Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital

SmartMetropolis – Plataforma e Aplicações para Cidades Inteligentes

WP4 – Infraestrutura

Ferramentas de Monitoramento e Análise de Infraestrutura da Plataforma Fiware

Natal-RN, Brasil
Abril 2017

Equipe Técnica

Docentes

Prof. Dr. Carlos Eduardo da Silva (Coordenador) – IMD/UFRN

Discentes
Gabriela Cavalcante da Silva
Marcelo Avelino de Medeiros

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O projeto Smart Metropolis, conduzido pelo Instituto Metrópole Digital (IMD) da Universidade Federal do Rio Grande do Norte (UFRN), tem como objetivo o desenvolvimento de soluções de tecnologia da informação e comunicação para Cidades Inteligentes e Humanas.

O projeto é organizado em seis Pacotes de Trabalho (Work Packages - WP) temáticos, liderados cada por um coordenador. Cada WP possui um conjunto de objetivos a serem alcançados ao longo dos cinco anos de execução do projeto.

Este relatório está inserido no contexto do WP 4 - Infraestrutura Computacional, que no contexto do segundo ano de execução do projeto Smart Metropolis, tem os seguintes objetivos principais:

- Operação de Infraestrutura: Este objetivo engloba a operação e manutenção da instância Fiware do projeto, incluindo ferramentas para monitoramento da infraestrutura, e para gerenciamento de aplicações.
- Segurança da Informação e Controle de Acesso: Este objetivo engloba o estudo aprofundado dos mecanismos de segurança da plataforma Fiware com o fim de oferecer uma solução de controle de acesso que possa ser utilizada pelas aplicações que irão executar sobre a infraestrutura Fiware.

Neste contexto, este relatório corresponde ao entregável definido pela **Meta 1.6: Estudo das ferramentas de monitoramento e análise disponibilizadas pela plataforma FIWARE (OPS-Health)**, que envolve o estudo em ferramentas de monitoramento e análise da infraestrutura computacional de uma instância Fiware.

Neste sentido, apresentamos uma descrição das diversas ferramentas disponibilizadas pelo OPS-Health (Seção 2), seguido de um breve relato sobre a experiência de utilização desta ferramenta (Seção 3).

2 OPS-Health

A plataforma Fiware, em seu capítulo técnico OPS Tools[1] [1] oferece um conjunto de ferramentas para operação, monitoramento e manutenção dos nós que compõem a infraestrutura computacional de instâncias Fiware. Dentre essas ferramentas, um sub-conjunto delas, denominado de OPS-Health, permite que administradores e usuários finais obtenham informações acerca da "saúde" dos nós que compõem a infraestrutura de maneira simplificada, além de emitirem alertas quando do mau-funcionamento de algum componente. Ele é composto por três ferramentas: o Sanity Check (Sanity Check Engine e Sanity Check Dashboard), Infographics & Status Page e Federation Monitoring, as quais serão brevemente descritas nesta seção. Para a descrição destas ferramentas foi tomado como referência o relatório técnico sobre o capítulo FIWARE OPS Tools [1].

2.1 Sanity Check

Esta seção descreve o que é o Sanity Check, sua importância, seus componentes e respectivas arquiteturas, bem como o seu funcionamento.

O Sanity Check é uma ferramenta de código aberto para o OpenStack verificar o estado de saúde de cada nó FIWARE Lab. Como mencionado anteriormente, ele dispõe de dois componentes: o Sanity Check Engine, o qual atua na verificação dos testes e o Sanity Check Dashboard, cujo papel principal é

[1] `http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_OPS_Tools`

exibir os resultados das execuções de monitoramento feitas pelo Sanity Check Engine. O Sanity Check executa uma coleção abrangente de casos de testes de sanidade sobre cada nó federado no FIWARE Lab para validar suas capacidades.

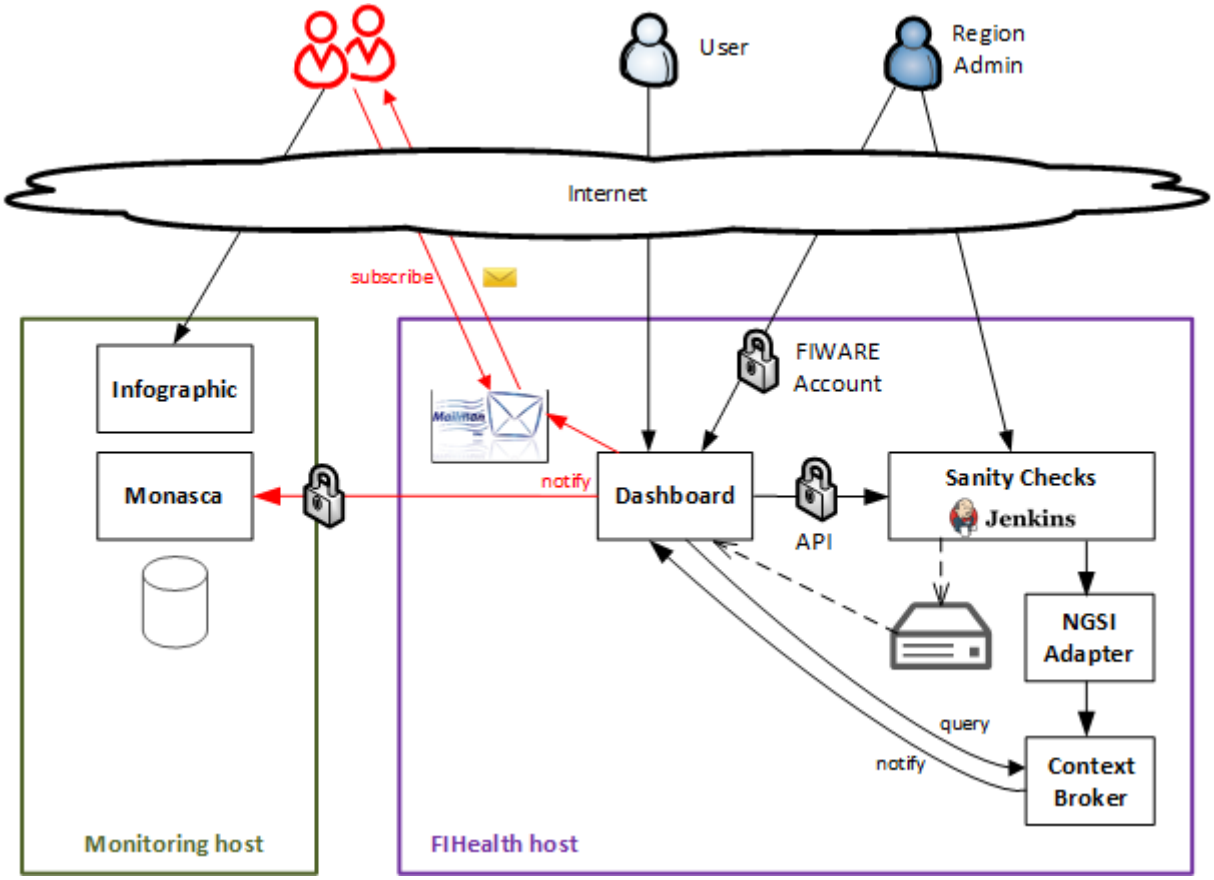

Figura 1: Arquitetura - Sanity Check

A arquitetura do Sanity Check é demonstrada na Figura 1. Podemos analisar essa arquitetura primeiro entendendo o que é o Sanity Check Engine (Sanity Check), o qual trata de fazer a verificação da saúde das regiões (nós). Para fazer isso, o framework Sanity Checks executa alguns testes em cada nó, testando seus principais recursos usando as ações que um provável usuário das regiões utilizaria para validar aquela região, o qual garante uma antecipação a problemas que os usuários poderiam enfrentar, bem como o administrador terá menos trabalho manual para encontrar as falhas.

O administrador da rede irá executar as verificações do Sanity Checks no Jenkins[2] (o qual é a ferramenta recomendada para executar estes testes, pois os de forma fará automatizada). Ter uma ferramenta como o Sanity Check Engine possibilita coletar dados com base numa coleção de casos de teste que abrange as seguintes características principais: operações de rede, operações de gerenciamento de imagens e operações de armazenamento de objetos.

Os resultados desses testes são então publicados em um Contex Broker utilizando um adaptador NGSI que transforma esses resultados em atributos de contexto NGSI e, enfim, encaminha esses dados através do Context Broker para atualizar o contexto da região. Neste processo é calculado o estado de sanidade das regiões de acordo com os testes individuais.

O Sanity Check Dashboard irá consultar esses dados e exibí-los de modo que tanto o administrador do

[2] https://jenkins.io/

nó (Region Admin) quanto os usuários poderão monitorar a saúde de cada um dos nós, permitindo que eles saibam quais nós estão funcionando. O Dashboard, portanto, mostra todos os resultados dos testes realizados no Sanity Checks Engine, permitindo que se veja logs detalhados (o que é de suma importância em caso de falhas encontradas nos testes). Além disso, o Region Admin também pode reiniciar os testes no Sanity Checks utilizando uma credencial de uma conta FIWARE e também se inscrever para receber notificações das mudanças nos status das regiões via e-mail (como mostrado nas setas vermelhas da figura 1). Essas mudanças também são publicadas como métricas na Monasca. Estas informações do estado de saúde de cada nó de uma rede sendo mostrada graficamente e em tempo real são de real importância visto que em caso de erro em qualquer nó, o tempo de resposta por parte do administrador da rede para se recuperar da falha é muito mais rápido do que a procura manual do erro.

2.2 Infographics & Status Page

Esta ferramenta está mais voltada para o marketing da infrestrutura, uma vez que ela consiste em permitir aos usuários verificar de forma bastante didática a quantidade de regiões, o status dos nós, etc. São exibidas informações a partir dos outros componentes OPS-Health: Federation Monitoring e Sanity Check. Além disso, o usuário também pode visualizar o histórico do estado dos nós.

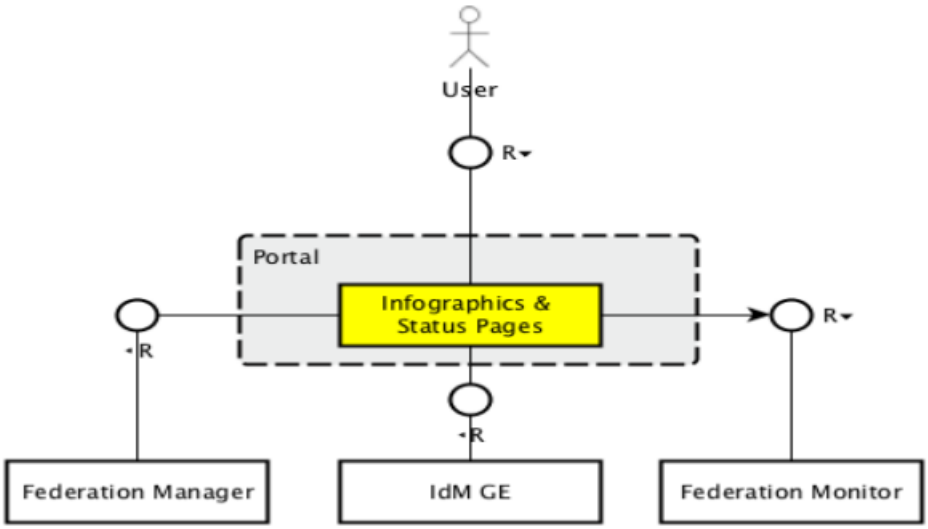

Figura 2: Arquitetura - Infographics & Status Page.

A Figura 2 descreve a arquitetura do Infographics & Status Page. Como mencionado anteriormente neste subseção, O componente Infographics & Status Page fornece as informações de capacidade a respeito das regiões OpenStack, bem como informações de status dos serviços de infra-estrutura (nós que estão funcionando, etc.). O componente Federation Monitor é responsável por mostrar os dados de capacidade das regiões e o status da infra-estrutura FIWARE Lab através de um conjunto de API RESTful que o Infographics & Status Page chama para obter esses dados. O componente IdM GE fornece os mecanismos de autenticação para suporte ao envio de mensagens pelo Federation Manager e do administrador da infraestrutura. O componente Federation Manager também é responsáevl por fornecer as informações dos status dos serviços da infraestrutura.

2.3 Federation Monitoring

O componente Federation Monitoring tem a função de fornecer dados de monitoramento dos nós FIWARE Lab em relação ao gerenciamento de falhas e de desempenho. Ele é baseado nas ferramentas Ceilometer e Monasca e destina-se a contribuir com a comunidade OpenStack em relação à federação de serviços de monitoramento, como descrito na wiki do OPS-Health.

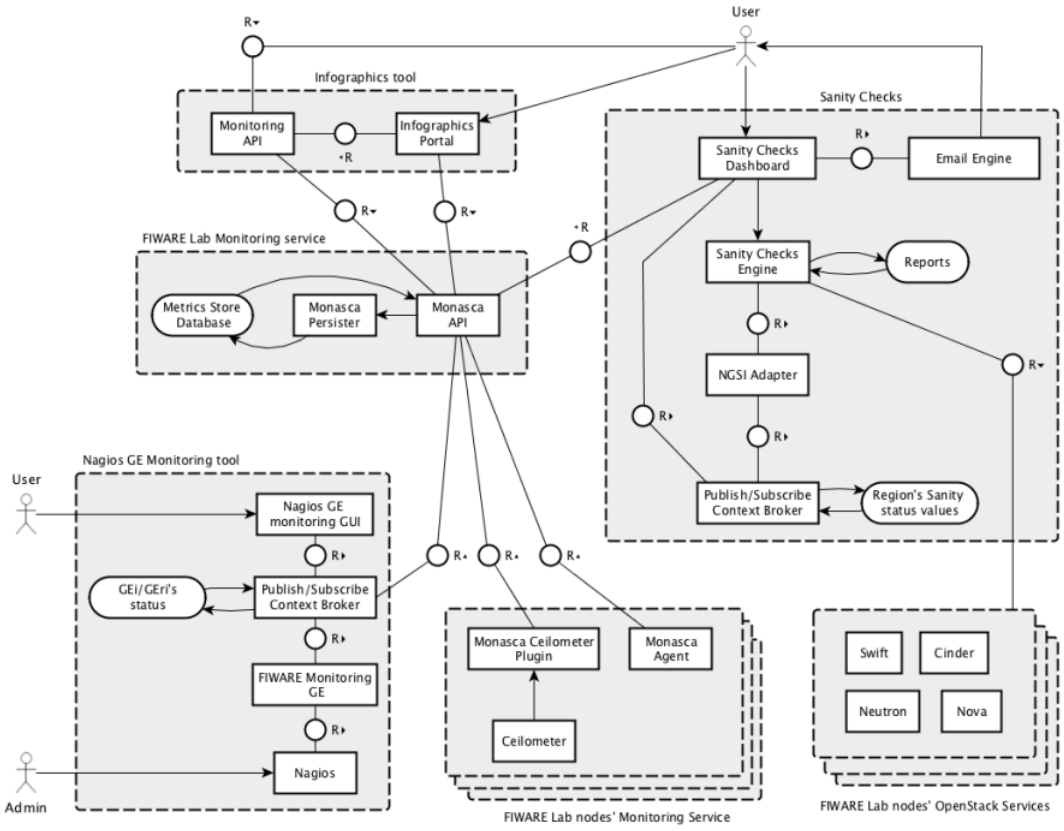

Figura 3: Visão Geral - Federation Monitoring

A Figura 3 descreve uma visão geral do que é o componente Federation Monitoring e de sua arquitetura. Esse componente é um sistema mais complexo e que envolve a colaboração de vários outros componentes.

São eles: o infographics tool, Sanity Cheks, Nagios GE Monitoring tool, OpenStack Celiomiter e Monasca Agent (FIWARE Lab nodes Monitoring Service) e Federation Monitoring API. O infographics tool trata de mostrar em um dashboard um resumo das informações de monitoramento, as quais foram coletadas usando diferentes componentes de monitoramento. O Sanity Checks permite saber o estado de cada um dos nós FIWARE Lab a partir dos testes que ele realiza, bem como ter um histórico das informações obtidas nesses testes. O componente Nagios GE Monitoring tool permite obter o estado de execução dos testes usando plugins Nagios. Ele é responsável pelo controle das instâncias mais importantes Generic Enablers (GEis) e Generice Enablers reference istances (GEris) que estão no FIWARE Lab (Orion, Mashup, COSMOS, etc.).

De acordo com a Figura 3 é possível ver que uma das características é que todos estes componentes se conectam a um componente central que está dentro do FIWARE LAb monitoring service e é baseado na API Monasca, a fim de centralizar as informações. No OpenStack Ceilometer e Monasca agente cada nó FIWARE Lab tem que executar uma instância do ceilometer com um conjunto de informações sobre

o estado, a performance e a utilização daquele nó. Com um Monasca Agent instalado é possível obter informações sobre os hosts físicos no nó. Dessa forma, permite saber os diferentes serviços que estão sendo executados em cada um dos nós FIWARE Lab.

3 Experimentação

3.1 Sanity Check

Para a instalação do Sanity Check para os testes iniciais foram utilizado os passos abaixo, utilizando como base os arquivos do repositório do OPS-Health[3]: Estes passos foram executados em uma máquina virtual, onde foi instalado o OpenStack Keystone.

Na sequência, apresentamos os passos seguidos para instalação do Sanity Check em um ambiente virtualizado.

Uma vez que a máquina virtual se encontra em funcionamento, iniciou-se com a configuração de um servidor de banco de dados MySQL, onde foram executados os seguintes comandos:

1. Abra o MySQL cliente e crie o banco de dados "keystone".

```
create database keystone;
```

2. O passo seguinte foi liberar o acesso ao banco de dados keystone. Troque com sua senha do banco o `KEYSTONE_DBPASS`:

```
grant all privileges on keystone.* to
'keystone'@'localhost' identified by '*KEYSTONE_DBPASS*';
grant all privileges on keystone.* to 'keystone'@'%'
identified by '*KEYSTONE_DBPASS*';
quit
```

3. Feito isso, deve-se instalar os pacotes keystone:

```
# apt-get install keystone apache2 libapache2-mod-wsgi
```

4. Após, deve ativar e iniciar o memcached service:

```
# systemctl enable memcached.service
# systemctl start memcached.service
```

5. Configure o keystone (Substitua `ADMIN_TOKEN` e `KEYSTONE_DBPASS` com os seus valores):

```
# vim /etc/keystone/keystone.conf

  [DEFAULT]
  admin_token = *ADMIN_TOKEN*

  [database]
  connection = mysql://keystone:*KEYSTONE_DBPASS*@controller/keystone

  [memcache]
```

[3] `https://github.com/Fiware/ops.Health.git`

```
servers = localhost:11211

[token]
provider = uuid
driver = memcache
expiration = 86400

[revoke]
driver = sql
```

5. Popule a base de dados do keystone:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

6. É necessário também configurar o nome do host no arquivo de configuração do servidor Web Apache,:

```
# vim /etc/httpd/conf/httpd.conf
ServerName controller
```

7. Configure wsgi:

```
# vim /etc/httpd/conf.d/wsgi-keystone.conf

Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone gr
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    ErrorLog /var/log/httpd/keystone-error.log
    CustomLog /var/log/httpd/keystone-access.log combined

    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
```

```
  </VirtualHost>

  <VirtualHost *:35357>
      WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone gro
      WSGIProcessGroup keystone-admin
      WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
      WSGIApplicationGroup %{GLOBAL}
      WSGIPassAuthorization On
      <IfVersion >= 2.4>
        ErrorLogFormat "%{cu}t %M"
      </IfVersion>
      ErrorLog /var/log/httpd/keystone-error.log
      CustomLog /var/log/httpd/keystone-access.log combined

      <Directory /usr/bin>
          <IfVersion >= 2.4>
              Require all granted
          </IfVersion>
          <IfVersion < 2.4>
              Order allow,deny
              Allow from all
          </IfVersion>
      </Directory>
  </VirtualHost>
```

8. Habilitar e iniciar o servidor Apache:

```
# systemctl enable httpd.service
# systemctl start httpd.service
```

9. Configurar os parâmetros de conexão (substitua `ADMIN_TOKEN` pelo seu próprio):

```
# export OS_TOKEN=*ADMIN_TOKEN*
# export OS_URL=http://controller:35357/v3
# export OS_IDENTITY_API_VERSION=3
```

10. Crie o domínio no Openstack, o projeto, o usuário e a função

```
 $ openstack --os-interface public project create --domain default test
 $ openstack --os-interface public user create --domain default test --passwor
 $ openstack --os-interface public role add --user test --project test owner
 $ openstack --os-interface public project show test > project
 $ openstack --os-interface public user show test > user
```

11. Rode o Sanity Check na command line. Edite o arquivo etc/settings.json. Por exemplo:

```
{
    "environment": "fiware-lab",
    "credentials": {
```

```
        "keystone_url": "http://cloud.lab.fiware.org:4731/v3/",
        "user_id": "",
        "tenant_id": "test",
        "tenant_name": "test",
        "user_domain_name": "default",
        "project_domain_name": "default",
        "username": "test",
        "password": "test"
    },
    "test_configuration": {
        "phonehome_endpoint": "",
        "glance_configuration": {
            "required_images": [
                "base_centos_6",
                "base_centos_7",
                "base_debian_7",
                "base_ubuntu_12.04",
                "base_ubuntu_14.04"
            ]
        },
        "swift_configuration": {
            "big_file_url_1": "https://github.com/telefonicaid/fiware-paas/arc
            "big_file_url_2": "https://github.com/telefonicaid/fiware-paas/arc
        },
        "openstack_metadata_service_url": "http://169.254.169.254/openstack/la
    },
    "key_test_cases": [
        "test_(.*)"
    ],
    "opt_test_cases": [
        "test_.*container.*"
    ],
    "region_configuration": {
        "RegionOne": {
            "external_network_name": "ext-net",
            "shared_network_name": "shared-net",
            "test_object_storage": false
        }
    }
}
```

12. Rodando `./sanity_check`. Esse comando vai executar o Sanity Checks.

Como nos testes só havia uma região, foi aberta uma issue[4] sobre os argumentos na tag `region_`

[4] `http://stackoverflow.com/questions/42446860/fiware-health-region-configuration/`

`configuration`.

Por fim, os resultados dos testes são gerados no arquivo `test_results.xml`.

4 Considerações Finais

Este relatório apresentou uma descrição das ferramentas que compõem o OPS-Health, disponibilizadas pela plataforma Fiware para atividades de monitoramento de uma nuvem OpenStack. A implantação das ferramentas do OPS-Health requerem uma instalação OpenStack funcional, o que exigiu a criação de ambientes simulados, que nem sempre capturam todos os aspectos de uma infraestrutura envolvendo diversos servidores físicos.

Durante a experimentação com as ferramentas do OPS-Health encontramos diversos problemas relacionados à instalação e configuração das mesmas. Durante nossos estudos, foi descoberto um bug em uma das ferramentas, que já foi reportado para a equipe de desenvolvimento do Fiware. Dessa forma, indicamos como próximos passos finalizar a implantação do Sanity Check para a realização de testes com um ambiente virtualizado, com vistas a elaboração de um plano de testes na infraestrutura computacional da nuvem do projeto.

Referências

[1] Silvio Cretti, *D.21.1.2: Platform deployment, operations, analytics and support tools*, Technical report Number: ICT-2013-FI-632893-WP21-D.21.1.2, Disponível em `https://forge.fiware.org/docman/view.php/7/5652/D.21.1.2+Platform+deployment%2C+operations%2C+analytics+and+support+tools.pdf`, 2016.

`42481202#42481202`