Desafio
Engenheiro de Networking

Apresentação Técnica

Paulo Dias

Ambiente Cloud

Objetivo

Implantar uma infraestrutura para provisionar um ambiente multicloud que irá rodar duas aplicações em cloud provider distintos.

Requisitos

- A comunicação entre a aplicação 1 (cloud provider 1) e a aplicação 2 (cloud provider 2), deverá ser feita de forma privada, exemplo VPN;
- Ambos os servidores devem responder pelo mesmo domínio de DNS;
- O acesso ao serviço web do servidor 1 deverá ser restrito somente ao servidor 2;
- A aplicação 2 devera chamar aplicação 1 através de DNS, e não por IP;

Ambiente Cloud

Aplicação 1 (HTTP Server)

Informações:

- Cloud Provider: AWS
- Sistema Operacional: Linux
- Web Server: Apache
- DNS Server: Bind
- Linguagem de Programação: Python

Aplicação de Teste:
Ativar o serviço de HTTP na porta 8080 para responder requisições GET originadas do endereço IP do servidor 2.

Ambiente Cloud

Aplicação 1 (HTTP Server)

Código:

```
# app1.py

from http.server import SimpleHTTPRequestHandler

import socketserver

# set ip and port to listen (vpn)
http_port = 8080
http_ip = "10.0.0.1"

class Handler(SimpleHTTPRequestHandler):
    server_version = 'Lab Server'
    sys_version = 'Simple HTTPS'

handler = Handler
httpd = socketserver.TCPServer((http_ip, http_port), handler)
print("Starting HTTP Server", "on Port:", http_port)

if __name__ == '__main__':
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass

    httpd.server_close()
```

Ambiente Cloud

Aplicação 2 (HTTP Server)

Informações:

- Cloud Provider: GCP
- Sistema Operacional: Linux
- Web Server: Apache
- DNS Server: Bind
- Linguagem de Programação: Python

Aplicação de Teste:
Envia requisições GET para a nome de domínio do servidor 1 na porta 8080 e exibe na tela as informações sobre a endereço de destino, HTTP port, tempo de resposta e o status code da requisição.

Ambiente Cloud

Aplicação 2 (HTTP Server)

Código:

```
# app2.py

import requests
import time

# set address request
address_request = "http://srv1-aws.empresa.com:8080"
interval_request = 2

def httpget(address):
    request_start = time.time()
    response = requests.get(address)
    response_latency = time.time() - request_start
    http_status_code = response.status_code
    print('Request:', address, 'HTTP Response:', http_status_code, 'Request Time:',
'{:.2f}'.format(response_latency), 'ms')
    #print(response.headers)
    #print(response.text)

if __name__ == '__main__':
    while True:
        time.sleep(interval_request)
        httpget(address_request)
```

Ambiente Cloud

Outras Configurações

Configuração do DNS

- ✓ srv1-aws.empresa.com
- ✓ srv2-gcp.empresa.com

Configuração do Localhosts (/etc/hosts):
Como a comunicação deverá ser realizada via VPN e através DNS será necessário configurar os nomes dentro do localhosts.

```
# The following lines are desirable for IPv6 capacle hosts
::1                         ip6-localhost ip6-loopbacl
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
fe02::1       ip6-allnodes
fe02::2       ip6-allrouters


10.0.0.1 srv1-aws srv1-aws.empresa.com
10.0.0.2 srv2-gcp srv2-gcp.empresa.com
```

Ambiente Cloud

Outras Configurações

Segurança (Firewall) :

Para garantir a segurança será restrito na aplicação o envio de resposta apenas para o endereço privado (VPN) do servidor 2. Para aumentar ainda mais a segurança pode-se criar regras de firewall (iptables) restringido a comunicação baseado na origem e destino.

```
#Drop everything
iptables -P INPUT DROP
iptables -P OUTPUT DROP

#Allow VPN Address
iptables -A INTPUT -p tcp -s 10.0.0.2 --dport 8080 -j ACCEPT
iptables -A OUTPUT -p tcp -s 10.0.0.2 --dport 8080 -j ACCEPT
```

Ambiente Cloud

Tecnologias Utilizadas

Terraform: Utilizado para provisionamento da infraestrutura de cloud (Servidor Linux, Interface de Rede, VPN, Firewall, etc.)

Exemplo de códigos terraform.

```
# Configure the AWS Provider
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
}
provider "aws" {
  version = "~> 3.0"
  region  = "us-east-1"
}
# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "172.16.0.0/20"
}
```

```
# Configure the GCP Provider
provider "google" {
  region  = "us-central1"
  zone    = "us-central1-c"
}
resource "google_compute_instance" "vm_instance" {
  name         = "terraform-instance"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
    }
  }
  network_interface {
    network = "172.17.0.0/20"
    access_config {
}
```

Ambiente Cloud

Tecnologias Utilizadas

Ansible: Utilizado para provisionamento, gerenciamento das configurações e implantação das apliações desenvolvidas.

Exemplo de códigos ansible.

```
# Basic provisioning example
- name: Create AWS resources
  hosts: localhost
  connection: local
  gather_facts: False


  tasks:
  - name: Create a security group
    ec2_group:
      name: ansible
      description: "Ansible Security Group"
      region: "{{aws_region}}"
      vpc_id: "{{vpc_id}}"
      aws_access_key: "{{aws_access_key}}"
      aws_secret_key: "{{aws_secret_key}}"
```

```
# Basic provisioning example
rules:
        - proto: all
          cidr_ip: {{cidr_ip}}
        - proto: all
          group_name: ansible
      rules_egress:
        - proto: all
          cidr_ip: 0.0.0.0/0
    register: firewall
```

Ambiente Cloud

Log APP 1 / APP 2

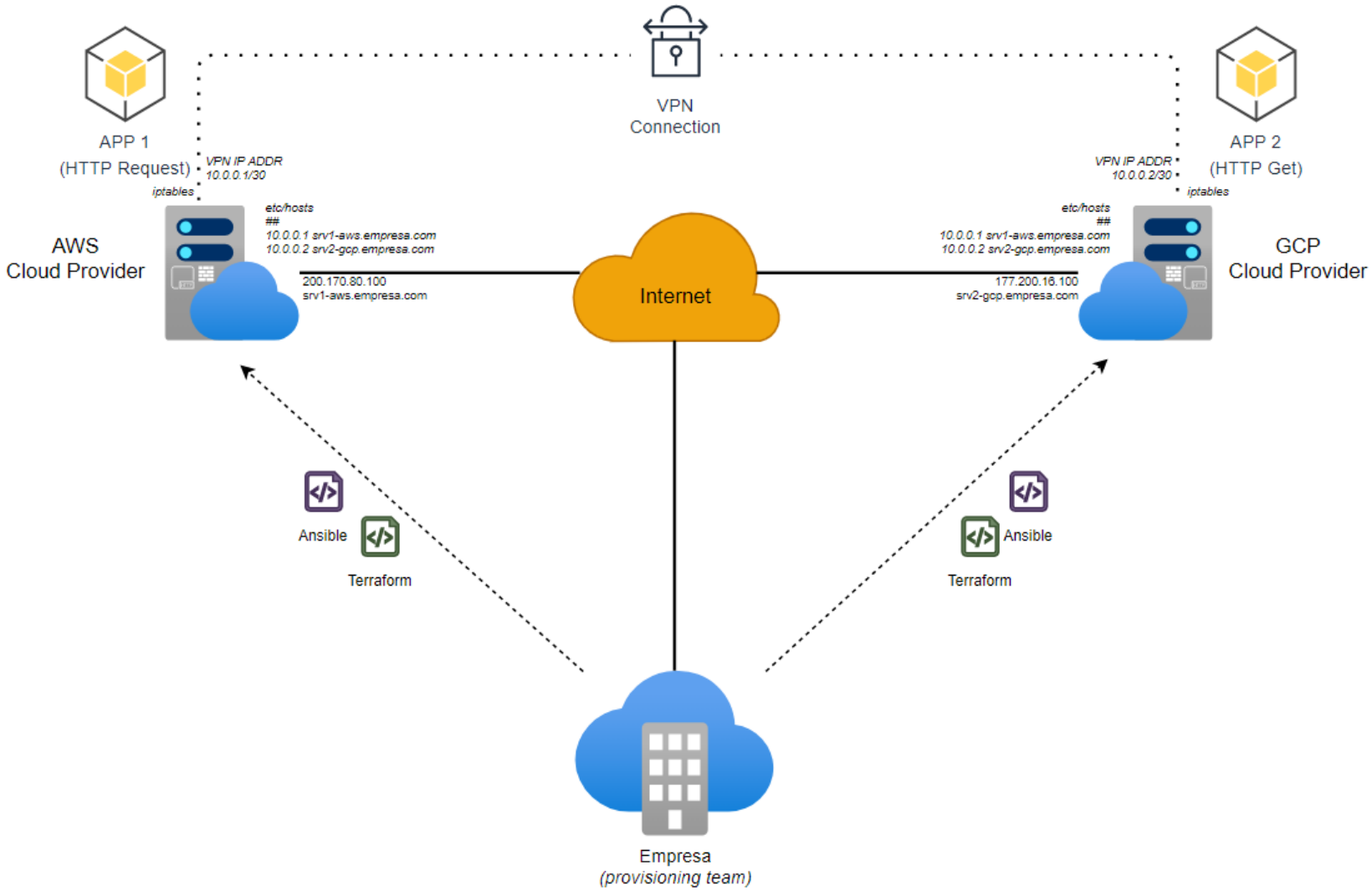
Saída esperada das aplicações 1 e 2:

```
#LOG APP1
(requester) userlabs@sandbox:~/Devel/local/requester/app$ python3 app1.python3
Starting HTTP Server on Port: 8080
10.0.0.2 - - [22/Jul/2021 19:37:44] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:46] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:48] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:50] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:52] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:54] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:56] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [22/Jul/2021 19:37:58] "GET / HTTP/1.1" 200 -
```

```
#LOG APP2
(requester) userlabs@sandbox:~/Devel/local/requester/app$ python3 app2.python3
Starting HTTP Server on Port: 8080
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.01 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.00 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.00 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.00 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.01 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.00 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.01 ms
Request: http://srv1-aws.empresa.com:8080 HTTP Response: 200 Request Time: 0.00 ms
```

Ambiente Cloud

Topologia



Arquitetura de Rede

Objetivo

Propor uma arquitetura de rede para suportar o projeto de migração de workload do Datacenter atual (onpremises) para o ambiente de cloud.

Requisitos

- Manter o acesso as aplicações, base de dados e demais serviços estável e sem interrupção;
- Otimizar custos mantendo a alta disponibilidade e qualidade dos links de todas unidades;
- Garantir a segurança no acesso das informações e na infraestrutura de rede LAN / WLAN.
- Prazo de execução estimado do projeto: 18 meses;

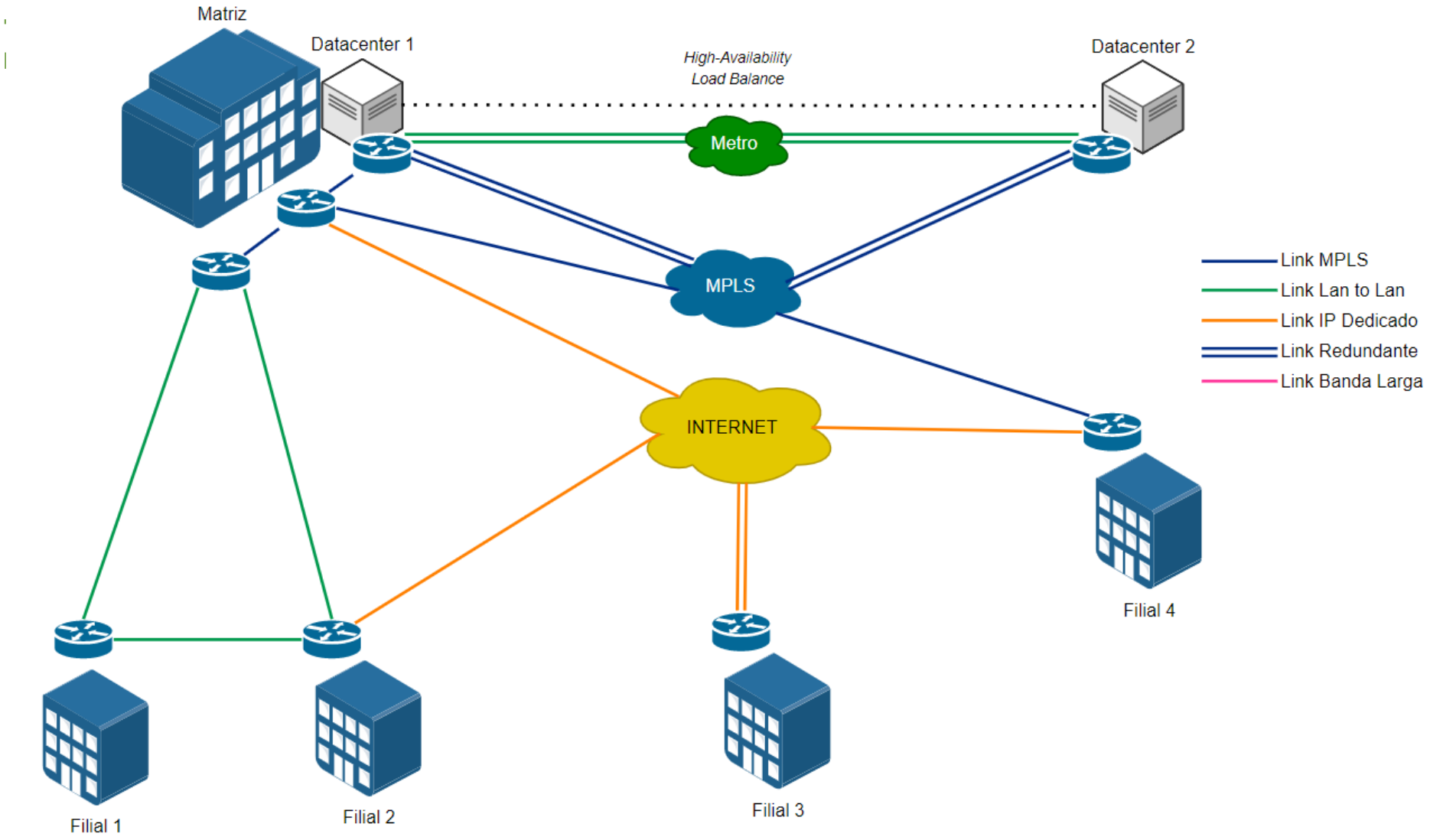
Arquitetura de Rede

Topologia Atual

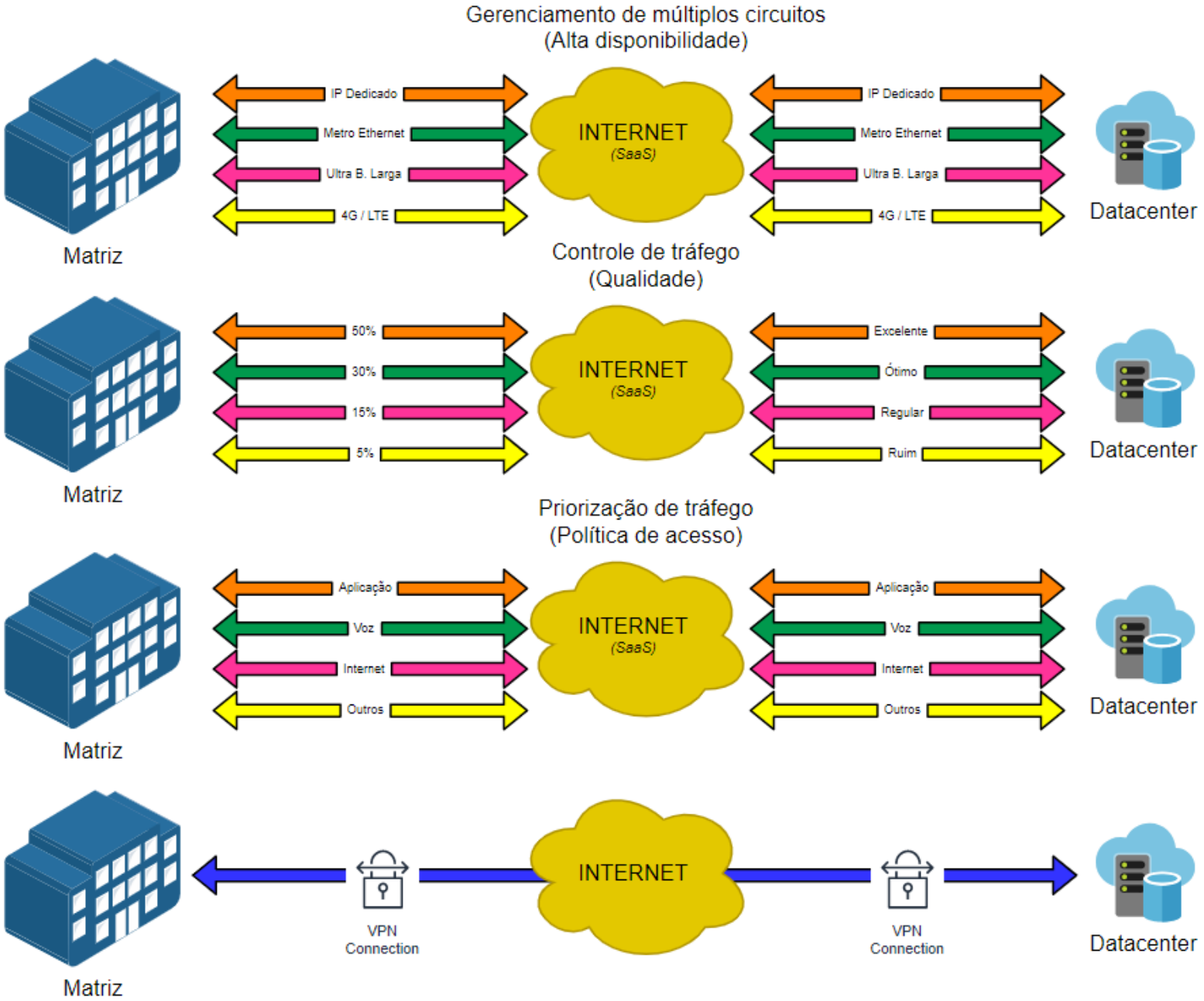
Descrição da topologia atual “sugerida”:

- Datacenter 1 está localizado dentro da Matriz;
- Datacenter 2 utiliza infraestrutura de um fornecedor local;
- Datacenter 1 e 2 tem link lan to lan dedicado para comunicação com dupla abordagem;
- Rede da matriz está conectada com a nuvem MPLS da operadora;
- Os links MPLS tem abordagens físicas distintas com resiliência para garantir alta disponibilidade;
- Redundância de equipamentos na camada CORE e link IP dedicado com mais de um ASN;
- Filias possuem links lan to lan com redundância, sendo algumas localidades com link IP dedicado com um ou mais fornecedor;

Arquitetura de Rede



SD-WAN


Gerenciamento de múltiplos circuitos
(Alta disponibilidade)
IP Dedicado
Metro Ethernet
Ultra B. Larga
4G / LTE
INTERNET
(SaaS)
IP Dedicado
Metro Ethernet
Ultra B. Larga
4G / LTE
Datacenter
Matriz
Controle de tráfego
(Qualidade)
50%
30%
15%
5%
INTERNET
(SaaS)
Excelente
Ótimo
Regular
Ruim
Datacenter
Matriz
Priorização de tráfego
(Política de acesso)
Aplicação
Voz
Internet
Outros
INTERNET
(SaaS)
Aplicação
Voz
Internet
Outros
Datacenter
Matriz
VPN
Connection
INTERNET
VPN
Connection
Datacenter
Matriz

Arquitetura de Rede

Arquitetura Proposta

Com a adoção dos serviços na Nuvem (migração do workload) a área de rede pode reduzir os custos operacionais e melhorar a utilização dos recursos de conectividade multisite. A arquitetura SD-WAN permite usar a largura de banda de maneira mais eficiente e ajuda a garantir altos níveis de desempenho para aplicações essenciais, sem sacrificar a segurança ou a privacidade dos dados.

Esta arquitetura permite que a empresa adote aplicações em nuvem na forma de SaaS e IaaS, provendo o caminho do tráfego de acesso às aplicações distribuídas. Além disso oferece ferramentas de segurança integrada, como por exemplo NGFW, fornecendo recursos robustos, com um custo mais eficiente.

Um dos principais recursos do SD-WAN é a capacidade de usar várias opções de conectividade simultaneamente e de forma inteligente, otimizando a experiência do usuário, simplificando e automatizando operações e o gerenciamento que visam melhorar o desempenho da aplicação.

Existem diversos fornecedores de SD-WAN no mercado que atendem bem esta demanta, tais como: Meraki (Cisco), Fortinet, Sonicwall, etc...

Arquitetura de Rede

Arquitetura Proposta

Vantagens e Benefícios do SD-WAN:

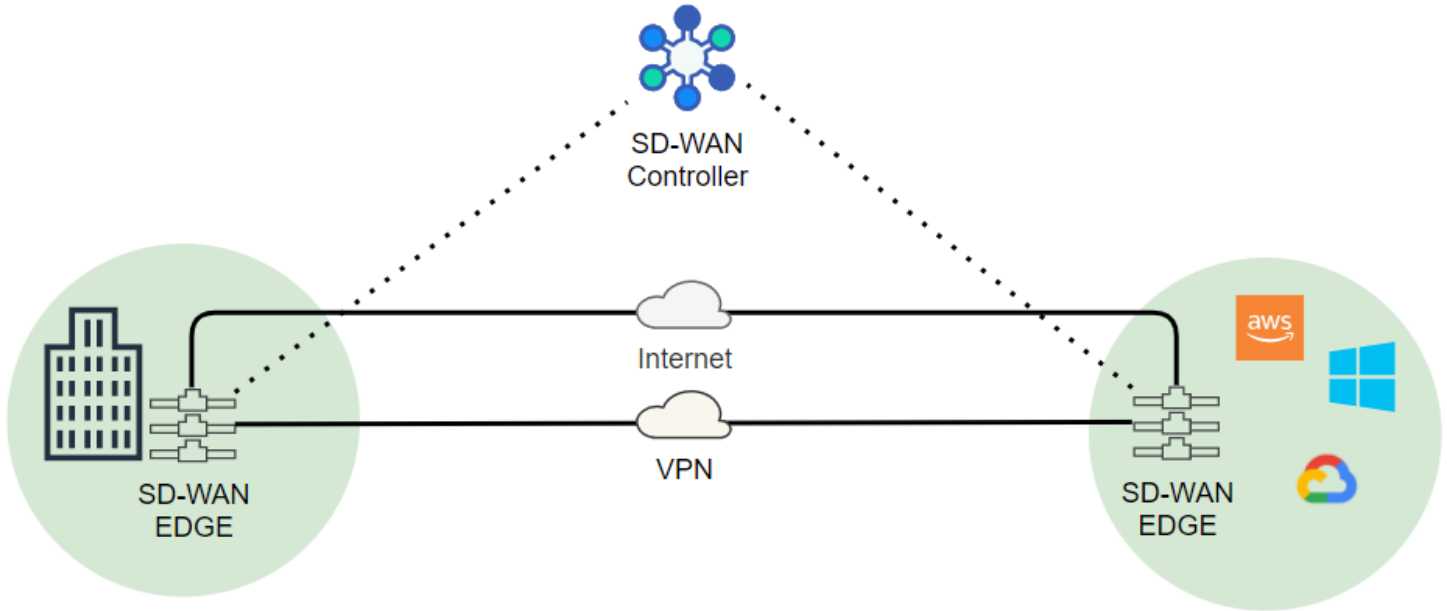
- Priorização da dinâmica de tráfego;
- Bloqueio de tráfego de acordo com protocolos de segurança;
- Redirecionamento instantâneo de tráfego em caso de queda ou instabilidade de uma ou mais conexões;
- Alta disponibilidade, com serviço previsível, de todas as principais aplicações empresariais;
- Redução de custos com os serviços de MPLS (Multiprotocol Label Switching) por conexões mais econômica e flexível (incluindo conexões VPN seguras);
- Segurança distribuída para a filial e os endpoints remotos (NGFW);
- Gerenciamento único e centralizado, provisionamento automatizado, relatórios das aplicações e desempenho de conectividade (WAN);

Arquitetura de Rede

SD-WAN Controller

O controlador SD WAN é um componente que permite gerenciar todos os recursos, dispositivos físico ou virtual para toda a rede associada ao controlador. Isso inclui a configuração e ativação, gerenciamento de endereço IP, envio de políticas, etc.

Você pode instalar novas localidades, gerenciar sua rede, bem como implementar políticas em Gateways ou EDGES.



Arquitetura de Rede

Segurança - SASE

O SASE (Secure Access Service Edge) é uma estrutura de segurança para serviços em nuvem que permite que usuários e dispositivos tenham acesso seguro à seus aplicativos, dados e serviços.

Com a adoção da nuvem, mais dispositivos, aplicativos e serviços são usados fora da infraestrutura local, aumentando o perímetro da empresa. Esse novo perímetro dinâmico exige uma nova abordagem de segurança e o gerenciamento dos riscos.

O SASE permiti que a área de segurança da empresa aplique identidade e contexto para especificar o nível de desempenho, confiabilidade, segurança e custo desejado para cada sessão da rede.

Seus principais benefícios são redução da complexidade, agilidade, aumento do desempenho (latência), transparência, política centralizada, menor custo, etc.

Arquitetura de Rede

Topologia Proposta