UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ADRIANO RIBEIRO DE ARAÚJO

DESENVOLVIMENTO DE UM SERVIÇO DE APOIO A GERÊNCIA DE RECURSOS DA NUVEM DO LSD

CAMPINA GRANDE - PB

2021

ADRIANO RIBEIRO DE ARAÚJO

DESENVOLVIMENTO DE UM SERVIÇO DE APOIO A GERÊNCIA DE RECURSOS DA NUVEM DO LSD

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Dr. Thiago Emmanuel Pereira da Cunha Silva.

CAMPINA GRANDE - PB

2021

A663d Araújo, Adriano Ribeiro de.
Desenvolvimento de um serviço de apoio à gerência de recursos da nuvem do LSD. / Adriano Ribeiro de Araújo. – 2021.

9 f.

Orientador: Prof. Dr. Thiago Emmanuel Pereira da Cunha Silva.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Nuvem do Laboratório de Sistemas Distribuídos - UFCG. 2. Serviço de nuvem. 3. Laboratório de Sistemas Distribuídos – UFCG - Nuvem. 4. Gerência de recursos. 5. Computação em nuvem. I. Silva, Thiago Emmanuel Pereira da Cunha. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

ADRIANO RIBEIRO DE ARAÚJO

DESENVOLVIMENTO DE UM SERVIÇO DE APOIO A GERÊNCIA DE RECURSOS DA NUVEM DO LSD

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Thiago Emmanuel Pereira da Cunha Silva

Orientador – UASC/CEEI/UFCG

Professor Maxwell Guimarães de Oliveira

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 20 de Outubro de 2021.

CAMPINA GRANDE - PB

ABSTRACT

The private cloud service maintained by Laboratório de Sistemas Distribuídos from UFCG's makes computing resources available to the community of employees and students at UASC/UFCG. As there is no charge for users, sometimes we have a misuse of reserved resources, and it is necessary to maintain a larger infrastructure than necessary. Thus, this work presents a monitoring service of available resources, generating periodic reports for administrators and users with the billing of resources used, as well as indications of underutilized resources. From the user's point of view, the objective is to make them aware of whether they are using resources efficiently or not, while in operation, to facilitate the mapping of how the resources are being allocated in an aggregate and individual way among the different projects in the cloud.

Desenvolvimento de um serviço de apoio a gerência de recursos da nuvem do LSD

Adriano Ribeiro de Araújo
adriano.araujo@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

Thiago Emmanuel Pereira da Cunha Silva
temmanuel@computacao.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

RESUMO

O serviço de nuvem privada mantido pelo Laboratório de Sistemas Distribuídos da UFCG disponibiliza recursos computacionais para a comunidade de funcionários e alunos da UASC/UFCG. Como não existe cobrança para os usuários, por vezes temos mau uso dos recursos reservados, sendo preciso manter uma infraestrutura maior que o necessário. Dessa forma, este trabalho apresenta um serviço de monitoramento dos recursos disponibilizados, gerando relatórios periódicos para os administradores e usuários com a bilhetagem dos recursos utilizados, bem como indicações de recursos subutilizados. No ponto de vista do usuário, o objetivo é torná-lo ciente se está usando os recursos de forma eficiente ou não, enquanto na operação, facilitar o mapeamento de como está sendo feita a alocação dos recursos de forma agregada e individual dentre os diferentes projetos na nuvem.

1 INTRODUÇÃO

Usuários de nuvens públicas tem um forte incentivo para utilizar os recursos de maneira eficiente: orçamentos limitados. Por sua vez, quando partimos para nuvens privadas, onde em muitos casos não há cobrança para os usuários, temos mau uso dos recursos reservados, por vezes sendo necessário manter uma infraestrutura maior que o necessário.

Nesse contexto, o Laboratório de Sistemas Distribuídos (LSD) desenvolveu o shylock, uma ferramenta que gera relatórios de bilhetagem incluindo informações dos recursos alocados, como utilização de memória, processamento e volumes. Por email, tais relatórios devem ser enviados aos respectivos responsáveis por aquele projeto. No âmbito de operação, é possível ter um agregado não somente dos recursos alocados, mas também de quais recursos foram reservados para o projeto e quais não estão sendo utilizados. No entanto, o shylock é um serviço pouco configurável e que exige atuação manual dos operadores periodicamente.

Para solucionar esse problema, este trabalho apresenta um serviço de bilhetagem configurável responsável por coletar periodicamente dados de uso dos recursos disponibilizados pela nuvem do LSD, armazenando-os de forma estruturada e enviando relatórios automaticamente para usuários e administradores.

O texto está estruturado de modo a fundamentar as limitações técnicas do shylock e apresentar como o serviço deste trabalho foi desenvolvido. Em seguida, são apresentados os conceitos e componentes principais do OpenStack, em especial aqueles que foram utilizados durante o desenvolvimento. Seguindo para a arquitetura da aplicação, é explorado como o serviço foi desenvolvido, incluindo arquivos de configuração, decisões de projeto e as ferramentas utilizadas. Por fim, é relatado quais as experiências e lições aprendidas, seguido de suas limitações e trabalhos futuros.

2 FUNDAMENTAÇÃO

Apesar de ter se mostrado útil durante os primeiros 5 meses, o shylock passou a não ser mais utilizado. Um dos motivos para isso ocorrer foi devido a dificuldades em sua operação. Em primeiro momento, o operador precisa executar um script que coleta dados de todas as instâncias da nuvem, armazenando suas informações em uma estrutura de dados. Esse serviço de coleta é demorado e não possui tolerância a falhas, ou seja, se durante o período de coleta ocorrer algum problema, todo o trabalho é perdido e o operador precisa executar tudo novamente. Após concluída esta etapa, é necessário executar um segundo script que processa, normaliza e agrupa os dados coletados por projetos e instâncias. Até que finalmente, o operador executa o último script para gerar os relatórios, para então, enviá-los manualmente por e-mail aos respectivos responsáveis.

Apesar de resolver parte do problema, o shylock exige que um operador da nuvem execute todos os scripts de forma manual. Além disso, o procedimento de coleta é pouco configurável, não permitindo operadores definirem quais métricas deverão ser coletadas. Ainda, como a ferramenta somente gera relatórios HTML, não é fácil disseminar os dados coletados entre os usuários da nuvem. Dados estes que poderão ser usados para maior controle entre os administradores, seja para consultar a relação de recursos alocados e disponíveis, ou gerar diferentes tipos de relatórios. Uma solução mais adequada para esse problema seria o uso de um serviço que realiza a coleta dos recursos de forma periódica, não limitando administradores a gerar apenas relatórios, mas também a realizar a filtragem dos recursos alocados entre os diferentes usuários da nuvem.

Este trabalho tem o mesmo objetivo do shylock: melhorar a eficiência na utilização dos recursos da nuvem privada do LSD. No entanto, todo o processo de coleta, processamento e envio de e-mails é feito de forma automática. Diferente do shylock que utiliza um arquivo de texto com os dados estruturados, a nossa solução utiliza dois bancos de dados, um SQL e outro temporal, o motivo dessa escolha é aprofundado em seções posteriores. Para controlar e coordenar toda a aplicação, administradores usarão dois arquivos de configuração, um deles com os dados mais sensíveis, que inclui credenciais de acesso aos bancos de dados e OpenStack, enquanto o segundo é um arquivo Yaml, que descreve a periodicidade e como deve ser feita a coleta dos recursos.

3 OPENSTACK

OpenStack é um sistema operacional open source que gerencia um conjunto de recursos de uma nuvem, tais como computação, armazenamento, rede, identidade e imagem, com o objetivo de provisionar uma infraestrutura como serviço. Ainda, por ter uma

arquitetura modularizada, é possível adicionar novos componentes como serviços, tais como orquestração, balanceamento de carga, monitoramento, dentre diversos outros.

Dividindo-se em cinco componentes principais, que serão descritos posteriormente, antes, é importante entendermos alguns conceitos OpenStack:

- Imagem: sistema operacional instalado e pré-configurado.
- Grupo de Segurança: conjunto de regras de firewall.
- Par de Chave: par de chaves assimétricas que permite acesso ao recurso através do protocolo SHH.
- Instância: uma máquina virtual, já com imagem, grupos de segurança e par de chaves inseridos.
- Flavor: são um conjunto de configurações pré-determinadas que definem as características de uma instância, como capacidade de processamento, armazenamento e memória.
- Volume: disco virtual que persiste dados de uma instância.
- Usuário: conjunto de regras de acesso para uma combinação de usuário e senha.
- Projeto: conjunto de instâncias, volumes e usuários.
- Domínio: conjunto de projetos.

Como ilustrado na Figura 1, uma implementação OpenStack é composta por diversos componentes que fornecem APIs para acessar os recursos da infraestrutura. Existem diversos tipos de componentes para resolver diferentes tipos de problema, no entanto, vamos focar apenas nos cinco principais, obrigatórios para a implementação do OpenStack, além de um sexto, responsável pelo monitoramento.

- Keystone: serviço responsável por fornecer autenticação e permissões de usuários aos recursos. Ele é quem gerencia todos os usuários, projetos e domínios.
- Nova: é o serviço que gerencia todos os recursos computacionais do OpenStack, provisionando instâncias, também chamadas de máquinas virtuais, assim como os flavors.
- Cinder: é responsável por fornecer volumes para instâncias do componente Nova.
- Glance: serviço responsável por gerenciar as imagens.
- Neutron: implementa o serviço de networking das máquinas. O Neutron é quem permite configurar o endereçamento e conectividade da nuvem, lidando com a criação e gerenciamento de uma rede virtual, incluindo componentes como switches, sub-redes, roteadores além de serviços mais robustos, como firewalls ou VPN.
- Monasca: fornece um serviço de monitoramento altamente escalável e tolerante a falhas. Com ele, é possível obter informações como taxa de entrada e saída de pacotes em uma instância ou projeto, assim como informações no que diz respeito a CPU, memória e armazenamento em disco.

Além dos conceitos já apresentados do OpenStack, na nuvem do LSD existe o conceito de sponsors. Estes são os usuários, tipicamente professores da Universidade Federal de Campina Grande (UFCG), responsáveis por um ou mais projetos, dessa forma, eles serão as pessoas para qual serão enviados os e-mails contendo os relatórios informando o uso dos recursos alocados.

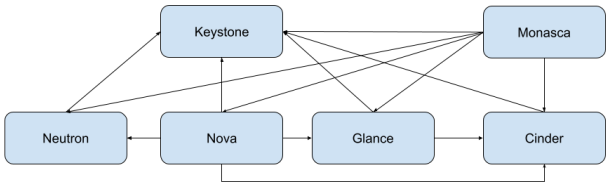


Figura 1: Comunicação entre os componentes OpenStack.

4 ARQUITETURA

A arquitetura da aplicação divide-se em três camadas principais, como ilustrado na Figura 2. O primeiro deles é o *collector*, nele o operador especifica quais métricas devem ser capturadas do Monasca, como por exemplo uso de CPU, memória RAM e rede. A camada de *databases* armazena esses dados normalizados, provendo uma interface simplificada para recuperar tais recursos. Nesse caso é utilizado dois tipos de banco de dados diferentes, um SQL e outro temporal. No banco de dados SQL será armazenado informações do OpenStack, tais como estado das instâncias, projetos e demais recursos, enquanto no banco de dados temporal, InfluxDB, serão armazenadas as métricas coletadas através do Monasca. Por fim, a última camada é o *worker*, que é responsável por gerar os relatórios a partir de templates Jinja, nos quais serão agregados os recursos especificados anteriormente e enviados ao usuário responsável pelo projeto. Jinja é um mecanismo de templates HTML para Python, com ele é possível incorporar lógica Python dentro de arquivos HTML, como pode ser visto na Figura 3.

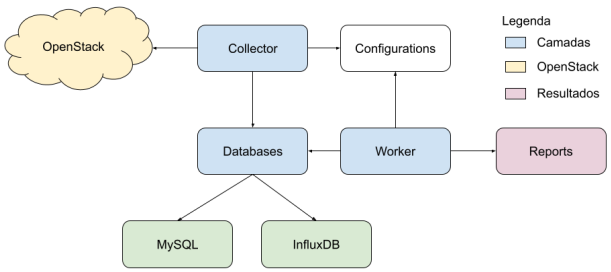


Figura 2: Visão geral da arquitetura.

Para iniciar a aplicação, o operador precisará fornecer dois arquivos de configuração. O primeiro será um arquivo no formato Yaml, como mostra na Figura 4, responsável por informar dados mais específicos sobre quais métricas, tamanho do grão e período de coleta que deve ser feito no Monasca, assim como o dia no qual deve ser enviado e-mail contendo a fatura final para os responsáveis. O segundo é um arquivo contendo variáveis de ambiente, especificado por um arquivo .env localizado na raiz do projeto, que armazena dados mais sensíveis, tais como informações de credenciais de acesso ao banco de dados SQL e Influx, além das credências do usuário OpenStack e do e-mail de origem no qual será feito o disparo dos relatórios.

```
{% extends "base.html" %}
{% block title %}Index{% endblock %}
{% block head %}
    {{ super() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
{% block content %}
    <h1>Index</h1>
    <p class="important">
      Welcome to my awesome homepage.
    </p>
{% endblock %}
```

Figura 3: Exemplo de código Python em um template Jinja.

```
billing:
  admin:
    mail: cloud@lsd.ufcg.edu.br
  users:
    send_report_on_day: 1

openstack:
  collect_period: 1

monasca:
  collect_period: 1
  statistics:
    - name: [
        vm.cpu.utilization_norm_perc,
        vm.mem.free_perc,
        vm.net.in_bytes,
        vm.net.out_bytes,
      ]
      type: avg
      dimension: resource_id
      period: 60
    - name: [
        cpu.percent,
        mem.free_perc,
      ]
      type: avg
      dimension: hostname
      period: 60
```

Figura 4: Modelo do arquivo de configuração Yaml.

4.1 Ferramentas

Para desenvolvimento do serviço, foi escolhido o framework Django, destinado ao desenvolvimento de aplicações web na linguagem de programação Python. Com mais de 15 anos de existência, o Django permite a integração de algumas ferramentas importantes para o desenvolvimento do serviço, a primeira delas é sua ORM, que provê uma interface única para os principais banco de dados SQL, como PostgreSQL, MariaDB, MySQL, Oracle e SQLite. Além disso,

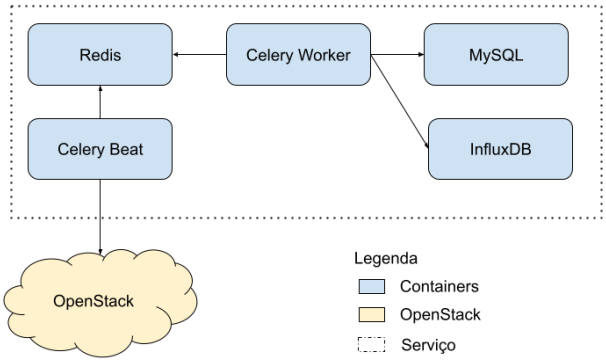


Figura 5: Comunicação entre containers.

o Django ainda possui integração nativa ao Celery: um sistema distribuído que permite executar mensagens de forma assíncrona. Seu foco está no processamento em tempo real, mas também suporta agendamento de tarefas.

O Celery é uma das ferramentas principais do serviço, pois todo o sistema de coleta é feito a partir dele. Fundamentalmente, o Celery divide-se em dois componentes principais: clients e workers. O client é o responsável por enviar uma tarefa, enquanto o worker executa essa tarefa. Toda essa comunicação é feita através de um broker[1], geralmente a escolha fica entre RabbitMQ e Redis. Para essa comunicação acontecer, o client insere novas tarefas na fila de mensagens, enquanto o worker consulta periodicamente a fila de mensagens em busca de novas tarefas. Ainda, o Celery possui um outro módulo chamado Beat. O Celery Beat é responsável por realizar o agendamento de tarefas, inserindo-as no broker.

Dessa forma, através do Celery Beat, o serviço periodicamente envia uma mensagem ao broker, informando que deve ser realizado uma nova consulta aos componentes do OpenStack para coletar as informações necessárias, enquanto os workers disponíveis serão os responsáveis por executar tais tarefas. Isso mostra-se uma vantagem, visto que por ter sua arquitetura distribuída e dependendo apenas do broker como gerenciador centralizado de mensagens, a depender do tamanho da nuvem OpenStack, é possível escalar o número de workers de forma horizontal, sendo necessário apenas adicionar novos workers.

Como os dados do Monasca possuem a característica de temporalidade, ou seja, dependem do tempo em que foram criados para terem sentido, o serviço também necessitou de um segundo banco de dados. Para isso, foi escolhido o InfluxDB, banco de dados temporal, o mesmo utilizado pelo Monasca para armazenar as métricas.

Para implantação do serviço foi utilizado containers Docker. Um container é uma unidade que empacota o código e todas as dependências de um aplicativo para que o mesmo seja executado de forma rápida e confiável em diversos ambientes. Dessa forma, o serviço pode ser implantado facilmente em diferentes sistemas operacionais, dispensando preparação do ambiente sempre que for

[1] "Um broker é como um corretor que atua como uma entidade intermediária, gerenciando transações entre entidades. Geralmente essas transações são especificadas como mensagens enviadas por clients e lida por workers."

feito uma nova implantação. A única exigência é que o ambiente tenha o Docker instalado.

4.2 Requisitos não-funcionais

No que tange pontos arquiteturais, o serviço foi desenvolvido tendo em mente os seguintes requisitos não-funcionais. O primeiro deles é a usabilidade, isso se reflete extraindo os principais arquivos de configuração em dois tipos, um deles com dados mais sensíveis e que deve ser guardado em sigilo, enquanto o segundo, no formato Yaml, que pode ser armazenado publicamente em ferramentas de controle de versão, tais como Git, por exemplo. Já no ponto de vista de implantação, pelo serviço ser estruturado em containers, é facilitada a portabilidade caso seja necessário alterar o ambiente no qual ele é executado.

Além disso, o serviço é tolerante a falhas e resiliente a momentos de downtime [2]. Isso permite que ele seja desativado a qualquer momento sem que exista perda de dados. Como o serviço realiza consultas periódicas na nuvem Openstack, as novas informações a respeito de projetos e instâncias serão consultadas no momento em que a aplicação for inicializada, enquanto para os dados do Monasca, referente às métricas dos recursos utilizados, será feita uma consulta no banco de dados do InfluxDB com o objetivo de encontrar qual data e hora da última métrica salva, a partir disso é realizada uma consulta no Monasca partindo dessa data até o dia atual. Caso o banco de dados InfluxDB esteja vazio ou será a primeira vez que o serviço será executado, serão salvas as métricas do dia atual em diante.

Podem ocorrer erros nas consultas periódicas no OpenStack, no entanto, como o Celery Beat empilha tarefas e estas ocorrem periodicamente com base no último registro, caso uma dessas tarefas falhe, ela não finalizará o serviço e a mesma consulta será recriada para ser processada na próxima iteração.

5 EXPERIÊNCIA E LIÇÕES APRENDIDAS

A realização deste trabalho possibilitou enriquecer o aprendizado em diferentes campos da computação, o primeiro deles em grande ascensão nos últimos anos: nuvem privada. Junto ao LSD, foi possível ter maior familiaridade com a operação diária de uma nuvem OpenStack, trabalhando de perto em como manter seus componentes com a maior disponibilidade possível. Ainda, pelo OpenStack ser implantado com diferentes componentes que funcionam em uma arquitetura em microsserviços, também foi necessário o aprofundamento de como é feita a mensageria entre eles. O estudo destes conceitos foram fundamentais para o desenvolvimento deste trabalho.

Por ter uma estrutura e documentação amplas, um dos maiores desafios foi conseguir estudar como os componentes OpenStack trocam mensagens e encaixar isso ao escopo do trabalho, visto que alguns componentes não deixavam claro como deveriam ser feitas as consultas em suas bibliotecas, por vezes sendo necessário ter que recorrer ao código fonte para compreender. O que nos levou a despender bastante tempo até entender como fazer as consultas da forma correta. No que tange o desenvolvimento, também foi um desafio aprender como realizar o processamento periódico das consultas ao OpenStack, ao mesmo tempo que era necessário garantir tolerância a falhas.

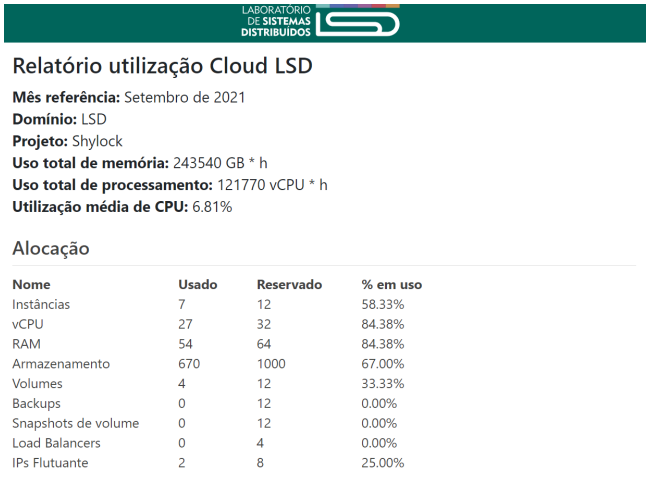
Tais pontos resultaram em algumas limitações do serviço atual, como não existir uma interface gráfica para operadores, assim como uma linha de comando no qual ele possa interagir diretamente, realizando consultas em tempo real ao serviço. O que limita sua interação apenas aos arquivos de configurações.

Em trabalhos futuros será desenvolvida uma interface gráfica no qual operadores possam não apenas realizar a configuração do serviço, mas também consultar informações a respeito dos recursos que estão alocados ou em uso. Ao mesmo tempo, os sponsors terão suas próprias credenciais no serviço, o que permitirá consultar qual o custo atual dos recursos alocados em cada projeto de forma individual ou agregada. Ainda, será realizado um estudo a respeito do quanto o serviço impactou na forma que os usuários utilizam a nuvem, mensurando se o novo serviço cumpriu com o seu objetivo ou se é necessário outra abordagem para o problema.

Appendices

A RELATÓRIOS

As figuras a seguir ilustram os relatórios produzidos pelo serviço. Eles são enviados para o e-mail de cada sponsor, enquanto o relatório com o agregado de todos os projetos será enviado por e-mail aos operadores da nuvem, e-mails estes que devem ser fornecidos através do arquivo de configuração Yaml.


LABORATÓRIO DE SISTEMAS DISTRIBUÍDOS

Relatório utilização Cloud LSD

Mês referência: Setembro de 2021
Domínio: LSD
Projeto: Shylock
Uso total de memória: 243540 GB * h
Uso total de processamento: 121770 vCPU * h
Utilização média de CPU: 6.81%

Alocação

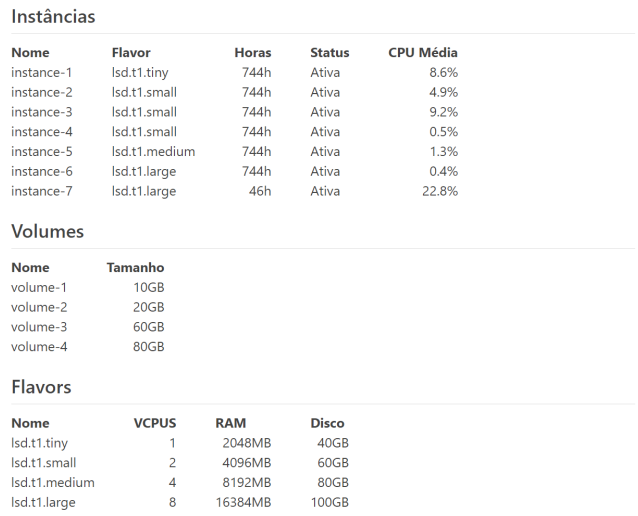
Nome	Usado	Reservado	% em uso
Instâncias	7	12	58.33%
vCPU	27	32	84.38%
RAM	54	64	84.38%
Armazenamento	670	1000	67.00%
Volumes	4	12	33.33%
Backups	0	12	0.00%
Snapshots de volume	0	12	0.00%
Load Balancers	0	4	0.00%
IPs Flutuante	2	8	25.00%

Figura 6: Modelo de relatório enviado aos usuários.

Como ilustrado na Figura 6, na primeira parte os usuários poderão visualizar o mês de referência, assim como o nome do projeto e o domínio no qual ele está cadastrado. Os campos de uso total de memória e processamento são obtidos ao calcular o produto entre a quantidade do respectivo recurso alocado e a quantidade de horas que a instância estava em uso. Além dessas informações, em percentual, ainda é possível obter a utilização média desse mesmo

[2] "Também conhecido como tempo de inatividade, downtime é o momento que um sistema ou serviço fica indisponível."

projeto, seguido do detalhamento dos recursos alocados e reservados, assim como o seu uso. Continuando o relatório, conforme ilustrado na Figura 7, os usuários ainda poderão visualizar o uso dos recursos de forma detalhada para cada instância usada naquele período, incluindo os flavors.



Instâncias

Nome	Flavor	Horas	Status	CPU Média
instance-1	lsd.t1.tiny	744h	Ativa	8.6%
instance-2	lsd.t1.small	744h	Ativa	4.9%
instance-3	lsd.t1.small	744h	Ativa	9.2%
instance-4	lsd.t1.small	744h	Ativa	0.5%
instance-5	lsd.t1.medium	744h	Ativa	1.3%
instance-6	lsd.t1.large	744h	Ativa	0.4%
instance-7	lsd.t1.large	46h	Ativa	22.8%

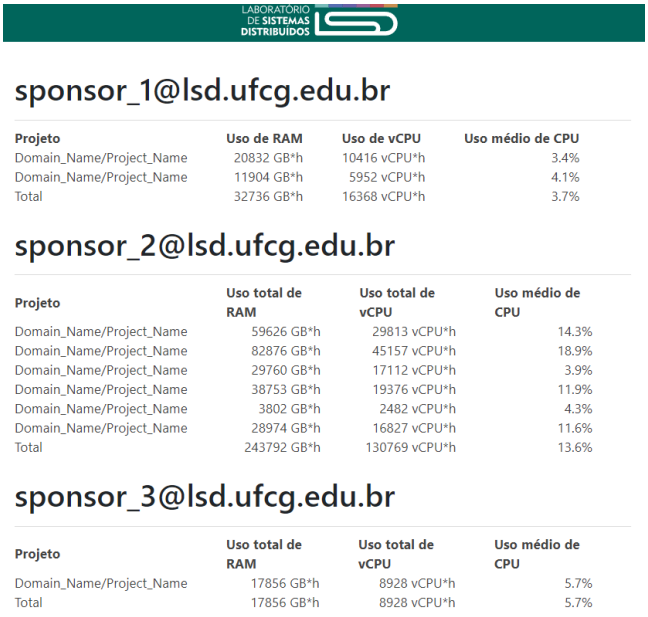
Volumes

Nome	Tamanho
volume-1	10GB
volume-2	20GB
volume-3	60GB
volume-4	80GB

Flavors

Nome	VCPUS	RAM	Disco
lsd.t1.tiny	1	2048MB	40GB
lsd.t1.small	2	4096MB	60GB
lsd.t1.medium	4	8192MB	80GB
lsd.t1.large	8	16384MB	100GB

Figura 7: Detalhamento de uso dos recursos pelos usuários.



LABORATÓRIO DE SISTEMAS DISTRIBUÍDOS

sponsor_1@lsd.ufcg.edu.br

Projeto	Uso de RAM	Uso de vCPU	Uso médio de CPU
Domain_Name/Project_Name	20832 GB*h	10416 vCPU*h	3.4%
Domain_Name/Project_Name	11904 GB*h	5952 vCPU*h	4.1%
Total	32736 GB*h	16368 vCPU*h	3.7%

sponsor_2@lsd.ufcg.edu.br

Projeto	Uso total de RAM	Uso total de vCPU	Uso médio de CPU
Domain_Name/Project_Name	59626 GB*h	29813 vCPU*h	14.3%
Domain_Name/Project_Name	82876 GB*h	45157 vCPU*h	18.9%
Domain_Name/Project_Name	29760 GB*h	17112 vCPU*h	3.9%
Domain_Name/Project_Name	38753 GB*h	19376 vCPU*h	11.9%
Domain_Name/Project_Name	3802 GB*h	2482 vCPU*h	4.3%
Domain_Name/Project_Name	28974 GB*h	16827 vCPU*h	11.6%
Total	243792 GB*h	130769 vCPU*h	13.6%

sponsor_3@lsd.ufcg.edu.br

Projeto	Uso total de RAM	Uso total de vCPU	Uso médio de CPU
Domain_Name/Project_Name	17856 GB*h	8928 vCPU*h	5.7%
Total	17856 GB*h	8928 vCPU*h	5.7%

Figura 8: Exemplo de relatório enviado aos administradores, agregando as informações de todos os projetos por sponsor.

A Figura 8 ilustra o template do relatório enviado aos administradores, agregando as principais informações de cada sponsor. Por serem configuráveis, os templates podem ser customizados através da sintaxe do Jinja, permitindo que os administradores removam ou adicionem novas informações.