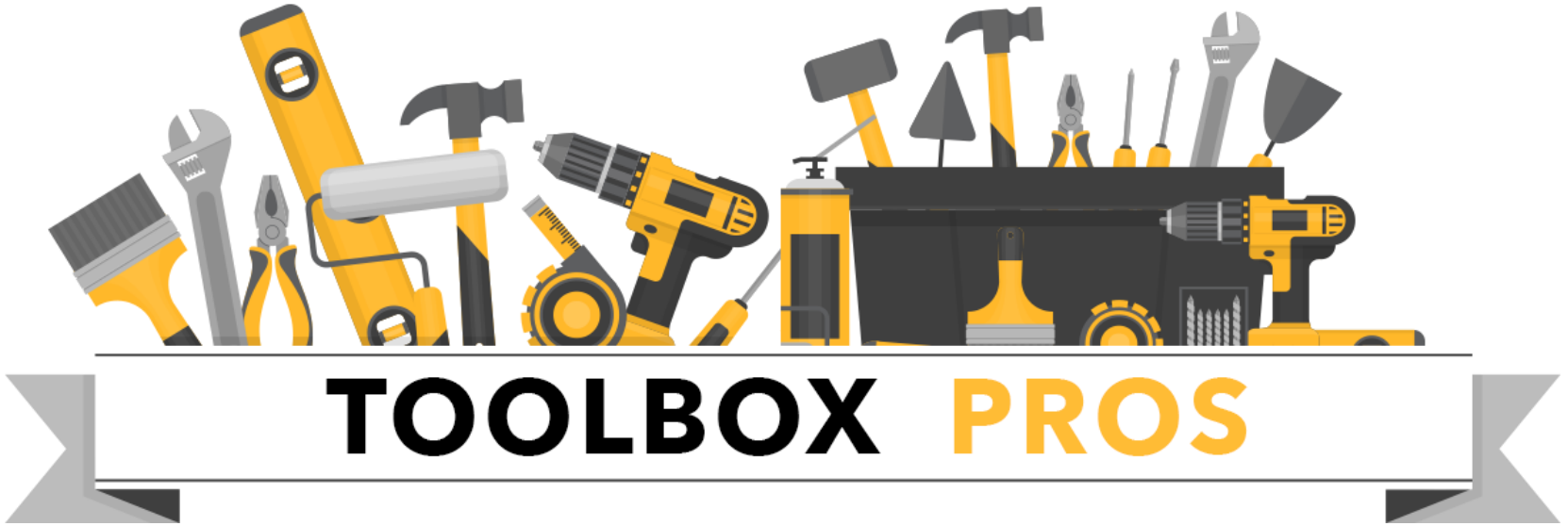
Con tecnologia DALL·E 3



Con tecnologia DALL·E 3

Take care of...

Microsoft® SQL Server®



Grafana

Prometheus

influxdb

Kapacitor

PowerShell

Microsoft che risorse mette a disposizione?

Learn / PowerShell /

Microsoft.PowerShell.Diagnostics

Reference

Feedback

This section contains the help topics for the cmdlets that are installed with the PowerShell Microsoft.PowerShell.Diagnostics module, which contains cmdlets that manage data from event logs.

This module is only available on the Windows platform.

Microsoft.PowerShell.Diagnostics

Expand table

Get-Counter	Gets performance counter data from local and remote computers.
Get-WinEvent	Gets events from event logs and event tracing log files on local and remote computers.
New-WinEvent	Creates a new Windows event for the specified event provider.

Learn / PowerShell /

Microsoft.PowerShell.Management

Get-EventLog

Reference

Feedback

Module: Microsoft.PowerShell.Management

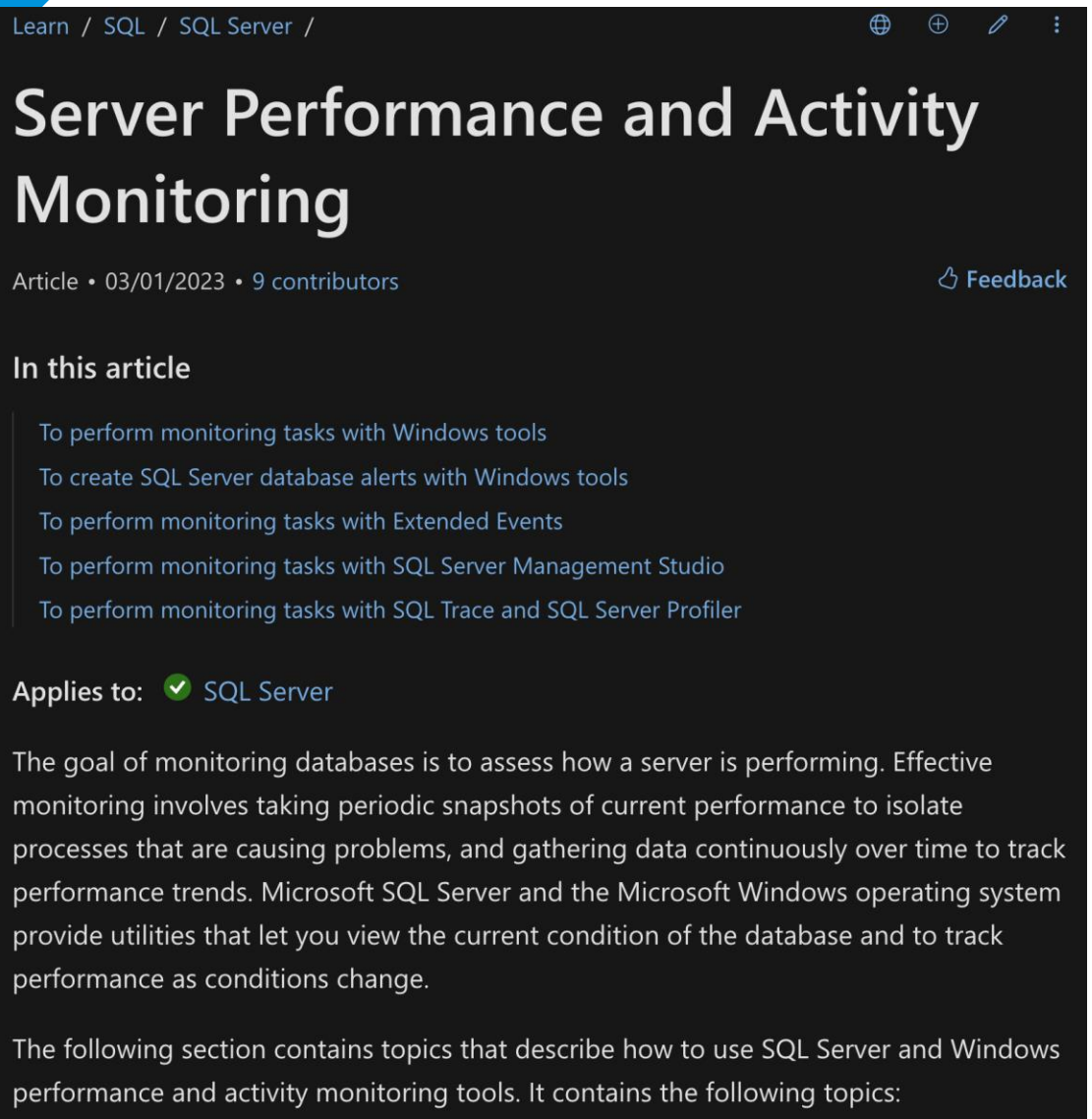
In this article

- Syntax
- Description
- Examples
- Parameters

Show 4 more

Gets the events in an event log, or a list of the event logs, on the local computer or remote computers.

Microsoft che risorse mette a disposizione?



Learn / SQL / SQL Server /

Server Performance and Activity Monitoring

Article • 03/01/2023 • 9 contributors

Feedback

In this article

- To perform monitoring tasks with Windows tools
- To create SQL Server database alerts with Windows tools
- To perform monitoring tasks with Extended Events
- To perform monitoring tasks with SQL Server Management Studio
- To perform monitoring tasks with SQL Trace and SQL Server Profiler

Applies to: SQL Server

The goal of monitoring databases is to assess how a server is performing. Effective monitoring involves taking periodic snapshots of current performance to isolate processes that are causing problems, and gathering data continuously over time to track performance trends. Microsoft SQL Server and the Microsoft Windows operating system provide utilities that let you view the current condition of the database and to track performance as conditions change.

The following section contains topics that describe how to use SQL Server and Windows performance and activity monitoring tools. It contains the following topics:

Microsoft Learn

Learn / Azure / Azure Monitor /

Azure Monitor overview

Article • 09/11/2024 • 37 contributors

Feedback

In this article

- High level architecture
- Data sources
- Data collection and routing
- Data platform

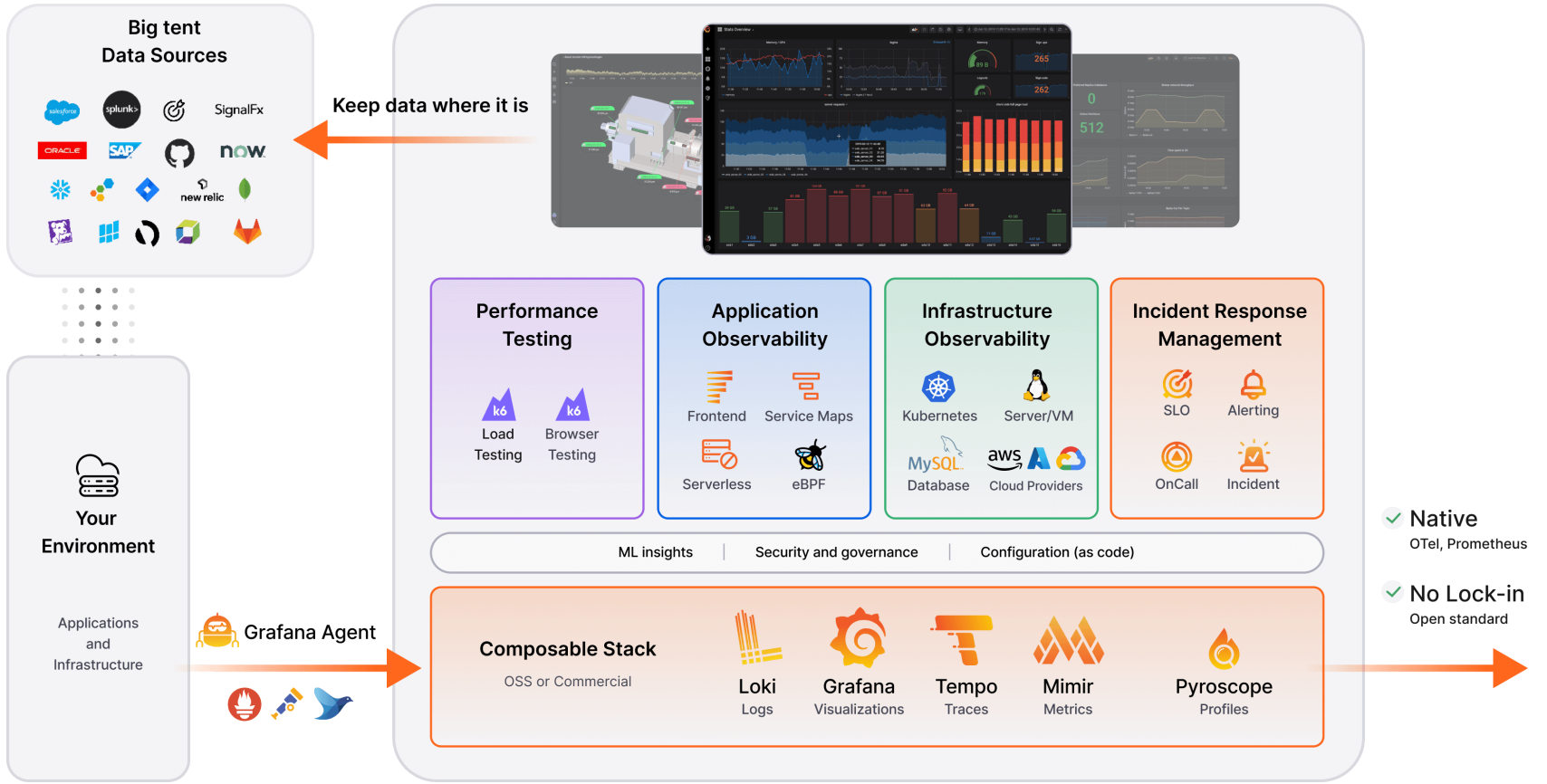
Show 4 more

Azure Monitor is a comprehensive monitoring solution for collecting, analyzing, and responding to monitoring data from your cloud and on-premises environments. You can use Azure Monitor to maximize the availability and performance of your applications and services. It helps you understand how your applications are performing and allows you to manually and programmatically respond to system events.

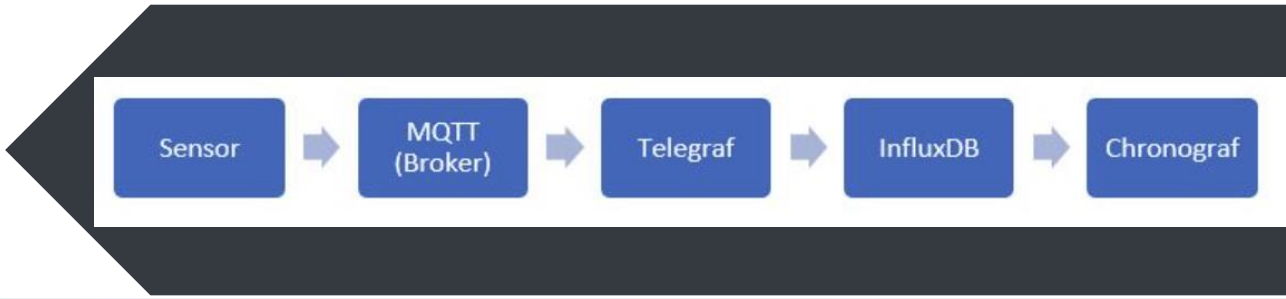
LGTM Stack & Prometheus

The actually useful free plan

Grafana Cloud Free Tier
- 10k series Prometheus metrics
- 50GB logs, 50GB traces, 50GB profiles
- 500VUh k6 testing
- 20+ Enterprise data source plugins
- 100+ pre-built solutions



TICK or TIG Stack



< MOST POPULAR >

InfluxDB Cloud Serverless

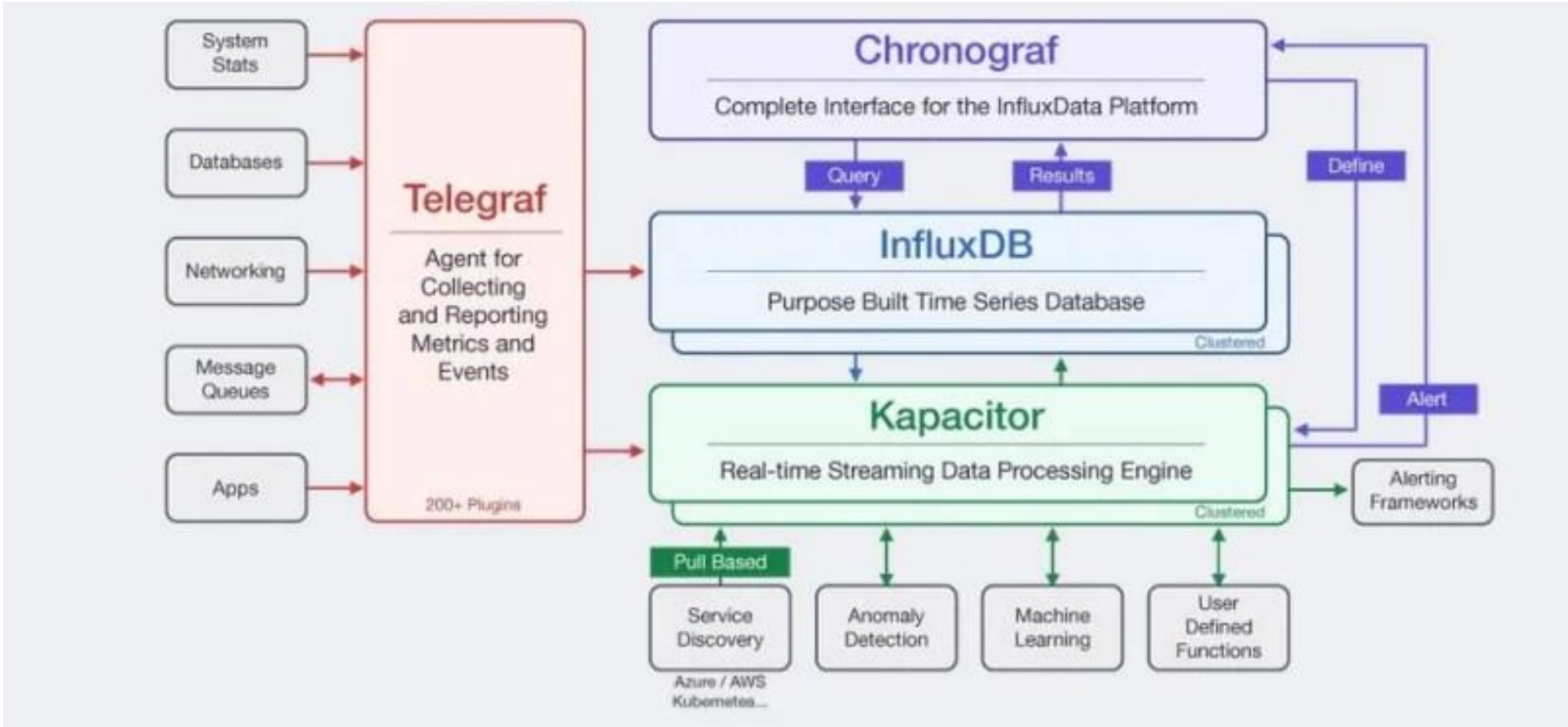
Start for Free

Cloud service for smaller workloads on shared infrastructure

Fully-managed, elastic, multi-tenant service. Pay only for what you use—no minimums or long-term commitment.

Get started for free, and get a $250 credit for 30 days upon upgrade.

Get More Details →



Take care of...

Microsoft® SQL Server®

Grafana

Prometheus

influxdb

Kapacitor

PowerShell

Shortcut to monitoring

Shortcut to monitoring

Silent install options

- `/CONFIG=<path>` Path to the configuration file. Default: `$INSTDIR\config.alloy`
- `/DISABLEREPORTING=<yes|no>` Disable data collection. Default: `no`
- `/DISABLEPROFILING=<yes|no>` Disable profiling endpoint. Default: `no`
- `/ENVIRONMENT="KEY=VALUE\0KEY2=VALUE2"` Define environment variables for Windows Service. Default: ``

Service Configuration

Alloy uses the Windows Registry `HKLM\Software\GrafanaLabs\Alloy` for service configuration.

- `Arguments` (Type `REG_MULTI_SZ`) Each value represents a binary argument for alloy binary.
- `Environment` (Type `REG_MULTI_SZ`) Each value represents a environment value `KEY=VALUE` for alloy binary.

Shortcut to monitoring

Expose the UI to other machines

By default, Alloy listens on the local network for its HTTP server. This prevents other machines on the network from being able to access the UI for debugging.

To expose the UI to other machines, complete the following steps:

1. Follow Change command-line arguments to edit command line flags passed to Alloy.
2. Add the following command line argument:

```shell
--server.http.listen-addr=LISTEN_ADDR:12345
```

Replace the following:

- `<LISTEN_ADDR>`: An IP address which other machines on the network have access to. For example, the IP address of the machine Alloy is running on.

  To listen on all interfaces, replace `<LISTEN_ADDR>` with `0.0.0.0`.

Shortcut to monitoring

Microsoft SQL Server

Grafana Cloud

alloy Copy

```
prometheus.exporter.mssql "integrations_mssql" {
    connection_string = "sqlserver://user:pass@127.0.0.1:1433"
}

discovery.relabel "integrations_mssql" {
    targets = prometheus.exporter.mssql.integrations_mssql.targets

    rule {
        target_label = "instance"
        replacement  = constants.hostname
    }

    rule {
        target_label = "job"
        replacement  = "integrations/mssql"
    }
}

prometheus.scrape "integrations_mssql" {
    targets    = discovery.relabel.integrations_mssql.output
    forward_to = [prometheus.remote_write.metrics_service.receiver]
    job_name   = "integrations/mssql"
}
```

Shortcut to monitoring

Third component: Write metrics to Prometheus

Paste the following component configuration below the previous component in your `config.alloy` file:

alloy — Copy

```
prometheus.remote_write "metrics_service" {
    endpoint {
        url = "http://localhost:9090/api/v1/write"

        // basic_auth {
        //   username = "admin"
        //   password = "admin"
        // }
    }
}
```

This final component creates a `prometheus.remote_write` component named `metrics_service` that points to `http://localhost:9090/api/v1/write`.

This completes the simple configuration pipeline.

Monitoring. This is the Way?

Microsoft SQL Server

telegraf influxdb Grafana

Collect Store Display

Monitoring. This is the Way?

telegraf

Collect


```
# Read metrics from Microsoft SQL Server
[[inputs.sqlserver]]

  servers = [

"Server=192.168.1.10;Port=1433;User Id=<user>;Password=<pw>;app name=telegraf;log
=1;"
,
  ]

  database_type = "SQLServer"


## A list of queries to include. If not specified, all the below listed queries a
re used.
  include_query = []

  ## A list of queries to explicitly ignore.
  exclude_query = ["SQLServerAvailabilityReplicaStates",
"SQLServerDatabaseReplicaStates"]
```

```
## Queries enabled by default for database_type = "SQLServer" are -
## SQLServerPerformanceCounters, SQLServerWaitStatsCategorized, SQLServerDatabase
IO, SQLServerProperties, SQLServerMemoryClerks,
## SQLServerSchedulers, SQLServerRequests, SQLServerVolumeSpace, SQLServerCpu, SQ
LServerAvailabilityReplicaStates, SQLServerDatabaseReplicaStates,
## SQLServerRecentBackups

## database_type =  SQLServer by default collects the following queries
## - SQLServerPerformanceCounters
## - SQLServerWaitStatsCategorized
## - SQLServerDatabaseIO
## - SQLServerProperties
## - SQLServerMemoryClerks
## - SQLServerSchedulers
## - SQLServerRequests
## - SQLServerVolumeSpace
## - SQLServerCpu
## - SQLServerRecentBackups
## and following as optional (if mentioned in the include_query list)
## - SQLServerAvailabilityReplicaStates
## - SQLServerDatabaseReplicaStates
```

Monitoring. This is the Way?

telegraf

Collect

Microsoft SQL Server

TELEGRAF

INFLUXDB

GRAFANA

Windows Server

```
[[inputs.win_perf_counters]]
  [[inputs.win_perf_counters.object]]
    # Processor usage, alternative to native, reports on a per core.
    ObjectName = "Processor"
    Instances = ["*"]
    Counters = [
      "% Idle Time",
      "% Interrupt Time",
      "% Privileged Time",
      "% User Time",
      "% Processor Time",
    ]
    Measurement = "win_cpu"
    # Set to true to include _Total instance when querying for all (*).
    #IncludeTotal=false
  [[inputs.win_perf_counters.object]]
    # Disk times and queues
    ObjectName = "LogicalDisk"
    Instances = ["*"]
    Counters = [
      "% Idle Time",
      "% Disk Time","% Disk Read Time",
      "% Disk Write Time",
      "% User Time",
      "Current Disk Queue Length",
    ]
    Measurement = "win_disk"
    # Set to true to include _Total instance when querying for all (*).
    #IncludeTotal=false
```

cisco Catalyst

```
[[inputs.snmp]]
  agents = ["192.168.10.11", "192.168.10.12"]
  version = 2
  community = "super_string"
  name = "snmp"
  interval = "60s"

[[inputs.snmp.field]]
   name = "hostname"
   oid = "RFC1213-MIB::sysName.0"
   is_tag = true

[[inputs.snmp.table]]
   name = "snmp"
   inherit_tags = [ "hostname" ]
   oid = "IF-MIB::ifXTable"

  [[inputs.snmp.table.field]]
     name = "ifName"
     oid = "IF-MIB::ifName"
     is_tag = true
```

Monitoring. This is the Way?

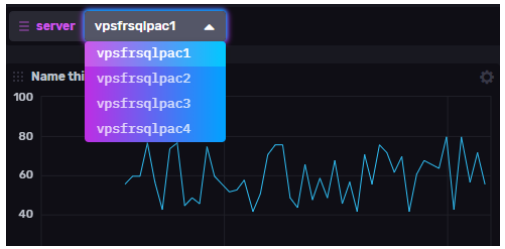
Store

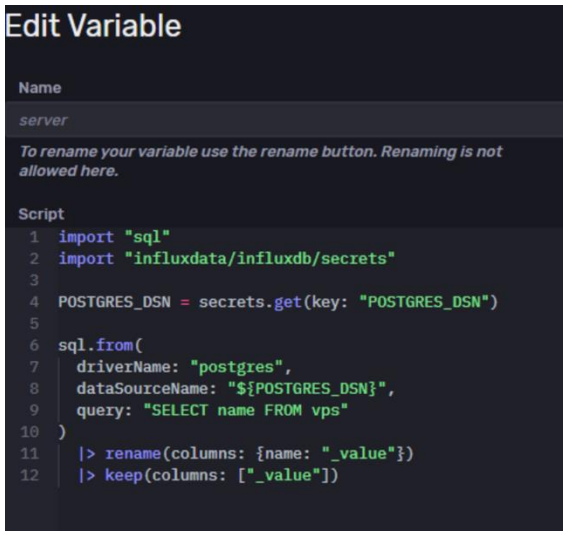
■ Querying SQL databases, sql.from

Very easy to query SQL databases using `sql` package and the `from` function, the code does not need comments :

```
import "sql"
import "influxdata/influxdb/secrets"

POSTGRES_DSN = secrets.get(key: "POSTGRES_DSN")

sql.from(
  driverName: "postgres",
  dataSourceName: "${POSTGRES_DSN}",
  query: "SELECT name, totalmemory, \"date creation\" FROM vps WHERE name like \"vps
)
```

```
Result: _result
Table: keys: []
          name:string              totalmemory:int              date creation:time
----------------------  --------------------------  ------------------------------
          vpsfrsqlpac1                        4000  2020-09-18T00:00:00.000000000Z
          vpsfrsqlpac2                        2000  2020-09-25T00:00:00.000000000Z
          vpsfrsqlpac3                        2000  2020-09-29T00:00:00.000000000Z
```





- InfluxDB time series data are enriched and dashboard variables are populated with reference data coming from SQL databases using `sql.from`.
- Aggregated data are stored for long term purposes in SQL databases using `sql.to`. Mechanism is described.

Monitoring. This is the Way?

Grafana

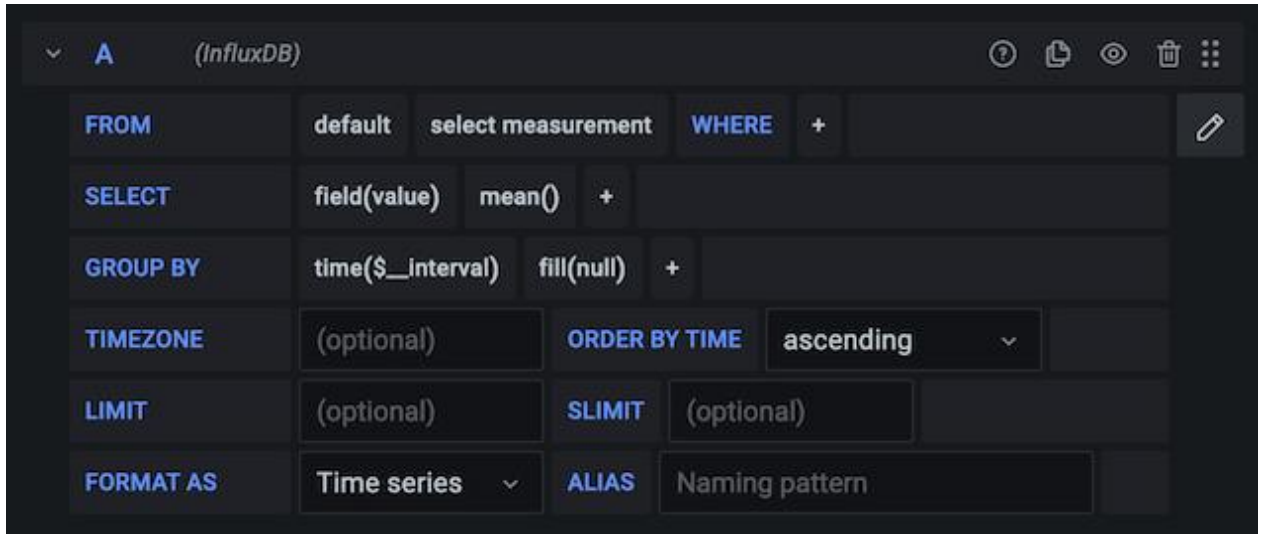
Display

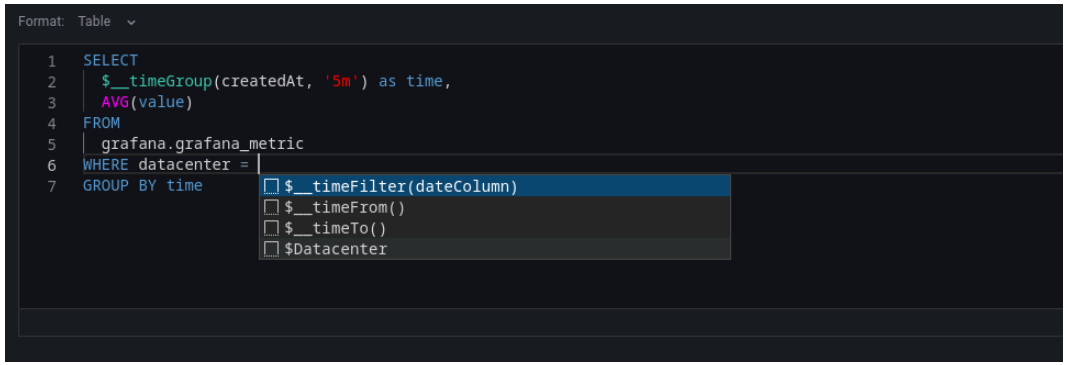
Microsoft SQL Server

TELEGRAF

INFLUXDB

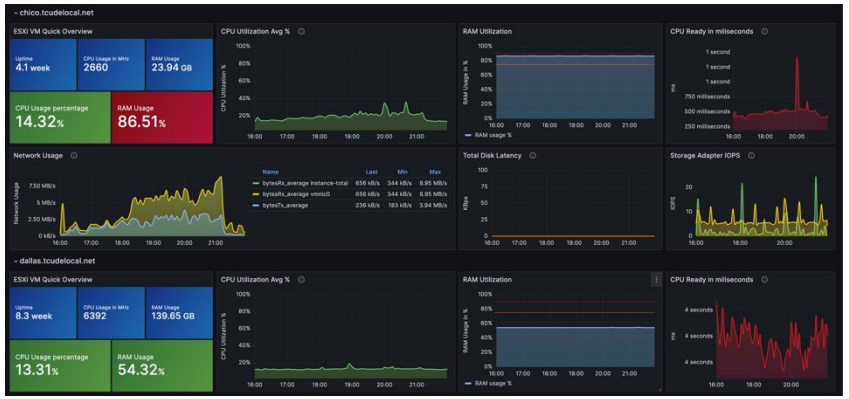
GRAFANA

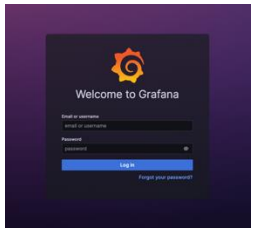





```sql
SELECT derivative(mean("value"), 10s) / 10 AS "REQ/s"
FROM....
```

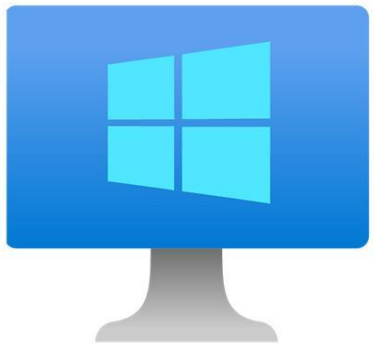
Copy





Monitoring. This is the Way?

Microsoft SQL Server



influxdb Grafana

Collect Store Display

Kapacitor

RESILIENCY

Monitoring. This is the Way?

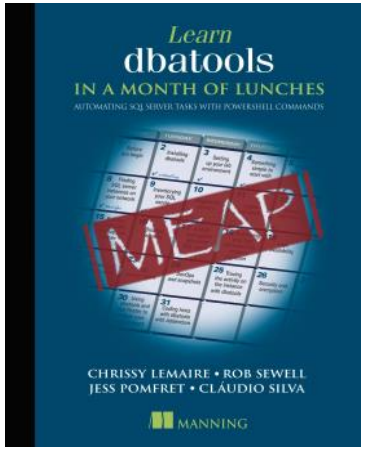
PowerShell

Microsoft SQL Server

TELEGRAF INFLUXDB GRAFANA

dbatools

dbatools




```powershell
try {
    $t = Get-CimInstance MSAcpi_ThermalZoneTemperature -Namespace "root/wmi" -
ErrorAction Stop | Where-Object -Property instancename -EQ "ACPI\ThermalZone\TZ01_0" |
Select-object -first 1
    $currentTempKelvin = $t.CurrentTemperature / 10
    $currentTempCelsius = $currentTempKelvin - 273.15
    $currentTempCelsius.ToString() + "°C"
}
Catch {
    Write-Host "Could not get CPU temperature. Your manafacturer might not support this.
"
}
```

Monitoring. This is the Way?

PowerShell

Microsoft SQL Server

TELEGRAF INFLUXDB GRAFANA

```powershell
$Results = invoke-restmethod -uri "
https://azportscan.azurewebsites.net/api/PortScanner?ports=1234,8080,3389,3333"
$OpenPorts = $Results | Where-Object { $_.status -eq "open"}
$ClosedPorts = $Results | Where-Object { $_.status -eq "closed"}

if(!$OpenPorts) {
$PortScanResult = "Healthy"
} else {
$PortScanResult = $OpenPorts
}
```

Monitoring. This is the Way?

PowerShell

Microsoft SQL Server

TELEGRAF INFLUXDB GRAFANA

```
import-module SQLPS
$Instances = Get-ChildItem "SQLSERVER:\SQL\$($ENV:COMPUTERNAME)"
foreach ($Instance in $Instances) {
    $databaseList = get-childitem "SQLSERVER:\SQL\$($ENV:COMPUTERNAME)\$($Instance.Displayname)\Databases"
    $SkipDatabases = @("Master","Model","ReportServer","SLDModel.SLDData")
    $Errors =  foreach ($Database in $databaselist | Where-Object {$_.Name -notin $SkipDatabases}) {
        if ($Database.status -ne "normal") {"$($Database.name) has the status: $($Database.status)" }
        if ($Database.RecoveryModel -ne "Simple") {  "$($Database.name) is in logging mode $($
Database.RecoveryModel)" }
        if ($database.filegroups.files.MaxSize -ne "-1") { "$($Database.name) has a Max Size set." }
        if ($database.filegroups.files.filename -contains "C:") { "$($Database.name)
 is located on the C:\ drive." }
    }
}
if (!$errors) { $HealthState = "Healthy" } else { $HealthState = $Errors }
```

Monitoring. This is the Way?

PowerShell

Microsoft SQL Server

TELEGRAF INFLUXDB GRAFANA

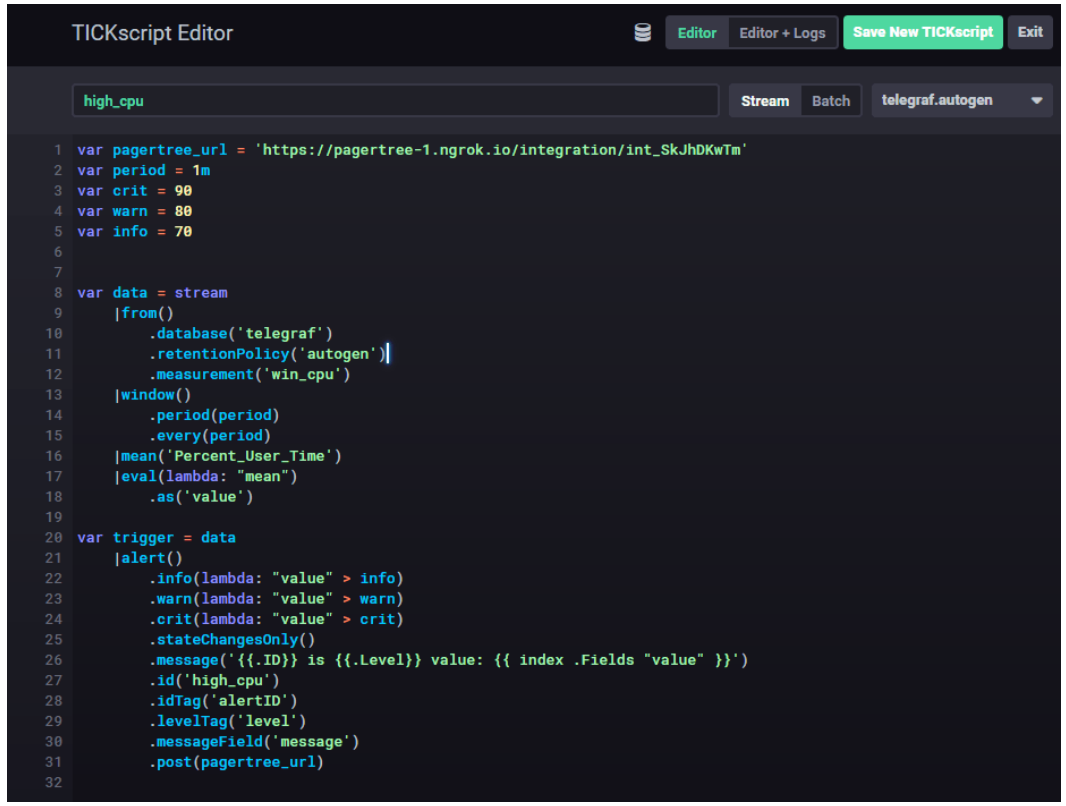
```powershell
function Write-InfluxDB {
    param (
        [string]$measurement, [hashtable]$fields, [string]$token, [string]$InfluxDBUrl
    )

    # Placeholder for InfluxDB logic
    Write-Output "InfluxDB: $measurement - $fields"
    $hostname = $env:COMPUTERNAME
    $headers = @{
        Authorization = "Token $token"
        Accept = "application/json"
    }
    $data = "$measurement,host=$hostname " + "accessible=$($fields.Accessible)," + "load_time=$($fields.LoadTime)" `
        + ",matches=$($fields.Matches)"
    Invoke-RestMethod -Uri "$InfluxDBUrl/api/v2/write?org=pve&bucket=pve&precision=ms" -Method Post `
    -Body $data -ContentType "text/plain; charset=utf-8" -Headers $headers
}
```

Monitoring. This is the Way?

Kapacitor

Microsoft SQL Server

TELEGRAF INFLUXDB GRAFANA




```
var key = 'battery:capacity'

stream
    |from()
        .measurement('battery')
    |eval(lambda: string("value"))
        .as('power')
    |alert()
        .id('{{ .Name }}/{{ .Group }}')
        .message('hello there')
        .crit(lambda: "value" < 0.05)
        .log('/tmp/alerts.log')
        .exec('alert.sh', key, power)
```

Goals to Arrive – What’s the right way?

- Minimizzare la complessità di realizzo e mantenimento.
- Limitare la fragilitá dell’infrastruttura per P.o.F.
- Stima dell’impronta dell’agent sulle risorse monitorate.
- Rapporto Costi/Benefici.
- Documentare i risultati e condividerli con la community.
- Vantaggi/Svantaggi rispetto le soluzioni commerciali

My Goals, Your Takeaways

My Goals, Your Takeaways

“

- *Costruire un trend di riferimento, per **individuare** più velocemente le **anomalie**.*
- *Anticipare i blocchi, per **pianificare** le attività di **manutenzione**.*
- *Sfruttare l’**OpenSource**, per ‘**ritagliarmi su misura**’ una soluzione adatta alle mie esigenze.*
- ***Sperimentare** per accrescere la mia **consapevolezza** su necessità e problemi.*

”

Grazie!!!