## **Nội dung kịch bản**

Monitor thông tin:

* Alarm 1: “host-offline”: host không respone metric trong 120 s, recheck 60s 1 lần
* Alarm 2: “host-high-resoucre-usage”: tài nguyên (cpu hoặc ram) >90% trong vòng 5 phút

Auto healing:

* Evacuate các instance từ node có alarm 1
* Live migrate các instance từ node có alarm 2

## **Thực hiện:**

### **1. Cấu hình monitor:**

Tại kapacitor, ta định nghĩa alarm host down như sau:

Tạo file tick\_script\_host\_down.txt:

|  |
| --- |
| var db = 'telegraf'  var rp = 'autogen'  var measurement = 'cpu'  var groupBy = ['host']  var whereFilter = lambda: ("cpu" == 'cpu-total') AND ("host" == 'controller' OR "host" == 'node2' OR "host" == 'node3')  var period = 5m  var name = 'host offline'  var idVar = name + '-{{.Group}}'  var message = 'host offline'  var levelTag = 'level'  var messageField = 'message'  var durationField = 'duration'  var triggerType = 'deadman'  var threshold = 0.0  var data = stream     |from()         .database(db)         .retentionPolicy(rp)         .measurement(measurement)         .groupBy(groupBy)         .where(whereFilter)  var trigger = data     |deadman(threshold, period)         .message(message)         .id(idVar)         .levelTag(levelTag)         .messageField(messageField)         .durationField(durationField)         .exec('/etc/kapacitor/kapacitor\_vitrage.py', 'rabbit:// rabbit\_user: rabbit\_pass @controller') |

Tạo file tick\_script\_host\_resource\_problem.txt:

|  |
| --- |
| var db = 'telegraf'  var rp = 'autogen'  var measurement = 'cpu'  var groupBy = []  var whereFilter = lambda: ("cpu" == 'cpu-total')  var period = 5m  var every = 30s  var name = 'host high resource usage'  var idVar = name  var message = 'host high resource usage'  var idTag = 'alertID'  var levelTag = 'level'  var messageField = 'message'  var durationField = 'duration'  var outputMeasurement = 'alerts'  var triggerType = 'threshold'  var crit = 10  var data = stream  |from()  .database(db)  .retentionPolicy(rp)  .measurement(measurement)  .groupBy(groupBy)  .where(whereFilter)  |window()  .period(period)  .every(every)  .align()  |mean('usage\_idle')  .as('value')  var trigger = data  |alert()  .crit(lambda: "value" < crit)  .stateChangesOnly()  .message(message)  .id(idVar)  .idTag(idTag)  .levelTag(levelTag)  .messageField(messageField)  .durationField(durationField)  .exec('/usr/bin/python', '/etc/kapacitor/kapacitor\_vitrage.py’, 'rabbit://rabbit\_user:rabbit\_pass@controller') |

Ở các script trên, quan trọng là ta định nghĩa đưa các alarm kapacitor về với message queue của vitrage: ta đã thêm vào phần định nghĩa dòng:

|  |
| --- |
| .exec('/usr/bin/python', '/etc/kapacitor/**kapacitor\_vitrage.py**','rabbit://rabbit\_user:rabbit\_pass@controller') |

-- thay rabbit\_user , rabbit\_pass và controller bằng transport url của hệ thống

Vitrage sẽ đọc được alarm từ messeage queue và cho vào graph

File **kapacitor\_vitrage.py** download tại <https://github.com/openstack/vitrage/blob/master/vitrage/datasources/kapacitor/auxiliary/kapacitor_vitrage.py>

Chạy lệnh:

|  |
| --- |
| kapacitor define check\_host\_down -tick tick\_script\_host\_down.txt -dbrp 'telegraf'.'autogen'  kapacitor define check\_host\_resource\_problem -tick tick\_script\_host\_resource\_problem.txt -dbrp 'telegraf'.'autogen' |

### **2. Cấu hình vitrage.**

**a) Cấu hình daemon:**

Sửa file **/etc/vitrage/vitrage.conf:**

|  |
| --- |
| [DEFAULT]  notifiers = nova,mistral  …  [datasources]  types = **kapacitor**,zabbix,nova.host,nova.instance,nova.zone,static\_physical,aodh,cinder.volume,neutron.network,neutron.port,heat.stack  …  [kapacitor]  config\_file = /etc/vitrage/kapacitor\_conf.yaml |

Tạo file /etc/vitrage/kapacitor\_conf.yaml , mục đích ánh xạ alarm vitrage nhận được từ kapacitor ở trên đến các đối tượng của openstack :

|  |
| --- |
| kapacitor:  - alert:  host: compute-(.\*)  vitrage\_resource:  type: nova.host  name: ${kapacitor\_host}  - alert:  host: (.\*)  vitrage\_resource:  type: nova.instance  name: ${kapacitor\_host} |

Theo như cấu hình trên:

* Nếu có alarm trên máy có hostname “compute-01” sẽ map với nova.host “compute01” ,

tương tự hostname “compute-02” sẽ map với nova.host “compute-02” …

* Các alarm trên máy có hostname khác với compute-\* sẽ được coi là instance

**b) Thêm template vitrage**

cấu hình vitrage khi có alarm với title “host offline” đến sẽ gọi ra workflow mistral evacuate host:

tạo file: auto\_evacuate\_scenario.yaml

|  |
| --- |
| metadata:  version: 2  name: evacuate instane when host problem  type: standard  description: evacuate instane when host problem  definitions:  entities:  - entity:  category: ALARM  name: host offline  type: kapacitor  template\_id: host\_alarm  - entity:  category: RESOURCE  type: nova.host  template\_id: host  - entity:  category: RESOURCE  type: nova.instance  template\_id: instance  relationships:  - relationship:  source: host\_alarm  target: host  relationship\_type: on  template\_id : alarm\_on\_host  - relationship:  source: host  target: instance  relationship\_type: contains  template\_id : host\_contains\_instance  scenarios:  - scenario:  condition: alarm\_on\_host and host\_contains\_instance  actions:  - action:  action\_type : raise\_alarm  properties:  alarm\_name: instance\_offline  severity: critical  action\_target:  target: instance  - action:  action\_type : execute\_mistral  properties:  workflow: instance\_evacuate  instance\_id: get\_attr(instance, id)  retries: 5 |

Cấu hình vitrage khi có alarm với title “host high resource usage” đến sẽ gọi ra workflow mistral live migrate host:

Tạo file auto\_live\_migrate\_scenario.yaml

|  |
| --- |
| metadata:  version: 2  name: live migrage instane when host problem  type: standard  description: evacuate instane when host problem  definitions:  entities:  - entity:  category: ALARM  name: host high resource usage  type: kapacitor  template\_id: host\_alarm  - entity:  category: RESOURCE  type: nova.host  template\_id: host  - entity:  category: RESOURCE  type: nova.instance  template\_id: instance  relationships:  - relationship:  source: host\_alarm  target: host  relationship\_type: on  template\_id : alarm\_on\_host  - relationship:  source: host  target: instance  relationship\_type: contains  template\_id : host\_contains\_instance  scenarios:  - scenario:  condition: alarm\_on\_host and host\_contains\_instance  actions:  - action:  action\_type : raise\_alarm  properties:  alarm\_name: instance\_resource problem  severity: critical  action\_target:  target: instance  - action:  action\_type : execute\_mistral  properties:  workflow: instance\_live\_migrate  instance\_id: get\_attr(instance, id)  retries: 5 |

Chạy lệnh:

|  |
| --- |
| $ vitrage tempalte validate --type standard --path auto\_evacuate\_scenario.yaml  $ vitrage add validate --type standard --path evacuate\_ auto\_evacuate\_scenario.yaml |

### **3. Cấu hình workflow mistral:**

Định nghĩa **evacuate\_instance** workflow:

Tạo file evacuate\_instance.yaml

|  |
| --- |
| ---  version: '2.0'  instance\_evacuate:  type: direct  input:  - instance\_id  tasks:  get\_instance\_status\_before:  action: nova.servers\_get server=<% $.instance\_id %>  publish:  instance\_name: <% task(get\_instance\_status\_before).result.name %>  status\_before: <% task(get\_instance\_status\_before).result %>  host\_before: <% task(get\_instance\_status\_before).result["OS-EXT-SRV-ATTR:host"] %>  on-success: evacuate\_instance  on-error: send\_error\_email    evacuate\_instance:  action: nova.servers\_evacuate server=<% $.instance\_id %>  retry:  count: 10  delay: 30  on-success: wait\_for\_instance\_rebuild  on-error: send\_error\_email    wait\_for\_instance\_rebuild:  action: nova.servers\_get server=<% $.instance\_id %>  retry:  count: 30  delay: 2  continue-on: <% task(wait\_for\_instance\_rebuild).result.status !="REBUILD" %>  on-success: wait\_instance\_status\_active  wait\_instance\_status\_active:  action: nova.servers\_get server=<% $.instance\_id %>  retry:  delay: 10  count: 30  continue-on: <% task(wait\_instance\_status\_active).result.status !="ACTIVE" %>  on-success: get\_instance\_status\_after    get\_instance\_status\_after:  action: nova.servers\_get server=<% $.instance\_id %>  publish:  status\_after: <% task(get\_instance\_status\_after).result.status %>  host\_after: <% task(get\_instance\_status\_after).result["OS-EXT-SRV-ATTR:host"] %>  on-complete: check\_diffrent\_host  check\_diffrent\_host:  action: std.noop  on-complete:  - send\_success\_email: <% $.status\_before = $.status\_after and $.host\_before != $.host\_after %>  - send\_error\_email: <% $.status\_before != $.status\_after or $.host\_before = $.host\_after %>    send\_error\_email:  action: std.email  input:  to\_addrs: 'admin@demo.com'  subject: ERROR evacuate vm  body: |  We try to evacuate vm <% $.instance\_id> when host have problem  Please look at mistral workflow <% execution().id %> for more detail  from\_addr: admin@demo.com  smtp\_server: smtp.google.com  smtp\_password: SECRET    send\_success\_email:  action: std.email  input:  to\_addrs: [admin@demo.com]  subject: SUCCESS evacuate vm  body: |  We evacuate vm <% $.instance\_id> when host have prolem.  Please look at mistral workflow <% execution().id %> for more detail  from\_addr: admin@demo.com  smtp\_server: smtp.google.com  smtp\_password: SECRET |

Định nghĩa live\_migrate workflow:

Tạo file live\_migrate\_instance.yaml

|  |
| --- |
| ---  version: '2.0'  instance\_live\_migrate:  type: direct  input:  - instance\_id  - target\_host:  - block\_migration: False  - disk\_over\_commit: False  tasks:  get\_instance\_status\_before:  action: nova.servers\_get server=<% $.instance\_id %>  publish:  instance\_name: <% task(get\_instance\_status\_before).result.name %>  status\_before: <% task(get\_instance\_status\_before).result.status %>  host\_before: <% task(get\_instance\_status\_before).result["OS-EXT-SRV-ATTR:host"] %>  on-success: live\_migrate\_instance  on-error: send\_error\_email    live\_migrate\_instance:  action: nova.servers\_live\_migrate server=<% $.instance\_id %> host=<% $.target\_host %> block\_migration=<% $.block\_migration %> disk\_over\_commit=<% $.disk\_over\_commit %>  retry:  count: 10  delay: 30  on-success: wait\_for\_instance\_migrate  on-error: send\_error\_email    wait\_for\_instance\_migrate:  action: nova.servers\_get server=<% $.instance\_id %>  retry:  count: 30  delay: 2  continue-on: <% task(wait\_for\_instance\_migrate).result.status !="MIGRATING" %>  on-success: wait\_for\_instance\_active  wait\_for\_instance\_active:  action: nova.servers\_get server=<% $.instance\_id %>  retry:  count: 30  delay: 2  continue-on: <% task(wait\_for\_instance\_active).result.status !="ACTIVE" %>  on-success: get\_instance\_status\_after    get\_instance\_status\_after:  action: nova.servers\_get server=<% $.instance\_id %>  publish:  status\_after: <% task(get\_instance\_status\_after).result.status %>  host\_after: <% task(get\_instance\_status\_after).result["OS-EXT-SRV-ATTR:host"] %>  on-complete: check\_diffrent\_host  check\_diffrent\_host:  action: std.noop  on-complete:  - send\_success\_email: <% $.status\_before = $.status\_after and $.host\_before != $.host\_after %>  - send\_error\_email: <% $.status\_before != $.status\_after or $.host\_before = $.host\_after %>    send\_error\_email:  action: std.email  input:  to\_addrs: 'admin@demo.com'  subject: ERROR evacuate vm  body: |  We try to migrate vm <% $.instance\_id> when host have problem  Please look at mistral workflow <% execution().id %> for more detail  from\_addr: admin@demo.com  smtp\_server: smtp.google.com  smtp\_password: SECRET    send\_success\_email:  action: std.email  input:  to\_addrs: [admin@demo.com]  subject: SUCCESS evacuate vm  body: |  We migrate vm <% $.instance\_id> when host have prolem.  Please look at mistral workflow <% execution().id %> for more detail  from\_addr: admin@demo.com  smtp\_server: smtp.google.com  smtp\_password: SECRET |

Thay các trường [from\_email, smtp\_server, smtp\_password, to\_email] cho phù hợp.

Chạy lệnh:

|  |
| --- |
| mistral workflow-validate evacuate\_instance.yaml  mistral workflow-create evacuate\_instance.yaml  mistral workflow-validate live\_migrate\_instance.yaml  mistral workflow-create live\_migrate\_instance.yaml |

Evacuate và live migrate error trong các trường hợp:

* Trạng thái instance sau và trước khác nhau. Vd: instance trước ở trạng thái shutoff thì sau phải ở shutofff; trước active thì sau ở trạng thái active
* Host của instance không thay đổi

Đề xuất nên trả error trong trường hợp không vượt qua một “verify\_instance workflow” ( vd kiểm tra ip có ping, curl từ host cài mistral đến instance - hoặc có thể lấy thông tin một host monitor trung gian nào khác). Workflow này tùy mỗi instance-id mà có các task riêng.