Docker Container Monitoring With Zabbix

Credit **Dmitry Lambert** Original URL: https://blog.zabbix.com/docker-container-monitoringwith-zabbix/20175/

Importing the official Docker template

Ζ	Templates									Create template
Q	Import							×		
0	Host groups type h * Import	file Choose File templa	ate_app_do	cker.yaml						
õ	Linked templates type h Re	iles	Update exi	sting Create	new Delete mis	sing			alue	Remove
:=	Name	Groups	✓	✓						
		Templates	✓	✓						
		value mappings	✓	✓						
3		remplate dashboards	⊻	×						
	Name 🔺	Template linkage		~						Linked to templates
	AIX	Items	✓	✓						
-		Discovery rules	✓	~						
	Aicatel Timetra TIMOS SNMP	Triggers	\checkmark	~					P, Generic SIMP,	
	Apache ActiveMQ by JMX	Graphs	✓	~						
	Apache by HTTP	Web scenarios	~	~						
	Apache by Zabbix agent						Import	Cancel		
	Apache Cassandra by JMX	Hosts Items 67	Triggers 6	Graphs 7	Dashboards	Discovery 1	Web			
	Apache Kafka by JMX	Hosts Items 62	Triggers 11	Graphs 9	Dashboards	Discovery 3	Web			

Importing the Docker by Zabbix agent 2 template

Since we will be using the official *Docker by Zabbix agent 2* template, first, we need to make sure that the template is actually available in our Zabbix instance. The template is available for **Zabbix versions 5.0, 5.4, and 6.0**. If you cannot find this template under *Configuration – Templates*, chances are that you haven't imported it into your environment after upgrading Zabbix to one of the aforementioned versions. Remember that Zabbix does not modify or import any templates during the upgrade process, so we will have to import the template manually. If that is so, simply download the template and import it into your Zabbix instance by using the Import button in the *Configuration – Templates* section.

Installing and configuring Zabbix agent 2

Before we get started with configuring our host, we first have to install Zabbix agent 2 and configure it according to the template guidelines. Follow the steps in the download section of the

Zabbix website and install the zabbix-agent2 package. Feel free to use any other agent deployment methods if you want to (like compiling the agent from the source files).

Plugin specific Zabbix agent 2 configuration

Zabbix agent 2 provides plugin-specific configuration parameters. Mostly these are optional parameters related to a specific plugin. You can find the full list of plugin-specific configuration

parameters in the Zabbix documentation. In the newer versions of Zabbix agent 2, the pluginspecific parameters are defined in separate plugin configuration files, located in /etc/zabbix/zabbix_agent2.d/plugins.d/, while in older versions, they are defined directly in the zabbix_agent2.conf file.

Before we move on to Zabbix frontend, I would like to point your attention to the Docker socket file permission – the zabbix user needs to have access to the Docker socket file. The zabbix user should be added to the docker group to resolve the following error messages.

[Docker] cannot fetch data: Get http://l.28/info: dial unix /var/run/docker.sock: connect: permission denied

ZBX_NOTSUPPORTED: Cannot fetch data.

You can add the zabbix user to the Docker group by executing the following command:

usermod -aG docker zabbix

Configuring the docker host

Ζ	Hosts												Create host
Q		New host										×	
Ο		Host IPMI Tags	Macros	Inventory Encryption	Value mappin	g							
Ō		* Host name	Docker host										
:=		Visible name	Docker host										
		Templates	Docker by Z	abbix agent 2 🗙			Sele	ect					ve
٩		* Groups	Linux server	rs X			Sele	ect					
8		Interfaces	type here to	search		DNS name		Connect to	Port	Default			
		Interfaces	1ype	11 8001633		Divo name		Sonnect to	T OIL				
	Name 🛦		Agent	192.168.50.141				IP DNS	10050	• Re	emove		Tags
	Anache Wet		Add										-
		Description	Docker cont	ainer host									
	Application I												
	Backroom V												
							h						
		Monitored by proxy	(no proxy)	~									
9		Enabled	~										
Z													
											Add	Cancel	
?								Uy Z	авых адент,		_	•	

Configuring the host representing our Docker environment

After importing the template, we have to create a host which will represent our Docker instance. Give the host a name and assign it to a Host group – I will assign it to the Linux servers host group. Assign the *Docker by Zabbix agent 2* template to the host. Since the template uses Zabbix agent 2 to collect the metrics, we also have to add an agent interface on this host. The address of the interface should point to the machine running your Docker containers. Finish up the host configuration by clicking the Add button.

Docker by Zabbix agent 2 template

Ζ	Iter	ms													Crea
Q	All t	emplate	es / Docker by Zabbix agent 2	Items 44	Triggers 3	Graphs 5	Dashboards	1 Discovery rules	2 Web scenarios						Fi
\odot			Name 🔺				Triggers	Кеу		Interval	History	Trends	Туре	Status	Tags
ā		••• Docker: Get info: Docker: Architecture						docker.architecture			7d		Dependent item	Enabled	Application: Docker
		••••	Docker: Get info: Docker: Cgro	oup driver				docker.cgroup_driver	r		7d		Dependent item	Enabled	Application: Docker
≔		•••	Docker: Get info: Docker: Con	ntainers paus	ed			docker.containers.pa	used		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: Con	ntainers runni	ing			docker.containers.ru	nning		7d	365d	Dependent item	Enabled	Application: Docker
4		•••	Docker: Get data_usage: Dock	ker: Contain	ers size			docker.containers_si	ze		7d	365d	Dependent item	Enabled	Application: Docker
m		•••	Docker: Get info: Docker: Con	tainers stop	bed			docker.containers.sto	opped		7d	365d	Dependent item	Enabled	Application: Docker
-		•••	Docker: Get info: Docker: Con	tainers total				docker.containers.tot	al		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: CPL	J CFS Period	d enabled			docker.cpu_cfs_perio	od.enabled		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: CPL	J CFS Quota	enabled			docker.cpu_cfs_quot	a.enabled		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: CPL	J Set enable	d			docker.cpu_set.enab	led		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: CPL	J Shares ena	abled			docker.cpu_shares.e	nabled		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: Deb	oug enabled				docker.debug.enable	d		7d	365d	Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: Defa	ault runtime				docker.default_runtin	ne		7d		Dependent item	Enabled	Application: Docker
		•••	Docker: Get info: Docker: Doc	ker root dir				docker.root_dir			7d		Dependent item	Enabled	Application: Docker
9		•••	Docker: Get containers					docker.containers		1m	0		Zabbix agent	Enabled	Application: Zabbix ra
2		•••	Docker: Get data_usage					docker.data_usage		1m	0		Zabbix agent	Enabled	Application: Zabbix ra
2		•••	Docker: Get images					docker.images		1m	0		Zabbix agent	Enabled	Application: Zabbix ra
Ì	Ocker: Get info					docker.info 1m			0		Zabbix agent	Enabled	Application: Zabbix ra		
÷		•••	Docker: Get info: Docker: Gord	outines				docker.goroutines			7d	365d	Dependent item	Enabled	Application: Docker
ڻ ا		•••	Docker: Get images: Docker: I	Images avail	able			docker.images.top_le	evel		7d	365d	Dependent item	Enabled	Application: Docker

Regular docker template items

The template contains a set of regular items for the general Docker instance metrics, such as the number of available images, Docker architecture information, the total number of containers, and more.

Ζ	Discovery rules								Create dis
Q	All templates / Docker by Zabbix ag	gent 2 Items 44 Trigg	ers 3 Graphs 5 Das	shboards 1 Discovery	rules 2 Web scenarios				
0	Host groups	type here to search		Select	Туре	all	✓ Status all	Enabled Disa	abled
Ō	Templates	Docker by Zabbix agent	2 🗙	Select	Update interva	1			
:=	Name	type here to search			Keep lost resources period	i			
	Key								
٩					Apply Reset				
				_					
	Template	Name 🔺	Items	Triggers	Graphs	Hosts	Key	Interval	Туре
	Docker by Zabbix agent 2	Containers discovery	Item prototypes 36	Trigger prototypes 2	Graph prototypes 4	Host prototypes	docker.containers.discovery[false]	15m	Zabbix agent
	Docker by Zabbix agent 2	Images discovery	Item prototypes 2	Trigger prototypes	Graph prototypes	Host prototypes	docker.images.discovery	15m	Zabbix agent
									Displaying 2

Docker template Low-level discovery rules

On top of that, the template also gathers container and image-specific information by using lowlevel discovery rules.

Once Zabbix discovers your containers and images, these low-level discovery rules will then be used to create items, triggers, and graphs from prototypes for each of your containers and images. This way, we can monitor container or image-specific metrics, such as container memory, network information, container status, and more.

Z	Item	prototy	/pes

Ζ	Ite	m pi	rototypes										Create item pr
ৎ	All 1	empla	tes / Docker by Zabbix agent 2 Discovery list	t / Containers discovery	em prototypes 36	Trigger prototypes 2	Graph prototypes	4 Ho	st prototy	rpes			
\odot			Name 🛦		Key		Interval	History	Trends	Туре	Create enabled	Discover	Tags
õ		•••	Container (#NAME): Get stats: Container (#NAMe) per second	ME}: CPU kernelmode usage	docker.container_ {#NAME}"]	_stats.cpu_usage.kerne	I.rate["	7d	365d	Dependent item	Yes	Yes	Application: Dock
=		•••	Container {#NAME}: Get stats: Container {#NAME}second	ME}: CPU total usage per	docker.container_ NAME}"]	_stats.cpu_usage.total.r	ate["{#	7d	365d	Dependent item	Yes	Yes	Application: Dock
• •		•••	Container {#NAME}: Get stats: Container {#NAMeper second	ME}: CPU usermode usage	docker.container_ NAME}"]	_stats.cpu_usage.user.r	ate["{#	7d	365d	Dependent item	Yes	Yes	Application: Dock
		• • •	Container {#NAME}: Get info: Container (#NAM	/E}: Created	docker.container_	_info.created["{#NAME}']	7d	365d	Dependent item	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get info: Container {#NAM	/E}: Dead	docker.container_	_info.state.dead["{#NAM	IE}"]	7d	365d	Dependent item	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get info: Container {#NAM	/E): Error	docker.container_	_info.state.error["{#NAM	E}"]	7d		Dependent item	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get info: Container {#NAM	//E}: Exit code	docker.container_	_info.state.exitcode["{#N	IAME}"]	7d	365d	Dependent item	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get info: Container {#NAM	/E): Finished at	docker.container_	_info.finished["{#NAME}	"]	7d	365d	Dependent item	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get info		docker.container	_info["{#NAME}"]	1m	0		Zabbix agent	Yes	Yes	Application: Dock
		•••	Container {#NAME}: Get stats		docker.container	_stats["{#NAME}"]	1m	0		Zabbix agent	Yes	Yes	Application: Dock
5		•••	Docker: Get containers: Container {#NAME}: In	nage	docker.container	_info.image["{#NAME}"]		7d		Dependent item	Yes	Yes	Application: Dock
?		•••	Container {#NAME}: Get stats: Container {#NA	ME): Memory commit bytes	docker.container_ {#NAME}"]	_stats.memory.commit_I	bytes["	7d	365d	Dependent item	Yes	Yes	Application: Dock
•		•••	Container (#NAME): Get stats: Container (#NAI bytes	ME): Memory commit peak	docker.container_ ytes["{#NAME}"]	_stats.memory.commit_	peak_b	7d	365d	Dependent item	Yes	Yes	Application: Dock
ሳ		•••	Container {#NAME}: Get stats: Container {#NAI usage	ME): Memory maximum	docker.container_ NAME}"]	_stats.memory.max_usa	ge["{#	7d	365d	Dependent item	Yes	Yes	Application: Dock

Docker templates Low-level discovery item prototypes

Verifying the host and template configuration

To verify that the agent and the host are configured correctly, we can use Zabbix get commandline tool and try to poll our agent. If you haven't installed Zabbix get, do so on your Zabbix server or Zabbix proxy host:

dnf install zabbix-get

Now we can use zabbix-get to verify that our agent can obtain the Docker-related metrics. Execute the following command:

zabbix get -s docker-host -k docker.info

Use the -s parameter to specify your agent host's host name or IP address. The -k parameter specifies the item key for which we wish to obtain the metrics by polling the agent with Zabbix get. zabbix_get -s 192.168.50.141 -k docker.info

{"Id": "SJYT: SATE: 7XZE: 7GEC: XFUD: KZO5: NYFI: L7M5: 4RG0: P2KX: QJFD: TAVY", "Containers": 2, "Containers Running": 2, "ContainersPaused": 0, "ContainersStopped": 0, "Images": 2, "Driver": "overlay2", "MemoryLi mit": true, "SwapLimit": true, "KernelMemory": true, "KernelMemoryTCP": true, "CpuCfsPeriod": true, "Cpu CfsQuota": true, "CPUShares": true, "CPUSet": true, "PidsLimit": true, "IPv4Forwarding": true, "BridgeNf Iptables": true, "BridgeNfIP6tables": true, "Debug": false, "NFd": 39, "OomKillDisable": true, "NGorouti nes": 43, "LoggingDriver": "jsonfile", "CgroupDriver": "cgroupfs", "NEventsListener": 0, "KernelVersion": "5. 4. 17-2136. 300. 7. el8uek. x86_64", "OperatingSystem": "Oracle Linux Server 8. 5", "OSVersion": "8. 5", "OSType": "linux", "Architecture": "x86_64", "IndexServerAddress": "https: // index. docker. io/v1/", "NCPU": 1, "MemTotal": 1776848896, "DockerRootDir": "20. 10. 14", "ClusterStore": "', "ClusterAdvertise": "', "DefaultRuntime": "runc", "LiveRestoreEnabled": false, "InitBinary": "dockerinit", "SecurityOptions": ["name=seccomp, profile=default"], "Warnings": null}

In addition, we can also use the low-level discovery key – *docker.containers.discovery[false]* to check the result of the low-level discovery.

```
zabbix_get -s 192.168.50.141 -k docker.containers.discovery[false]
[{"{#ID}":"alad32f5ee680937806bba62alaa37909a8a6663d8d3268db0ledblac66a49e2","{#NAME}":"/apach
e-
server"},{"{#ID}":"120d59f3c8b416aaeeba50378dee7aeleb89cb7ffc6cc75afdfedb9bc8cae12e","{#NAME}"
:"/mysql-server"}]
```

We can see that Zabbix will discover and start monitoring two containers – *apache-server* and *mysql-server*. Any agent low-level discovery rule or item can be checked with Zabbix get.

Docker template in action

Ζ	Iter	ns															Crea
Q	All h	nosts /	/ Docker host	Enabled ZB	Items 122	Triggers 7	Graphs 13	Discovery ru	iles 2	Web scenarios							F
Θ			Name 🛦					Triggers	Key		Interval	History	Trends	Туре	Status	Tags	
Ō		•••	Containers dis /apache-serve	covery: Contair r: CPU kernelm	er /apache-ser ode usage per	ver: Get stats: second	Container		dock el.rat	ker.container_stats.cpu_usage.kern te["/apache-server"]		7d	365d	Dependent item	Enabled	component: cpu container: /apache-ser	
:=		•••	Containers dis /apache-serve	covery: Contair r: CPU percent	er /apache-ser usage	ver: Get stats:	Container		dock ["/apa	er.container_stats.cpu_pct_usage ache-server"]		7d	365d	Dependent item	Enabled	component: cpu container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: CPU total usa	er /apache-ser ge per second	ver: Get stats:	Container		dock I.rate	er.container_stats.cpu_usage.tota e["/apache-server"]		7d	365d	Dependent item	Enabled	component: cpu container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: CPU usermo	er /apache-ser le usage per se	ver: Get stats: econd	Container		dock r.rate	er.container_stats.cpu_usage.use e["/apache-server"]		7d	365d	Dependent item	Enabled	component: cpu container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: Created	er /apache-ser	ver: Get info: 0	Container		dock -serv	ker.container_info.created["/apache /er"]		7d	365d	Dependent item	Enabled	component: system container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: Dead	er /apache-ser	ver: Get info: 0	Container		dock che-s	er.container_info.state.dead["/apa server"]		7d	365d	Dependent item	Enabled	component: system container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: Error	er /apache-ser	ver: Get info: 0	Container	Triggers 1	dock he-se	er.container_info.state.error["/apac erver"]		7d		Dependent item	Enabled	component: system container: /apache-ser	
		•••	Containers dis /apache-serve	covery: Contair r: Exit code	er /apache-ser	ver: Get info: (Container	Triggers 1	dock pach	er.container_info.state.exitcode["/a ne-server"]		7d	365d	Dependent item	Enabled	component: system container: /apache-ser	
		• • •	Containers dis /apache-serve	covery: Contair r: Finished at	er /apache-ser	ver: Get info: (Container		dock -serv	ker.container_info.finished["/apache /er"]		7d	365d	Dependent item	Enabled	component: system container: /apache-ser	
0 17		•••	Containers dis	covery: Contair	er /apache-ser	ver: Get info			dock r"]	er.container_info["/apache-serve	1m	0		Zabbix agent	Enabled	component: raw container: /apache-ser	
?		•••	Containers dis	covery: Contair	er /apache-ser	ver: Get stats			dock r"]	er.container_stats["/apache-serve	1m	0		Zabbix agent	Enabled	component: raw container: /apache-ser	
÷		•••	Containers dis Image	covery: Docker	Get containers	: Container /a	pache-server:		dock serve	er.container_info.image["/apache- er"]		7d		Dependent item	Enabled	component: images container: /apache-ser	

Discovered items on our Docker host

Now that we have configured our agent and host, applied the Docker template, and verified that everything is working, we should be able to see the discovered entities in the frontend.

Ζ	Latest data						
۹	< T						
0	Subfilter affects only filtered	d data					
Ō	TAGS component 122 container 74	image 4					
≡ • •	TAG VALUES component: application 7 co container: /apache-server 3 image: httpd:2.4 2 myse DATA With data 114 Without data 8	ntainers 5 opu 21 <u>health 1 images 5 memory 15 network 17 os</u> 7 /mysql-server 37 gkjatest 2	s5 <u>raw</u> 8 <u>storage</u> 8 <u>sy</u>	istem 34			
	Host	Name 🛦	Last check	Last value	Change	Tags	
	Docker host	Container /apache-server: CPU kernelmode usage per second	43s	0		component: cpu container: /apache-ser	Graph
	Docker host	Container /apache-server: CPU percent usage	43s	0.002789 %	+0.0004557 %	component: cpu container: /apache-ser	Graph
	Docker host	Container /apache-server: CPU total usage per second	43s	0.021ms	+0.00012ms	component: cpu container: /apache-ser	Graph
	Docker host	Container /apache-server: CPU usermode usage per second	43s	0		component: cpu container: /apache-ser	Graph
	Docker host	Container /apache-server: Created	10m 12s	2022-04-14 13:09:33		component: system container: /apache-ser	Graph
	Docker host	Container /apache-server: Dead	12s	False (0)		component: system container: /apache-ser	Graph
Ģ	Docker host	Container /apache-server: Error	10m 12s			component: system container: /apache-ser	History
2	Docker host	Container /apache-server: Exit code	10m 12s	0		component: system container: /apache-ser	Graph
	Docker host	Container /apache-server: Finished at	10m 12s	0001-01-01 00:00:00		component: system container: /apache-ser	Graph
?	Docker host	Container /apache-server: Get info				component: raw container: /apache-ser	
*	Docker host	Container /apache-server: Get stats				component: raw container: /apache-ser	
ი	Docker host	Container /apache-server: Image	10m 39s	httpd:2.4		component: images container: /apache-ser	History

Collected Docker container metrics

In addition, our metrics should have also started coming in. We can check the *Latest data* section and verify that they are indeed getting collected.

Host			
Host IPMI Tags Macros Inventory	Encryption Value mapping		
Host macros Inherited and host macros	1		
Macro	Effective value	Template value	Global value (c
{\$DOCKER.LLD.FILTER.CONTAINER.MAT CHES}	*	T \sim Change \leftarrow Docker by Zabbix agent 2: ".*"	
Filter of discoverable containers			
{\$DOCKER.LLD.FILTER.CONTAINER.NOT _MATCHES}	CHANGE_IF_NEEDED	$T \overset{\bullet}{\sim} \underline{Change} \leftarrow Docker by Zabbix agent 2: "CHANGE_IF_NEEDED"$	
Filter to exclude discovered containers			
{\$DOCKER.LLD.FILTER.IMAGE.MATCHE S}	* .	T \sim Change \leftarrow Docker by Zabbix agent 2: ".*"	
Filter of discoverable images			
{\$DOCKER.LLD.FILTER.IMAGE.NOT_MAT CHES}	CHANGE_IF_NEEDED	$T \stackrel{\scriptstyle \checkmark}{} \underline{Change} \Leftarrow Docker by Zabbix agent 2: "CHANGE_IF_NEEDED"$	
Filter to exclude discovered images			

Macros inherited from the Docker template

Lastly, we have a few additional options for further modifying the template and the results of our low-level discovery. If you open the *Macros* section of your host and select *Inherited and host macros*, you will notice that there are 4 macros inherited from the Docker template. These macros are responsible for filtering in/out the discovered containers and images. Feel free to modify these values if you wish to filter in/out the discovery of these entities as per your requirements.

Notice that the container discovery item also has one parameter, which is defined as *false* on the template:

- docker.containers.discovery[false] Discover only running containers
- *docker.containers.discovery[true]* Discover all containers, no matter their state.

Revision #4 Created 13 June 2022 16:43:30 by Dino Edwards Updated 6 May 2024 20:23:14 by Dino Edwards