Article <u>Mikhail Khomenko</u> · Oct 11, 2021 9m read

Using Cloud Monitoring to Monitor IRIS-Based Applications Deployed in GKE

In this article, we 'II look at one of the ways to monitor the InterSystems IRIS data platform (IRIS) deployed in the <u>Google Kubernetes Engine</u> (GKE). The GKE integrates easily with <u>Cloud Monitoring</u>, simplifying our task. As a bonus, the article shows how to display metrics from Cloud Monitoring in <u>Grafana</u>.

Note that the Google Cloud Platform used in this article is not free (price list), but you can leverage a free tier. This article assumes that you already have a project in the Google Cloud Platform (referred to as <yourprojectid>) and have permission to use it.

We ' II use the following tools:

- gcloud version Google Cloud SDK 346.0.0
- kubectl version v1.17.12
- k9s version v0.24.14 (just for fun)
- <u>helm</u> version v3.6.3
- <u>helmfile</u> version v0.139.9

GKE Creation

Let 's start with Kubernetes cluster creation. <u>Aegional private cluster</u> looks good. Let 's also create network address translation (NAT) to allow traffic from nodes to the Internet. Initialize connection to Google Platform:

```
$ gcloud init
$ gcloud auth login
$ gcloud projects list
$ gcloud config set project <your_project_id>
```

Create a dedicated network:

```
$ gcloud compute networks create kubernetes --subnet-mode custom
$ gcloud compute networks subnets create kubernetes --network kubernetes --range 10.2
0.0.0/24 --region us-eastl --secondary-range pods=10.42.0.0/16,services=172.20.0.0/16
```

Assign a couple of static IPs (for NAT and Grafana endpoint, see below):

```
$ gcloud compute addresses create kubernetes --region=us-east1
$ gcloud compute addresses create kubernetes-grafana --region=us-east1
```

Using Cloud Monitoring to Monitor IRIS-Based Applications Deployed in GKE Published on InterSystems Developer Community (https://community.intersystems.com)

Create NAT:

```
$ gcloud compute routers create kubernetes --network=kubernetes --region us-east1
$ gcloud compute routers nats create kubernetes --router=kubernetes --nat-all-subnet-
ip-ranges --nat-external-ip-pool=kubernetes --region=us-east1
```

Finally create GKE. Kubernetes version is 1.20.8-gke.900.

Last line "enable-stackdriver-kubernetes" means that metrics, logs and events from the cluster will be gathered and sent to Cloud Monitoring – see <u>Configuring Cloud Operations for GKE</u>:

```
$ gcloud container clusters create kubernetes
                                                  \backslash
  --region us-east1
                                                                     /
 --machine-type e2-small
 --num-nodes 1
  --enable-private-nodes
  --no-enable-master-authorized-networks
                                                   ١
  --cluster-version 1.20.8-gke.900
  --enable-ip-alias
 --cluster-ipv4-cidr 10.24.0.0/14
  --master-ipv4-cidr 10.0.0/28
 --services-secondary-range-name services
  --cluster-secondary-range-name pods
  --no-enable-autoupgrade
  --enable-autorepair
  --network kubernetes
  --subnetwork kubernetes
  --enable-stackdriver-kubernetes
```

Let 's connect to the newly created cluster:

```
$ gcloud container clusters get-credentials kubernetes --region us-
east1 --project <your_project>
```

Default Monitoring

Let 's look at the workloads in our cluster (nice k9s output):

\$ k9s

			- Poo	ds(all)[19] ———							
NAMESPACE 1	NAME	PF	REAL	DY RESTARTS STATUS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R \$	%MEM/L IP	NODE
kube-system	event-exporter-gke-67986489c8-k5lhr	٠	2/2	2 0 Running	1	16	n/a	n/a	n/a	n/a 10.42.2.3	gke-kubernetes-defa
kube-system	fluentbit-gke-lj55b	•	2/2	2 0 Running		20		n/a	10	4 10.20.0.2	gke-kubernetes-defa
kube-system	fluentbit-gke-pt756	٠	2/2	2 0 Running		20		n/a	10	4 10.20.0.4	gke-kubernetes-defa
kube-system	fluentbit-gke-zvrbs	۲	2/2	2 0 Running		21		n/a	10	4 10.20.0.3	gke-kubernetes-defa
kube-system	gke-metrics-agent-q6xkx	۲	1/1	0 Running		28	66	n/a	57	57 10.20.0.2	gke-kubernetes-defa
kube-system	gke-metrics-agent-qlb8p	٠	1/1	0 Running		29	33	n/a	58	58 10.20.0.4	gke-kubernetes-defa
kube-system	gke-metrics-agent-wqzb5	٠	1/1	0 Running		30	66	n/a	60	60 10.20.0.3	gke-kubernetes-defa
kube-system	kube-dns-6c7b8dc9f9-hhgkh	۲	4/4	0 Running		23		n/a	21	n/a 10.42.0.6	gke-kubernetes-defa
kube-system	kube-dns-6c7b8dc9f9-ncjck	۲	4/4	0 Running		22		n/a	20	n/a 10.42.1.3	gke-kubernetes-defa
kube-system	kube-dns-autoscaler-844c9d9448-zsl78	٠	1/1	0 Running				n/a	48	n/a 10.42.1.2	gke-kubernetes-defa
kube-system	kube-proxy-gke-kubernetes-default-pool-0a363374-znrf	٠	1/1	0 Running		19		n/a	n/a	n/a 10.20.0.4	gke-kubernetes-defa
kube-system	kube-proxy-gke-kubernetes-default-pool-78738a67-vj5q	٠	1/1	0 Running		18		n/a	n/a	n/a 10.20.0.3	gke-kubernetes-defa
kube-system	kube-proxy-gke-kubernetes-default-pool-ad61c69c-d9d9	۲	1/1	0 Running		19		n/a	n/a	n/a 10.20.0.2	gke-kubernetes-defa
kube-system	l7-default-backend-56cb9644f6-gqsln	٠	1/1	0 Running			10	10		9 10.42.0.2	gke-kubernetes-defa
kube-system	<pre>metrics-server-v0.3.6-9c5bbf784-pff84</pre>	٠	2/2	2 0 Running	18	26	37	12	25	7 10.42.2.4	gke-kubernetes-defa
kube-system	pdcsi-node-7sp77	٠	2/2	2 0 Running			n/a	n/a	40	8 10.20.0.3	gke-kubernetes-defa
kube-system	pdcsi-node-httpw	٠	2/2	2 0 Running			n/a	n/a	39	7 10.20.0.4	gke-kubernetes-defa
kube-system	pdcsi-node-zlh6p	۲	2/2	2 0 Running			n/a	n/a	40	8 10.20.0.2	gke-kubernetes-defa
kube-system	<pre>stackdriver-metadata-agent-cluster-level-bd98f4674-95ggt</pre>	٠	2/2	2 0 Running	12	27	12	25	13	13 10.42.0.5	gke-kubernetes-defa

We see three pods (one for each node). These pods are managed by a <u>DaemonSet</u>, and called by the Google Kubernetes Engine Metrics Agent (gke-metrics-agent). Under the hood, this agent leverages the <u>Open Telemetry Collector</u> enhanced by Google-specific features. You can spend some time looking at its <u>configuration</u>, which is thoroughly described on the page <u>OpenTelemetry Collector Configuration Format</u>.

Because of the gke-metrics-agent, we ' re immediately able to see a lot of useful GKE-related metrics on a dashboard in Cloud Monitoring:

÷	Monitoring		← GKE dashbo	oard - SEND	FEEDBACK								Auto-r	refresh	
:•	Metrics Scope 1 Project	>	add filter	Cluster: kubernetes 🗙							RESET 💡	Configure	e resource tables		•
á	Overview		Timeline 0 Ale	rts 0 Info events	0 Warning events Ti	ime selection is 14 Aug 17:3	3 - 18:33			RESET	Time Span 1 hour	•	🔚 Refine by status	•	^
H	Dashboards		> 0 alerts											in	
•°	Services		0 Kubarpatas												
th	Metrics explorer		events												
	Alerting			Aug 14, 17:33 1	7:40	17:50		18:00		18:10	18:20		1 Aug 14, 18:33		
۲	Uptime checks														
ធា	Groups		Clusters No acti	ve alerts NUM_ENTITIES,	plural, =0{0 {PLURAL_ENTITY	_NAME} with active alerts=1	(1 (SINGUI	LAR_ENTITY_NAME} with ac	tive alerts				VIEW ALL	ш	^
			Name	Alerts 😯	Labels			Container restarts 💡	Error logs	CPU utilisation ?	Memory u	tilisation 😧	Disk utilisation 💡		
			KULCHIEKE3	Ū		појесс диприска		U	193	02.00% 01101010	15.7440	1.55 018	0.901900.42010		
													1 – 1 of 1	< :	>
			Namespaces N	to active alerts NUM_ENT	TTIES,plural, =0{0 {PLURAL_E	NTITY_NAME} with active al	erts=1{1 (S	SINGULAR_ENTITY_NAME} v	with active alerts				VIEW ALL	ш	^
			Name	Alerts 🚱	Labels			Container restarts 😨	Error logs	CPU utilisation 🕑	Memory	utilisation 🔞	Disk utilisation 🔞)	
			default	0	Cluster: kubernetes	Location: us-east1	+1	0	0	0 CPU 🕑	ов 🚱		ов 🚱		
			kube-node-lease	0	Cluster: kubernetes	Location: us-east1	+1	0	0	O CPU 😧	0В 🚱		0В 🕜		
			kube-public	0	Cluster: kubernetes	Location: us-east1	+1	0	0	O CPU	ов 😧		0В 😢		
Ē	Release notes		kube-system	0	Cluster: kubernetes	Location: us-east1	+1	0	184	32.85% of 1.31 CPU	19.74% (of 1.33 GiB	0% of 958.42 GiB	3	

To add IRIS metrics (we do not have IRIS in the cluster yet), we deploy an application similar to the gkemetrics-agent, but with our configuration.

A Note About Application Installation

A common way to deploy applications in Kubernetes clusters is by using Helm. For this article, several Helm charts were prepared and published in a custom public <u>Helm repository</u>. You can see how to do this on <u>GitHub</u> and in <u>this guide that reviews how to make and share your Helm package</u>. You need to have a Helm chart in a directory for each application according to <u>The Chart Template Developer</u> 's <u>Guide</u> dike simple-iris. Then, we package a chart:

Then move a resulting gzipped archive into a repository directory (gh-pages branch). Then update the index-file:

\$ helm repo index . --url <u>https://myardyas.github.io/helm-charts/</u>

Push the updated index-file as well as the archive into GitHub.

We just created the charts for the gke-metrics-agent and IRIS itself. IRIS is a stateless application for our monitoring purpose, so for simplicity, we use the Deployment command instead of Statefulset.

Let 's look at the available charts:

```
$ helm repo add iris-monitoring <a href="https://myardyas.github.io/helm-charts/">https://myardyas.github.io/helm-charts/</a>
$ helm repo update
$ helm search repo iris-monitoring --versions --devel --max-col-width 35
NAME
                                                       CHART VERSION
APP VERSION
                      DESCRIPTION
iris-monitoring/gke-metrics-agen... 0.1.0
1.0.3-gke.0
                              GKE metrics agent chart (based o...
iris-monitoring/metrics-proxy
                                            0.1.0
7.70.0
                                 Service Chart to add a default m...
iris-monitoring/simple-iris
                                                0.1.0
2021.1.0.215.0
                          Simple stateless IRIS Community ...
```

Here we see yet another chart called metrics-proxy. IRIS exposes Prometheus metrics without metrics types, leading to the gke-metrics-agent complaining about an 'UNSPECIFIED " type. As a workaround, we send metrics from IRIS to the gke-metrics-agent using metrics-proxy, whose only job is to add a default " gauge " type. You can view the source code of the charts easily in this way:

```
$ helm fetch --untar iris-monitoring/simple-iris --version 0.1.0
$ cd simple-iris # browse files here
```

Install IRIS and Monitoring Tools into a Cluster

Having prepared the charts, we can install them. A common way to install Helm charts into clusters is helm install or helm upgrade --install. But this time, let 's try a more declarative way to do the same thing using <u>helmfile</u>.

Prepare the following file, helmfile.yaml:

```
$ cat helmfile.yaml
repositories:
- name: iris-monitoring
url: https://myardyas.github.io/helm-charts
- name: grafana
url: https://grafana.github.io/helm-charts
```

```
helmDefaults:
  wait: true
  timeout: 600
  force: false
  historyMax: 5
  createNamespace: true
  atomic: true
releases:
- name: simple-iris
  chart: iris-monitoring/simple-iris
  namespace: iris
  version: 0.1.0
- name: gke-metrics-agent-custom
  chart: iris-monitoring/gke-metrics-agent-custom
  namespace: iris
  version: 0.1.0
- name: metrics-proxy
  chart: iris-monitoring/metrics-proxy
  namespace: iris
  version: 0.1.0
- name: grafana
  chart: grafana/grafana
  namespace: grafana
  version: 6.15.0
  values:
  - grafana-values.yaml
  set:
  - name: service.loadBalancerIP
    value: {{ requiredEnv "LOAD_BALANCER_IP" }}
```

This code contains a list of repositories from which we ' re going to install some charts. Here are our IRISmonitoring repository and a public Grafana repository (we ' II use Grafana soon, so you need to install it too).

Besides repositories, the helmfile.yaml file sets default values for things like namespaces if they don 't exist. The most important part of the helmfile is a releases list. The release list is just an installed Helm chart. Each release contains a chart name, namespace, and version. The Grafana release list also sets a custom IP address (to reach the Grafana UI) and provisions a data source and dashboard in the grafana-values.yaml file. We must create this file as follows:

```
$ cat grafana-values.yaml
service:
  type: LoadBalancer
datasources:
  datasources.yaml:
    apiVersion: 1
    datasources:
    - name: Google Cloud Monitoring
    type: stackdriver
    isDefault: true
    jsonData:
        authenticationType: gce
```

```
editable: true
      readOnly: true
dashboardProviders:
  dashboardproviders.yaml:
    apiVersion: 1
    providers:
    - name: 'default'
      orqId: 1
      folder: ''
      type: file
      disableDeletion: false
      editable: true
      options:
        path: /var/lib/grafana/dashboards/default
dashboards:
  default:
    iris:
      gnetId: 14869
      revision: 1
      datasource: Google Cloud Monitoring
```

Note that the <u>dashboard has the number 14869</u>. This dashboard contains several panels with metrics, like those shown by the <u>System Alerting and Monitoring Guide</u> (SAM).

Now we ' re almost ready to run an installation procedure. The final step is to define an IP address to access Grafana (replace <yourprojectid> with your project ID):

```
$ export LOAD_BALANCER_IP=$(gcloud compute addresses list --project <your_project_id>
    --filter="name~'kubernetes-grafana'" --format="value(Address)")
$ echo $LOAD_BALANCER_IP
x.x.x
```

Let 's finally run an installation:

\$ helmfile sync
\$ k9s

|--|

			- Pod	s(all)[23]	_									<u></u>
NAMESPACE	NAME	PF	READ	Y RESTARTS	S	TATUS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE
grafana	grafana-65fff4c848-mt9zs	•	1/1	θ	R	unning	0	θ	n/a	n/a	n/a	n/a	10.42.1.5	gke-kubernetes-def
iris	gke-metrics-agent-custom-5cfb9d44d-8h5gm	٠	1/1Δ	0	R	unning∆	0	0	0	0	0	0	10.42.2.8	gke-kubernetes-def
iris	metrics-proxy-5568969545-5gwmc	٠	1/1Δ		R	unning∆			Θ		0		10.42.2.10	gke-kubernetes-def
iris	simple-iris-6d5d5c85b8-bmfc8		0/1			unning							10.42.2.9	gke-kubernetes-def
kube-system	event-exporter-gke-67986489c8-k5lhr	۲	2/2		R	unning		15	n/a	n/a	n/a	n/a	10.42.2.3	gke-kubernetes-def
kube-system	fluentbit-gke-lj55b	۲	2/2		R	unning		20		n/a	10		10.20.0.2	gke-kubernetes-def
kube-system	fluentbit-gke-pt756	۲	2/2		R	unning		24		n/a	12		10.20.0.4	gke-kubernetes-def
kube-system	fluentbit-gke-zvrbs	۲	2/2		R	unning	2	22	2	n/a	11	4	10.20.0.3	gke-kubernetes-def
kube-system	gke-metrics-agent-q6xkx	٠	1/1		R	unning		25	100	n/a	51	51	10.20.0.2	gke-kubernetes-def
kube-system	gke-metrics-agent-qlb8p	٠	1/1		R	unning		31	100	n/a	62	62	10.20.0.4	gke-kubernetes-def
kube-system	gke-metrics-agent-wqzb5	٠	1/1	Θ	R	unning	2	32	66	n/a	65	65	10.20.0.3	gke-kubernetes-def
kube-system	kube-dns-6c7b8dc9f9-hhgkh	۲	4/4	Θ	R	unning	2	21	0	n/a	19	n/a	10.42.0.6	gke-kubernetes-def
kube-system	kube-dns-6c7b8dc9f9-ncjck	۲	4/4	Θ	R	unning	2	22	0	n/a	20	n/a	10.42.1.3	gke-kubernetes-def
kube-system	kube-dns-autoscaler-844c9d9448-zsl78	۲	1/1	Θ	R	unning		10	5	n/a	103	n/a	10.42.1.2	gke-kubernetes-def
kube-system	kube-proxy-gke-kubernetes-default-pool-0a363374-znrf	۲	1/1		R	unning	1	23	1	n/a	n/a	n/a	10.20.0.4	gke-kubernetes-def
kube-system	kube-proxy-gke-kubernetes-default-pool-78738a67-vj5q	۲	1/1		R	unning		27	1	n/a	n/a	n/a	10.20.0.3	gke-kubernetes-def
kube-system	kube-proxy-gke-kubernetes-default-pool-ad61c69c-d9d9	۲	1/1		R	unning	2	17	2	n/a	n/a	n/a	10.20.0.2	gke-kubernetes-def
kube-system	l7-default-backend-56cb9644f6-gqsln	٠	1/1		R	unning			10	10	14	14	10.42.0.2	gke-kubernetes-def
kube-system	<pre>metrics-server-v0.3.6-9c5bbf784-pff84</pre>	۲	2/2		R	unning	25	37	52	17	35	10	10.42.2.4	gke-kubernetes-def
kube-system	pdcsi-node-7sp77	۲	2/2		R	unning		8	n/a	n/a	40	8	10.20.0.3	gke-kubernetes-def
kube-system	pdcsi-node-httpw	۲	2/2		R	unning			n/a	n/a	39		10.20.0.4	gke-kubernetes-def
kube-system	pdcsi-node-zlh6p	۲	2/2		R	unning		8	n/a	n/a	40	8	10.20.0.2	gke-kubernetes-def
kube-system	<pre>stackdriver-metadata-agent-cluster-level-bd98f4674-95ggt</pre>	•	2/2	Θ	R	unning	14	29	14	29	14	14	10.42.0.5	gke-kubernetes-def

Visualization in Cloud Monitoring

Note that we didn 't expose the IRIS web port to the world this time. We don 't need it for this article, but you can leverage the port-forward application to check IRIS metrics this way:

\$ kubectl -n iris port-forward svc/simple-iris 52773:52773

Then, in a different terminal, enter the following:

```
$ curl -s http://localhost:52773/api/monitor/metrics
iris_cpu_pct{id="AUXWD"} 0
iris_cpu_pct{id="CSPDMN"} 0
...
```

Having checked these metrics, let 's look in the Cloud Monitoring console Mstrics explorer. Select the Global resource type.



Now look at the irisprocesscount metric:

<u>~</u>	Monitoring	Metrics explorer	
:	Metrics Scope >	CONFIGURATION MQL Image: Construction of the cons	
M	Overview	Resource type 1 min interval (mean)	
Ħ	Dashboards		
•°	Services	custom/opencensus/iris_process_count • X 0	1
- 16	Metrics explorer	Filters @	ò
	Alerting	ADD FILTER 26-	4
٤	Uptime checks	How do you want to view that data?	
(ij)	Groups	26.	-
\$	Settings	Group by	
		Aggregator	8
		UTC+3 0725 0730 0735 0740 0735 0750 0755 0800 0805 0810 0815	
		1 minute Value	đ
		V SHOW ADVANCED OPTIONS -• iris_process_count 26	

We can create dashboards that combine these metrics in a single place. But these dashboards are available only for users of this Google project and aren 't very flexible in their settings. Because of that,

let 's look at IRIS metrics in a much handier tool: Grafana.

Visualization in Grafana

For a connection to Grafana, use the IP-address, x.x.x.x, stored in the LOADBALANCERIP variable:

\$ echo \${LOAD_BALANCER_IP}

😚 Grafana x +		0	- 1	e ×
← → C G xxxx/login			* 6	🧑 E
	Ō			
	Welcome to Grafana			
	Email or username			
	admin			
	Password			
	·····			
	Log in			
	Forgot your password?			

The username is admin.

The password is the output of:

\$ kubectl get secret --namespace grafana grafana -o jsonpath="{.data.adminpassword}" | base64 --decode ; echo

The IRIS dashboard appears.

Using Cloud Monitoring to Monitor IRIS-Based Applications Deployed in GKE Published on InterSystems Developer Community (https://community.intersystems.com)



Any person who knows the address and password can view this dashboard. Feel free to add your own dashboards.

← New dashboard / Edit Panel			Discard Save Apply
		Table view Fill Actual O Last 6 hours Q C	
	Panel Trie		
Cuery 1 53 Transform 0		MD = auto = 164 biterval = 155 Query inspector	✓ Tooltip Tooltip mode Single All Hidden
			 Legend Legend mode
Query type Metrics • Project • • Service Custom • Filter • +	opencensus/ins	28 Edit MQL	List Table Hildden Legend placement Bottom Right Legend values Genet values or calculations to show in legend
Pre-processing O None Group by O Choose label	custom.googleapis.com/opencensus/iris_cpu_pct Auto-created custom metric.		Choose ~
Alignment Function O mean ~ Alignment period	custom geogleapis.com/opencensus/iris_cpu_usage Auto-created custom metric.		Style Lines Bars Points Line Internetiatee

Cleanup

When you ' re done, don ' t forget to remove unused parts from the Google Cloud Platform. You can remove everything this way:

```
$ gcloud --quiet container clusters delete kubernetes --region us-east1
$ gcloud --quiet compute routers nats delete kubernetes --router=kubernetes --region=
us-east1
$ gcloud --quiet compute routers delete kubernetes --region=us-east1
$ gcloud --quiet compute addresses delete kubernetes-grafana --region=us-east1
$ gcloud --quiet compute addresses delete kubernetes --region=us-east1
$ gcloud --quiet compute addresses delete kubernetes --region=us-east1
$ gcloud --quiet compute networks subnets delete kubernetes --region us-east1
$ gcloud --quiet compute networks delete kubernetes --region us-east1
```

Conclusion

We 've shown one of the almost endless approaches to monitoring IRIS applications deployed in GKE. This time we 've focused on metrics stored in Cloud Monitoring and displayed in Grafana. But don 't forget about IRIS logs. Pods logs are, at the moment of writing, gathered by <u>fluentbit</u> and sent to Cloud Logging where they can be viewed.

E	Operations Logging	Logs explorer Options 🗸				GO SHARE LINK	C LAST 1 HOUR	PAGE LAYOUT
Ξ	Logs explorer	 New features are available in the 	Logs explorer.				Dismis	s Learn more 🛛
58	Logs Dashboard	Query Recent (4) Saved (0)	Suggested (5)				Save Strea	n logs Run query 🕺
- th	Logs-based Metrics	Resource v Log name v S	everity ~					
>\$	Logs Router	<pre>1 resource.type = ("k8s_containe 2 resource.labels.namespace_name</pre>	r" OR "k8s_pod") = ("iris")					
	Logs Storage	3 resource.labels.container_name 4 re	= ("simple-iris")					-
		resource.labels.instance_ resource labels location	id					
		Histo resource.labels.namespace resource.labels.namespace resource.labels.pod_name resource.labels.project. resource.labels.project. resource.labels.rome resource.labels.custer_	_name d name					ବ୍ ବ୍ × >
		-resource.labels.containe	_id	08:10	08:20	08:30	. 08:40	15 Aug, 08:47:00
		Query results []					Jump to now	Actions • Configure •
		SEVERITY TIMESTAMP EEST -	SUMMARY					
		 This query has been updated. To vi 	ew matching entries, select Run qu	ary				×
		> i 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:503 (409) 0 [Utility.Event] Processing MagTa	ape section			
		> (1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:504 (409) 0 [Utility.Event] Processing IO se	ection			
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:504 (409) 0 [Utility.Event] Processing SQL :	section			
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:521 (409) 0 [Generic.Event] Auditing to /use	r/irissys/mgr/irisaudit/			
		> (i) 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:523 (417) 0 [Utility.Event] LMF Info: Licens	sed for 5 users.			
		> (1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:535 (409) 0 [Utility.Event] Executing ^ZSTU	routine			
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:536 (409) 0 [Utility.Event] Executing ^%ZST/	ART routine			
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:536 (409) 0 [Utility.Event] Enabling logons				
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:4	3:603 (409) 0 [Database.MountedRW] Mounted dat	tabase /usr/irissys/mgr/user/ (SFN	i) read-write.		
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:980 (418) 0 [Utility.Event] Private webserve	er started on 52773			
Ē	Release notes	> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	3:980 (418) 0 [Utility.Event] Processing Shado	ows section (this system as shadow)			
		> 1 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	4:01/ (418) 0 [Utility.Event] Processing Monif	tor section			
<1		> 0 2021-08-15 08:05:51.830 EEST	simple-iris 08/15/21-05:05:	4:0/2 (56/) 0 [Utility.Event] Starting TASKNGP	<			

You can find examples of queries at <u>Kubernetes-related queries</u>. It 's interesting to show logs in Grafana too. Project <u>Grafana Loki</u> is exactly for that. An example of integration is available at <u>Use Loki to Manage</u> <u>k8s Application Logs</u>.

#DevOps #Kubernetes #InterSystems IRIS

Source

URL: https://community.intersystems.com/post/using-cloud-monitoring-monitor-iris-based-applications-deployed-gke