Central Workload Health Dashboards

Deployment Guide

5/12/2023

### 

[Azure Architecture 2](#_Toc152681537)

[Prerequisites 3](#_Toc152681539)

[Azure Log Collection Settings 3](#_Toc152681540)

[Provision Managed Grafana 4](#_Toc152681541)

[Resource Health Retriever Function App 5](#_Toc152681542)

[How HTTP Client calls Function App 7](#_Toc152681543)

[Import Grafana Dashboards 9](#_Toc152681544)

[Abbreviation 9](#_Toc152681545)

# Azure Architecture

### A diagram of a diagram Description automatically generated

# Prerequisites

## Azure Log Collection Settings

Grafana dashboard require the following logs to be collected by Azure Monitor

* Collect Performance Counters from VMs using [Create Data Collection Rule](https://learn.microsoft.com/en-us/azure/azure-monitor/agents/data-collection-rule-azure-monitor-agent?tabs=portal)
  + “Resources” including all monitored VMs
  + “Data Source” including Performance Counter – check all perf-counters
  + “Destination” to a single identified Log Analytics Workspace
* Migrate existing App Insights to [Workspace-based](https://learn.microsoft.com/en-us/azure/azure-monitor/app/convert-classic-resource), ensure all App Insights are integrated to a single Log Analytics Workspace

Info: App Insights Log tables on Log Analytics Workspace as shown below.

All logs from multiple App Insights are stored within these tables filterable by column “AppRoleName”

A screenshot of a computer

Description automatically generated

* Create [App Insights Standard Tests](https://learn.microsoft.com/en-us/azure/azure-monitor/app/availability-standard-tests) to perform availability tests to all App Services under CWHD context.

Info: Standard Tests logs are stored in AppAvailabilityResults

A screenshot of a computer

Description automatically generated

* Enable APIM Diagnostic logs for “all log types” to same Log Analytics Workspace integrated with all App Insights as described above.

## Provision Managed Grafana

* [Create an instance of Managed Grafana](https://learn.microsoft.com/en-us/azure/managed-grafana/quickstart-managed-grafana-portal) and [add Grafana Admin permission](https://learn.microsoft.com/en-us/azure/managed-grafana/how-to-authentication-permissions?tabs=azure-portal) for team members who are Grafana admins
* Enable [Deterministic Outbound IP](https://learn.microsoft.com/en-us/azure/managed-grafana/how-to-deterministic-ip?tabs=portal#enable-deterministic-outbound-ips)

A screenshot of a computer

Description automatically generated

# 

# Resource Health Retriever Function App

Private GitHub repository of Function App can be found [here](https://github.com/weixian-zhang/GCC-CWHD).

There is a current limitation to using [Resource Graph query](https://learn.microsoft.com/en-us/azure/service-health/resource-graph-samples?tabs=azure-cli#resource-health) to get resource health and that is query only supports Virtual Machine and VM Scale Sets resource type, other resources are not supported.

The workaround is to use [Resource Health API](https://learn.microsoft.com/en-us/rest/api/resourcehealth/availability-statuses/get-by-resource?view=rest-resourcehealth-2022-10-01&tabs=HTTP) (RH API)to query availability status for all [supported service types](https://learn.microsoft.com/en-us/azure/service-health/resource-health-checks-resource-types). Resource Health API requires OAuth authentication, which means callers of RH API need to support OAuth authn flow, while Grafana Json API plugin does not support OAuth authn flow but only Bearer Access Token input. Manually setting access token in Json API plugin is not practical as access token is short live and this manual step as to be done often. Hence the reason for building an additional Function App.

#### Create and configure Function App

* Required App settings
  + AzureWebJobsStorage: “{connection string of Azure Storage}”
  + SCM\_DO\_BUILD\_DURING\_DEPLOYMENT: “true”

(important if using [Zip Deploy](https://learn.microsoft.com/en-us/azure/azure-functions/deployment-zip-push))

* + WorkspaceID: “{log analytics workspace Id}”

APPLICATIONINSIGHTS\_CONNECTION\_STRING: “{ app insights connection string }”

* + HealthStatusThreshold: “{

"metricUsageThreshold": {

"vm": {

"cpuUsagePercentage": 80,

"memoryUsagePercentage": 80,

"diskUsagePercentage": 80

}

}

}”

* Enable “managed identity” and assign it as Reader role to the Subscription(s) (one or more) containing Azure resources used by system-under-monitor.
* In Network -> Access Restriction -> add the 2 Grafana deterministic outbound Public IPs

#### Deploy Function App

* Ignore this step if any DevOps tool is used.

Using [Zip Deploy](https://learn.microsoft.com/en-us/azure/azure-functions/deployment-zip-push) method:

* + A Python deployment script is available at src\tools\zip\_and\_deploy\_func\_app.py to simplify the Zip Deployment process.

This script will gather all .py files and create or override func-app.zip in deploy directory and execute Azure cli cmd “az functionapp deployment source config-zip -g {resource group of function} -n {function name} --build-remote --src 'func-app.zip'”

run cmd: “python zip\_and\_deploy\_func\_app.py”

## Troubleshoot

### OpenTelemetry Tracing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

OpenTelemetry tracing is implemented, you can view traces and their spans in Application Insights under **Transaction Search** -> filter by **Request**.

All API calls to Azure Resource Health API and Azure Monitor Log Query API are captured as spans.

Tracing is useful for troubleshooting slowness or exceptions when invoking these downstream APIs.

### Error and Warning Logs

A screenshot of a computer

Description automatically generated

Errors and warnings are log under **Trace** in Application Insights

# How HTTP Client calls Function App

Sample [HTTP request message](https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages) call from any HTTP client as well as Grafana Json API plugin

HTTP POST https://{function name}.azurewebsites.net/api/RHRetriever

x-functions-key: { function key }

{

"resources": [

{

"subscriptionId": "**nnn**",

"resourceId": "subscriptions/**nnn**/resourceGroups/**nnn**/providers/Microsoft.Web/sites/**nnn**"

},

{

"subscriptionId": "**nnn**",

"resourceId": "subscriptions/**nnn**/resourceGroups/**nnn**/providers/Microsoft.Web/sites/**nnn**"

},

{

"subscriptionId": "**nnn**",

"resourceId": "subscriptions/**nnn**/resourceGroups/**nnn**/providers/Microsoft.Web/sites/**nnn**"

},

…

]

}

Also see sample POSTMAN configuration

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

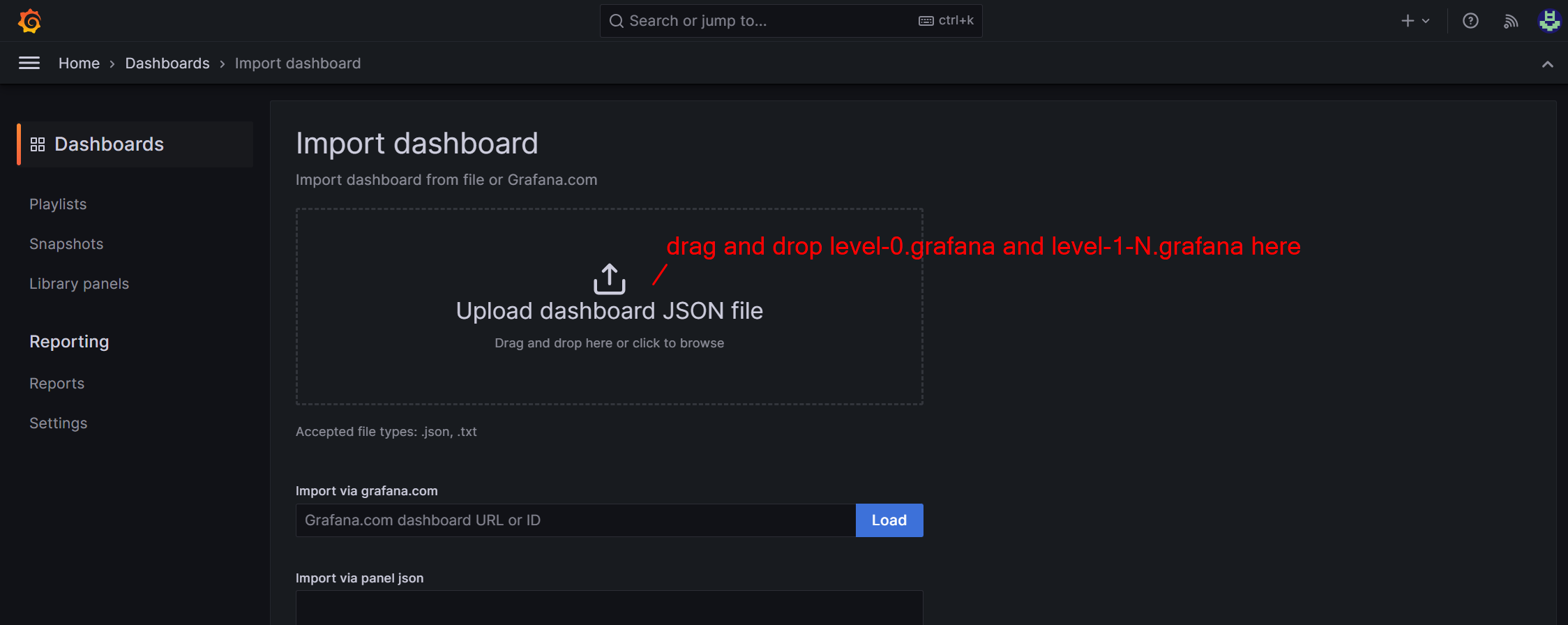
# Import Grafana Dashboards

Dashboards are in [private GitHub repository](https://github.com/weixian-zhang/GCC-CWHD/tree/main/src/dashboards) and have .grafana as file extension in Json format.

They can be imported into Grafana using the following way.

A screenshot of a computer

Description automatically generated



# Abbreviation

* CWHD – Central Workload Health Dashboards