DlteC do Brasil®
www.dltec.com.br
info@dltec.com.br|413045.7810

DLTEC DO BRASIL

AUTOMAÇÃO E PROGRAMABILIDADE DE REDES (TÓPICO 6.0 DO CCNA 200-301)

Automação e Programabilidade de Redes

O conteúdo dessa apostila é uma adaptação da matéria online do Curso Automação e Programabilidade de Redes.

Para mais informação visite www.dltec.com.br

Seja muito bem-vindo(a) ao curso Automação e Programabilidade de Redes em Redes Cisco, o qual faz parte da trilha da certificação Cisco CCNA 200-301, da DItec!

Aqui, você terá todo o background necessário para aprender sobre automação de redes, SDN, SD-Access e programabilidade com foco no CCNA e ser aprovado(a) no exame 200-301 da Cisco ao final da trilha. O exame citado anteriormente é conhecido também como exame CCNA ou Cisco Certified Network Associate.

Os assuntos encontram-se distribuídos conforme o Blueprint do exame – sendo assim, esteja bastante atento(a) a todo o conteúdo que aqui será apresentado. Não perca de vista o peso de cada tópico – isso é importante para você ter uma noção de quanto investirá o seu tempo em cada um.

Busque praticar o máximo de exercícios possíveis e, além disso, busque compreender cada assunto de maneira objetiva. Não esqueça o propósito principal: ser aprovado(a).

A DlteC estará com você em todos os momentos dessa jornada!

Bons estudos!

Introdução

Olá!

Como parte integrante da Trilha para a Certificação **CCNA 200-301** da DlteC do Brasil, esta apostila representa uma adaptação textual do material disponibilizado online do **Curso Automação e Programabilidade de Redes**.

O conteúdo desse curso cobre o tópico 6.0 (Automation and Programmability) da certificação Cisco CCNA 200-301.

Por isso, recomendamos que você a utilize como um importante recurso offline. Combinando-a com o conteúdo online, você estará muito mais bem preparado(a) para realizar o exame **200-301** (**CCNA**: **Cisco Certified Network Associate**).

É de suma importância que você, além de participar dos fóruns, realize o máximo possível de exercícios e simulados (todos encontrados na trilha do 200-301 Online).

Lembre-se que o curso Fundamentos de Redes Cisco é Pré-requisito para esse curso.

Esperamos que você aproveite ao máximo este material, que foi idealizado com o intuito verdadeiro de fazê-lo(a) obter êxito no exame. Estamos torcendo pelo seu sucesso!

Bons estudos!

Automação e Programabilidade de Redes

Peso: 10%

Objetivos

Ao final desse curso você deverá ter conhecimentos sobre:

- Como a automação afeta o gerenciamento de uma rede
- A diferença entre as redes tradicionais e redes baseadas em controladoras
- Características das arquiteturas baseadas em controladoras e definidas por software (conceito de overlay, underlay e fabric)
- Separação do plano de controle e plano de dados (Data Plane e Management Plane)
- APIs para norte e sul (North-bound e South-bound)
- Entender a diferença do gerenciamento de dispositivos de forma tradicional e o gerenciamento de dispositivos via Cisco DNA Center
- Características das APIs baseadas em REST (CRUD, verbos HTTP e codificação de dados)
- Reconhecer os recursos dos mecanismos de gerenciamento da configuração utilizando Puppet, Chef e Ansible
- Interpretar dados codificados no formato do JSON

Sumário

1 Introdução

Bem-vindo ao **Curso Automação e Programabilidade de Redes**, o qual também faz parte do conteúdo preparatório para a prova de certificação **CCNA 200-301**.

O curso **Automação e Programabilidade de Redes** possui como objetivo fornecer ao aluno uma visão abrangente sobre os principais conceitos que um CCNA precisa para entender e se adequar a esse novo panorama de redes programáveis.

Ao final do curso, você deverá ser capaz de entender:

- Como a automação afeta o gerenciamento de uma rede
- A diferença entre as redes tradicionais e redes baseadas em controladoras
- Características das arquiteturas baseadas em controladoras e definidas por software (conceito de overlay, underlay e fabric)
- Separação do plano de controle e plano de dados (Data Plane e Management Plane)
- APIs para norte e sul (North-bound e South-bound)
- Entender a diferença do gerenciamento de dispositivos de forma tradicional e o gerenciamento de dispositivos via Cisco DNA Center
- Características das APIs baseadas em REST (CRUD, verbos HTTP e codificação de dados)
- Reconhecer os recursos dos mecanismos de gerenciamento da configuração utilizando Puppet, Chef e Ansible
- Interpretar dados codificados no formato do JSON

Mesmo que você não esteja trilhando os estudos para a certificação CCNA 200-301 você pode sim fazer esse curso para aumentar seus conhecimentos no mundo de Redes e mais especificamente sobre os itens citados na lista acima.

Mas se você está na trilha da certificação, saiba que esse curso aborda o **Tópico 6.0 ou "Automation and Programmability"**, o qual corresponde a **10% das questões do exame CCNA 200-301**.

Como a nova prova terá aproximadamente entre 100 e 120 questões, podemos dizer que **devem cair de 10 a 12 questões** relacionadas ao conteúdo desse curso, dependendo da quantidade total de questões que forem sorteadas para seu exame específico.

Não esqueça que ao final do curso você poderá emitir o seu certificado!

1.1 Sobre a Cisco e o CCNA - Cisco Certified Network Associate

A Cisco é uma empresa líder mundial em TI e redes, tendo seus produtos e tecnologias utilizadas por diversas empresas dos mais variados segmentos de mercado no mundo todo.

Fundada em 1984 por Len Bosack e Sandy Lerner atua até os dias de hoje com tecnologia de ponta e inovações que auxiliam no crescimento do mercado de TI.

A Cisco atua na área de Redes (com os famosos Roteadores e Switches), Software, Internet das Coisas, Mobilidade e Comunicação sem fio, Segurança, Colaboração (Voz e Vídeo sobre IP), Data Center, Cloud, Pequenos e Médios Negócios e Provedores de Serviço.

Para garantir que os profissionais que atuam com seus produtos e tecnologias realmente tem os conhecimentos técnicos necessários para desempenhar um bom trabalho, a Cisco desenvolveu um programa de **Certificação** com **Três Níveis** no início:

- **Associate ou CCNA (Cisco Certified Network Associate)**
- Professional ou CCNP (Cisco Certified Network Professional)
- Expert ou CCIE (Cisco Certified Internetwork Expert)

Mais especificamente falando da certificação **CCNA ou Cisco Certified Network Associate** é uma das primeiras certificações lançadas pela Indústria de Redes e com certeza a mais famosa até os dias de hoje.

A primeira versão de CCNA data de 1998 chamada de 640-407, o qual foi atualizado sete vezes até a última mudança feita em 2016 com a versão 200-125 (CCNA Routing and Switching em uma prova) e as versões do 100-105 e 200-105 (Modelo em duas provas: CCENT/ICND-1 + ICND-2).

Em **julho de 2019** foi anunciada uma grande mudança em maioria das certificações Cisco e o CCNA volta ao que era no início, sendo uma certificação unificada para diversas áreas e englobando não somente assuntos de Roteamento e Switching, mas também segurança, redes sem fio e automação de Redes.

Esse curso que você está prestes a iniciar faz parte da nossa trilha para a certificação **CCNA 200-301**.

O que se espera de um CCNA no mercado de trabalho?

Um profissional certificado CCNA deve conhecer uma larga gama de tecnologias e configurações de diversos equipamentos Cisco, tais como Roteadores, Switches, Access Points e Wireless LAN Controlers.

Além disso, deve estar preparado para a nova geração da Infraestrutura de TI, a qual a automação e programabilidade será cada vez mais utilizada.

Não confunda programabilidade com a necessidade de ser um programador, pois um profissional CCNA no mercado faz a operação e manutenção da Rede, não necessariamente precisará ser um programador e sim entender como utilizar algumas ferramentas e interagir com APIs.

É o primeiro passo de uma carreira promissora e que tem muitas possibilidades de crescimento nas mais diversas áreas de tecnologia de rede.

A seguir vamos falar sobre como a preparação para o **CCNA** está dividida no **Portal da DlteC** e como você deverá utilizar nosso material para conquistar sua certificação.

1.2 Plano de Estudos para o CCNA

Nesse novo modelo de prova existe apenas um caminho para obtenção da certificação CCNA que é através do exame 200-301, ou seja, não existe mais opção em duas provas como na versão anterior.

O **plano de estudos** para você ter sucesso na **CCNA** é o seguinte:

1. Ativar a trilha do curso **CCNA 200-301** no Menu Cursos (somente se você ainda não ativou)
2. Estudar o conteúdo de cada Capítulo dentro da trilha (sequência de capítulos/cursos express a seguir)
3. Repetir os comandos e demonstrações práticas realizadas pelo Prof. Marcelo durante as vídeo aulas como laboratório
4. Fazer os simulados que estão dentro do curso "**CCNA 200-301**"
5. A qualquer momento tirar as dúvidas do conteúdo utilizando os fóruns correspondentes de cada capítulo (*)
6. Passar para o próximo capítulo
7. Realizar a prova Final para treinar e obter o certificado do curso CCNA 200-301 (média da aprovação igual ou acima a 70 pontos em um total de 100)
8. Fazer o preparatório Final com laboratórios e questionários (em inglês) específicos para a certificação
9. Agendar a prova e realizá-la

O exame CCNA 200-301 é composto por uma prova em computador que pode ter de 100 a 120 questões (depende do sorteio que é feito por candidato).

Essas questões devem ser resolvidas em 120 minutos no dia do exame.

Cada um dos capítulos do curso um **Peso** associado na prova e quanto maior o peso, maior será a quantidade de questões desse assunto no exame, sendo que seguimos as recomendações da Cisco na divisão de questões para que você treine em um ambiente o mais real possível.

Para ser aprovado(a), você deverá conseguir obter entre 800 e 850 pontos de um máximo 1000 pontos no exame.

Se você ativou esse curso com o objetivo de tirar a certificação então a partir de agora, foco total no objetivo: **OBTER A CERTIFICAÇÃO.**

Você será aprovado(a) – já coloque isso "na cabeça".

Para isso, pratique os comandos, leia os tópicos com cautela e, de preferência, marque logo o dia do seu exame (para você já ter uma data limite).

Faça o seu cronograma, estipule as horas de estudo e, sinceramente, não tem erro.

Repita essa frase todos os dias: **Eu serei aprovado(a).**

Se você assumir esse compromisso com sinceridade e vontade de vencer, tudo **dará certo**.

Estamos ao seu lado! Bons estudos!

(*) Os fóruns do curso são exclusivos para TIRAR AS DÚVIDAS DO CURSO, caso você tenha dúvidas do dia a dia ou que não tenham correlação com o curso utilize os grupos do Facebook ou Telegram para troca de ideias.

1.3 Como Estudar com o Material da DlteC

Nesse curso você terá **vídeo aulas**, **material de leitura** e **laboratórios em simuladores** para o aprendizado do conteúdo.

Posso somente ler ou assistir aos vídeos? NÃO RECOMENDAMOS!

O ideal é você assistir aos vídeos e na sequência ler os conteúdos ou...

... se preferir leia os conteúdos e depois assistia aos vídeos, tanto faz.

POR QUE LER E ASSISTIR?

Simples, porque **um conteúdo complementa o outro**. Principalmente se você está se preparando para a prova de certificação é crucial que você tanto veja os vídeos como a matéria de leitura!

Os questionários ou simulados com questões de prova estão dentro da estrutura da trilha do CCNA 200-301.

Siga a sequência sugerida no plano de estudos e **faça os questionários apenas depois** de ter lido, assistido aos vídeos e feito os laboratórios em simulador. Assim você terá um aproveitamento muito melhor do curso.

2 Tópicos do Curso vs Blueprint do CCNA 200-310 – Peso 10%

2.1 Introdução

Na tabela abaixo seguem os itens do blueprint ou conteúdo do exame Cisco CCNA 200-301 relacionados ao conteúdo do curso. Os capítulos que não aparecem explicitamente aqui fazem parte da matéria e complementam o aprendizado. Estude TODO o conteúdo do curso.

Automation and Programmability	
AUTOMAÇÃO E PROGRAMABILIDADE DE REDES	**Capítulos do Curso**
6.1 Explain how automation impacts network management	Impacto da Automação no Gerenciamento das Redes
6.2 Compare traditional networks with controller-based networking	Redes Tradicionais versus Controller Based Networks
6.3 Describe controller-based and software defined architectures (overlay, underlay, and fabric)	Software-Defined Access e sua Arquitetura
6.3.a Separation of control plane and data plane	Separação dos Planes
6.3.b North-bound and south-bound APIs	Controladoras e Arquitetura Centralizada
6.4 Compare traditional campus device management with Cisco DNA Center enabled device management	Software-Defined Access e sua Arquitetura
6.5 Describe characteristics of REST-based APIs (CRUD, HTTP verbs, and data encoding)	REST-based APIs, CRUD, Verbos HTTP e Data Encoding
6.6 Recognize the capabilities of configuration management mechanisms Puppet, Chef, and Ansible	Puppet, Chef e Ansible na Automação de Redes
6.7 Interpret JSON encoded data	Interpretando Dados Codificados em JSON

3 Redes Tradicionais versus Controller Based Networks

3.1 Introdução

Deixa eu começar essa história de **Controller Based Network** ou **Redes Baseadas em Controladoras** com algo que eu assisti em um vídeo onde o apresentador comentava que essa evolução das redes tradicionais, onde tudo ficava na infraestrutura do cliente, separado em redes e servidores físicos muito amarrados, para servidores virtualizados em data centers e depois para o conceito de nuvem acabou criando uma "ansiedade", pois no mundo virtualizado e em nuvem de um data Center tornou-se muito flexível, mas a rede não acompanhou esse ritmo e continua praticamente a mesma como nos primórdios.

Muitos switches têm sistemas operacionais proprietários, sem a possibilidade de instalação de aplicativos e com comandos de configuração que variam conforme o fabricante.

Por exemplo, se algo no OSPF não funciona bem na sua rede o que fazer? Nada, você não pode alterar o funcionamento do OSPF, ou seja, como dizia o querido treinador Zagalo "você vai ter que engolir o OSPF", ponto final.

Vamos a outro exemplo, você precisa aplicar políticas de segurança que foram requisitadas pela empresa em todos os dispositivos, ou seja, roteadores, switches, firewalls, APs e demais dispositivos de redes necessitam ser atualizados ou reconfigurados. Nessa empresa temos 1000 pontos com 5000 dispositivos de rede no total.

Você precisará nesse caso elaborar a configuração em um script em txt para aplicar via CLI, normalmente cada tipo de equipamento vai necessitar de uma configuração diferente, temos portas diferentes, com várias nomenclaturas, sem contar que se sua empresa tiver vários fabricantes teremos ainda comandos diferentes.

Em resumo: caos total e muito trabalho braçal!

Por exemplo, utilizando uma rede baseada em controladora, como no SDN ou Software Defined Networking (rede definida por software ou rede programável), traz um conceito parecido com o que estudamos sobre virtualização e nuvem, que é flexibilizar e possibilitar que inovações sejam realizadas mais facilmente na rede.

O SDN, quase que um sinônimo para redes baseadas em controladoras, traz uma visão diferente, sugerindo que os dispositivos sejam controlados de fora e possam até ter seu comportamento alterado frente a um tráfego conforme configurado em um aplicativo externo.

Ao invés de cada dispositivo ter seu "cérebro" na caixa eles são controlados por um dispositivo externo chamado "controladora" ou "controller", o qual acaba sendo similar aos hypervisors, que abstraem o hardware e possibilitam que ele seja dividido em máquinas virtuais, o controller pode comunicar-se com aplicativos externos que possibilitarão controlar a rede conforme a necessidade das aplicações da rede e não pelas necessidades da rede por si só.

Se você é bom observador deve notar a rede atual é meio "egoísta", pois as aplicações e aplicativos que fornecem serviços aos usuários muitas vezes precisam adequar-se ao que existe ou então os administradores de rede precisam fazer "gambiarras" para resolver os problemas de falta de flexibilidade da rede.

O SDN traz o foco para a aplicação, pois precisa haver um equilíbrio entre rede e aplicação para que o máximo proveito seja tirado dos serviços.

Um exemplo do que o SDN pode trazer de benefício é para implantações de rede, tradicionalmente o que levaria semanas de configuração (provisionamento ou "Provision") pode ser feito em horas ou até minutos.

A ideia central do SDN é a separação do control plane (camada de controle) da camada de encaminhamento (data plane), mas vamos estudar esses termos e suas funções no próximo tópico.

Existe também outra visão do SDN onde a inteligência permanece nos dispositivos das pontas, a qual é utilizada pela Cisco no "SD Access".

Vamos estudar mais sobre essa forma de SDN em capítulos posteriores ainda nesse curso.

3.2 Separação dos Planes

Tudo o que os dispositivos de rede fazem pode ser dividido em funções ou planos, chamado em inglês de **Planes**.

Por exemplo, um switch L2 encaminha quadros com base em seu endereço MAC, porém para fazer isso sem causar loops ele roda um algoritmo chamado Spanning-Tree, que tem mais a ver com controle que encaminhamento propriamente dito.

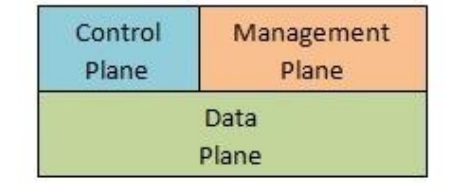
Além disso, o switch pode ser gerenciado via SNMP, por exemplo.

Cada função citada pode ser classificada entre controle, encaminhamento e gerenciamento.

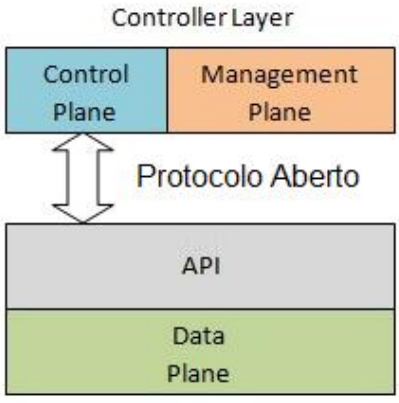
Essa é a base para entender e descrever como podemos criar redes programáveis, ou seja, separando as funções que existem nos dispositivos em três "planes": **data plane**, **control plane** e **management plane**.

Atualmente nos dispositivos tradicionais tudo está na caixa, por exemplo, um switch tem esses três planes e tudo é realizado dentro dele.

O SDN propõe a separação dessas funções, por isso é tão importante entendermos esses conceitos.



Dispositivo de Rede Atualmente



Arquitetura SDN

3.2.1 Data Plane

O Data Plane ou Forwarding Plane refere-se a camada responsável pelo encaminhamento dos quadros ou pacotes, dependendo da camada que o dispositivo atua (layer 2 ou layer 3).

Nos switches o encaminhamento é realizado em hardware utilizando ASICs (Application Specific Integrated Circuits), ou seja, circuitos integrados de aplicação específica.

Por exemplo, se lembrarmos como o roteador encaminha pacotes, quando um quadro é recebido em uma interface o roteador desencapsula o pacote IP, lê o endereço de destino do pacote, verifica se tem uma entrada na tabela de roteamento para a rede de destino do IP recebido, verifica que interface deve ser utilizada para encaminhamento, reescreve o pacote, encapsula o pacote conforme o quadro de camada 2 da interface de saída e o envia através dessa interface.

Portanto, em termos genéricos um roteador ou switch deve receber, processar e encaminhar a mensagem. Seguindo essa linha de raciocínio vamos listar as principais funções do data plane:

- Encapsular e desencapsular pacotes recebidos em um quadro (roteadores e switches Layer 3)
- Adicionar e remover cabeçalho 802.1Q recebidos em um trunk (roteadores e switches)
- Verifica se o MAC de destino existe na tabela de endereços MAC (switches layer 2)
- Verificar se o IP de destino tem entrada na tabela de roteamento (roteadores e switches layer 3)
- Criptografar dados e adicionar um no cabeçalho IP (processmento de VPNs)
- Alterar endereço IP de origem e destino (processamento do NAT)
- Descartar pacotes utilizando filtros (ACL e port security)

Todos esses itens fazem parte do data plane, sendo que essas ações são realizadas por mensagem recebida (quadro a quadro/ pacote a pacote).

3.2.2 Data Plane de um Switch

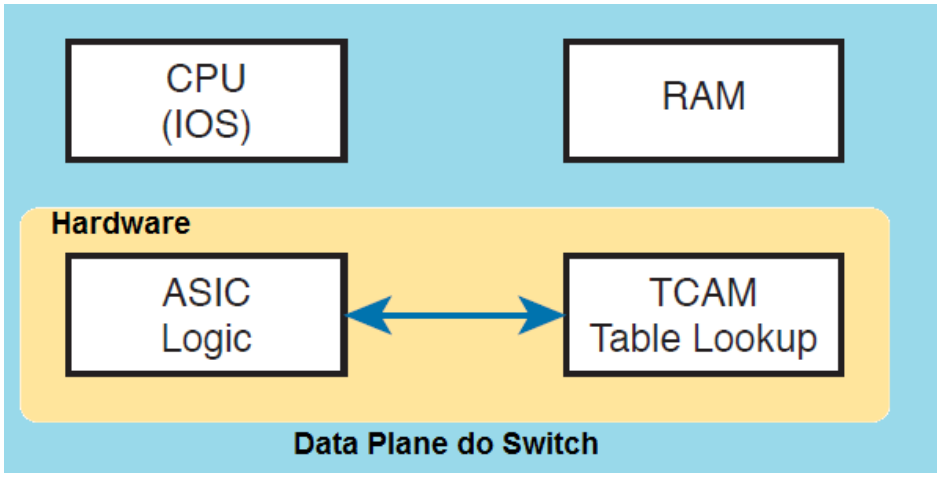
Citamos anteriormente que os switches têm seu data plane implementado em circuitos integrados, portanto, o switch faz o encaminhamento dos seus quadros através de hardware.

Por isso, lógica de encaminhamento dos switches não é realizada pela CPU em software, mas através de **Application Specific Integrated Circuit** (**ASIC**), que são chips construídos para fins específicos como processar quadros recebidos pelos switches.

Esses ASICs precisam verificar de maneira muito rápida os endereços MAC para tomar decisões de encaminhamento e para isso utilizam uma memória chamada ternary content-addressable memory ou TCAM.

Na realidade com essa memória chamada TCAM o ASIC não precisa fazer uma busca na tabela toda, ao invés disso ele verifica se determinados campos batem (match) e por isso não precisam utilizar algoritmos de busca, por exemplo, ele verifica se o MAC de destino bate com a entrada contida na TCAM e encaminha o quadro.

Esse processo é chamado de "Table Lookup".



Você vai estudar a seguir o control e management plane, os quais tem suas funções implementadas em software e rodam normalmente no Cisco IOS.

Já o data plane está implementado em hardware dentro dos ASICs, mas você sabe o porquê disso? Porque o encaminhamento em switches é realizado em hardware?

Por exemplo, vamos utilizar um switch de 24 portas, sendo que cada porta é fastethernet, ou seja, trabalha com a velocidade de 100 Mbps.

Essas portas também estão configuradas todas como full-duplex, transmitindo e recebendo simultaneamente.

Se utilizarmos quadros com 125 bytes, para termos 1000 bits de comprimento (125*8) e facilitar as contas, mais as características acima cada porta está recebendo 100.000 quadros por segundo ou frames per second (fps), que dá um total de 2.4 milhões de fps no total, ou seja, o data plane desse switch tem que estar preparado para processar e encaminhar dois milhões e quatrocentos mil quadros por segundo!

Por isso os switches utilizam hardware específico ao invés de software e CPU para processamento dos quadros, pois seria praticamente impossível a CPU tratar esse volume de informações e ainda fazer todas as outras funções dos demais planes (control e management) a um custo razoável.

3.2.3 Control Plane

O termo control plane refere-se a ações tomadas para controlar o data plane, ou seja, o control plane controla como o encaminhamento é realizado.

Maioria das ações tem a ver com a criação de tabelas que serão utilizadas pelo data plane para que o encaminhamento dos quadros e/ou pacotes possa ser realizado.

Por exemplo, o control plane cuida da criação de tabela de roteamento IP, tabela ARP, tabela de endereços MAC, tabela de vizinhos IPv6 (NDP) e assim por diante.

Portanto o control plane faz o controle do encaminhamento realizado pelo data plane adicionando, removendo ou alterando esses valores nas tabelas criadas.

Note que a essa altura do curso você já deve saber como funcionam esses protocolos do control plane, inclusive protocolos de roteamento RIPv2, OSPF e EIGRP. Portanto sem muita novidade.

A lista de protocolos da camada de controle incluem, por exemplo, protocolos de roteamento como OSPF, EIGRP, RIP e BGP, protocolos de resolução de endereços MAC como ARP (IPv4) e NDP (IPv6), aprendizado de endereços MACs por switches L2 (CAM Table) e spanning-tree (STP, RSTP e MSTP).

Pensando já na separação entre data e control plane imagine como seria um roteador "burro" sem a camada de controle?

Com certeza inútil, pois sem uma tabela de roteamento um roteador não teria sentido de existir, pois ele não conseguiria encaminhar pacotes.

Um switch L2 sem a função de aprendizagem de MACs, ou seja, sem construir uma tabela de endereços MAC ou CAM table poderia ainda encaminhar quadros fazendo o flooding, porém seu desempenho seria baixíssimo, pois ele acabaria sobrecarregando a rede LAN em comparação com um switch operando normalmente.

Portanto, para que o data plane funcione corretamente e de forma eficiente ele conta com as informações fornecidas pelo control plane.

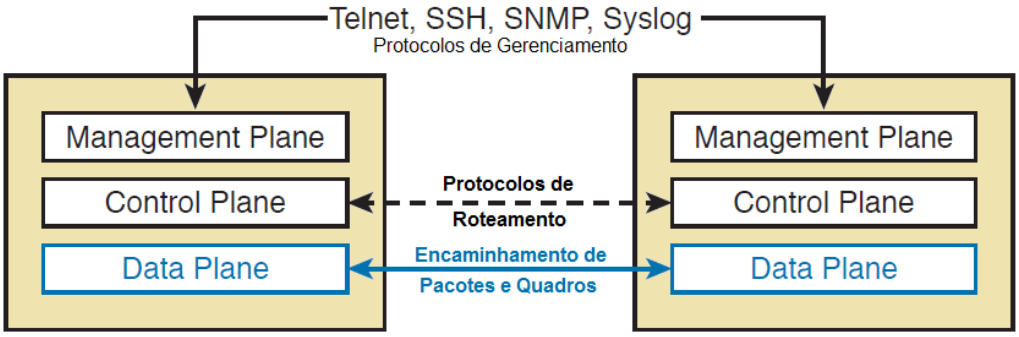
3.2.4 Management Plane

A camada de gerenciamento ou management plane diz tudo com o seu nome, pois é a camada que trata do gerenciamento dos dispositivos de rede.

Ela é representada por protocolos como Telnet, SSH, SNMP e Syslog.

Note que diferente do control plane, o management plane não afeta diretamente o funcionamento do data plane, ou seja, não afeta diretamente o encaminhamento dos quadros e pacotes rede, mas sim tratam de protocolos que os administradores de redes utilizam para gerenciar e configurar os dispositivos.

Na figura a seguir temos a representação de dois dispositivos de redes e suas camadas ou planes.



Dica: grave para o exame que Data Plane cuida do encaminhamento de quadros e pacotes, Control Plane dos protocolos de como o encaminhamento será realizado e Management Plane trata de como os dispositivos serão gerenciados.

3.3 Controladoras e Arquitetura Centralizada

Atualmente os dispositivos de rede tem uma arquitetura de controle (control plane) e gerenciamento (management plane) descentralizada, isso porque cada dispositivo ou caixa tem seu próprio controle e gerenciamento, simples assim.

Por exemplo, quando você precisa ativar um protocolo de roteamento como OSPF o que é preciso fazer?

Entrar em cada dispositivo e configurar o OSPF no control plane de cada um deles manualmente, seja através da CLI (com ou sem script) ou uma interface Web como o CCP, mas mesmo assim aplicar essa implementação (deploy) é uma tarefa feita dispositivo a dispositivo manualmente.

E após configurado cada dispositivo roda sua própria instância de OSPF localmente, portanto alterações mais uma vez precisam ser realizadas em cada dispositivo isoladamente.

Com o conceito de control plane os protocolos como STP, OSPF, EIGRP e demais da camada de controle poderiam ser centralizados e separados do hardware dos dispositivos, por exemplo, o control plane poderia estar rodando em um servidor ou dispositivo remoto de maneira centralizada ao invés de distribuído como é realizado na arquitetura convencional.

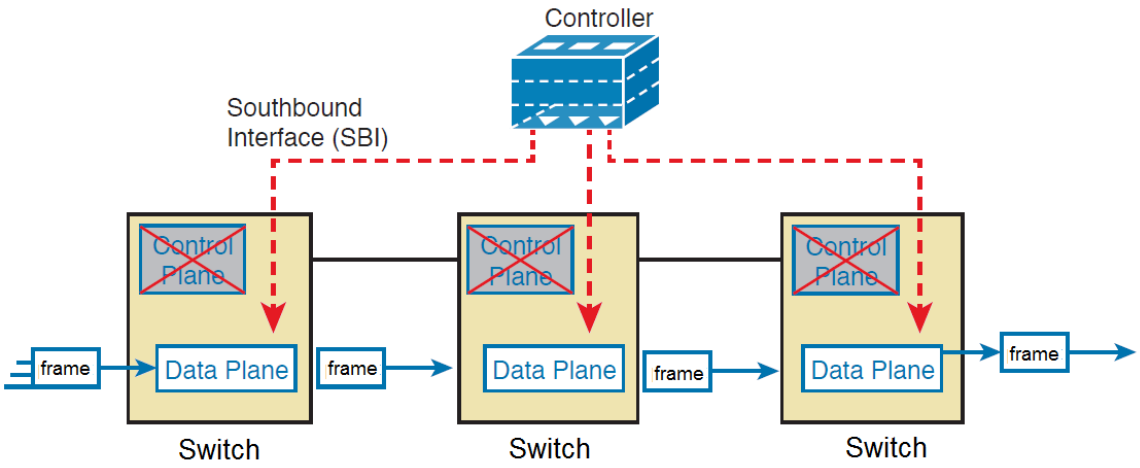
Sem discutir vantagens e desvantagens de arquiteturas centralizadas em relação a distribuídas, a centralização do control plane permitiria, por exemplo, que alterações em vários dispositivos fossem mais simples, pois as informações de controle estariam centralizadas em um único ponto, facilitando a distribuição dessas novas regras para todos os dispositivos.

As redes programáveis e o SDN normalmente utilizam-se desse conceito de arquitetura de rede centralizada (centralized architecture), com a separação e centralização do control plane em uma controladora ou "controller".

Essa controladora (controller ou SDN controller) tem, portanto, a função de centralizar o controle dos dispositivos de rede, sendo que o nível de controle pode variar conforme a implementação e/ou fabricante, indo de um controle completo de todas as funções do control plane, controle parcial ou até somente estar ciente do trabalho dos control planes distribuídos em dispositivos tradicionais.

Por exemplo, você poderia ter uma rede com uma controladora SDN centralizada e seus switches do data Center controladas por essa controladora, portanto os switches não teriam a inteligência neles.

Eles não decidiriam como aprender MACs ou encaminhar quadros, quem faria essas tarefas seria a controladora que apenas repassaria uma tabela de encaminhamento para os switches, os quais continuam tendo o data plane distribuído em cada dispositivo.



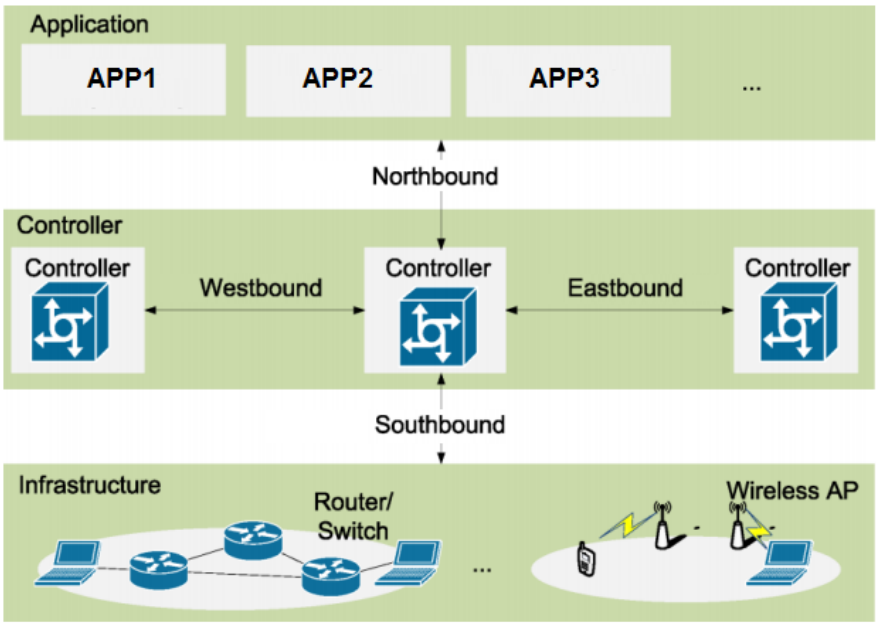
Note que a comunicação entre o controller SDN e os switches é realizada através de uma interface chamada Southbound Internet ou SBI, a qual utiliza protocolos específicos para comunicação, mas vamos estudar mais sobre esses detalhes no próximo tópico.

Como já citado anteriormente, existe também outra forma de Software Defined Network onde os dispositivos das pontas (routers e switches) mantém a sua "inteligência", vamos estudar mais esse modelo no capítulo sobre SD Access.

3.3.1 Interfaces Southbound e Northbound

Quando falamos em interfaces na realidade não são interfaces físicas, mas sim interfaces de software ou APIs (Application Programming Interface) que os controllers utilizam para fazer a comunicação com os dispositivos de rede e demais dispositivos da arquitetura SDN.

O SDN define quatro tipos de interfaces: Northbound, Southbound, Westbound e Eastbound (norte, sul, leste e oeste), sendo que para o exame apenas as duas primeiras são importantes, pois as duas últimas tratam da comunicação entre controllers e não estão ainda bem definidas.



A arquitetura SDN normalmente é dividida em três camadas: Infraestrutura (infrastructure), Controller (camada de controle onde estão as controladoras) e Aplicação (application onde estão os aplicativos).

Como você pode notar na figura anterior a interface southbound ou SBI faz a comunicação entre o controller e os dispositivos da infraestrutura de redes como switches e roteadores.

A SBI é uma interface entre dois programas de computador (um contido no controller e outro no dispositivo de rede) que permite a comunicação entre eles, sendo que seu objetivo final é permitir que as tabelas de encaminhamento possam ser controladas nos dispositivos.

Essa comunicação pode ser realizada através de protocolos ou APIs.

Os APIs que são utilizados para que uma aplicação ou programa troque dados com outra aplicação ou programa, ou seja, um API é uma interface de aplicação (programa ou aplicativo) que permite que ele troque informações com outras aplicações.

A diferença para o protocolo é que para ele existe uma documentação, normalmente um padrão aberto ou proprietário, por exemplo, uma RFC que normatiza o protocolo, enquanto para a API temos funções, variáveis e estruturas de programação.

Existem vários tipos de SBI que podem ser utilizados, sendo que os três mais importantes para o exame são: OpenFlow (padrão da ONF - https://www.opennetworking.org), OpFlex (padrão Cisco utilizado no ACI), NetConf e CLI/SNMP (padrão Cisco utilizado no APIC-EM e DNA Center).

A interface Northbound ou NBI conecta o controller com outros programas ou aplicações permitindo que a rede seja programada, daí o nome rede definida por software ou SDN, pois agora podemos alterar ou controlar o comportamento dos dispositivos de rede através de aplicativos.

Os aplicativos ou programas externos podem obter informações a partir da NBI do controller e programar a partir daí o data plane utilizando as interfaces SBIs que se conectam com os dispositivos de rede.

Essa comunicação entre a aplicação e o controller via interface northbound pode ser realizada, por exemplo, através de APIs em Java ou então utilizando o REST (Representational State Transfer).

Vamos estudar mais sobre APIs e REST posteriormente ainda nesse curso.

3.3.2 Open SDN, OpenFlow e OpenDaylight Controller

O protocolo OpenFlow é a base de uma rede definida por software (SDN) aberta e baseada em padrões abertos, essa rede SDN aberta é chamada também de Open SDN.

O desenvolvimento do OpenFlow começou em 2007, e é uma colaboração entre o mundo acadêmico e o mundo comercial.

Conduzido originalmente pela Universidade de Stanford e pela Universidade da Califórnia em Berkeley, o padrão agora está sendo definido pela Open Networking Foundation (ONF - https://www.opennetworking.org).

O modelo definido pela ONF coloca o OpenFlow como SBI dos controllers SDN, ou seja, o padrão de comunicação para a interface southbound é o openflow, o qual utiliza uma comunicação via IP para comunicação entre o controller e os dispositivos de rede.

O OpenFlow também define a ideia de capacidades para os switches, as quais são baseadas nos ASICs e TCAMs encontradas nos switches atualmente. Essa padronização faz uma abstração do switch, a separação entre data e control plane (switch abstraction).

No OpenFlow o switch pode ser Layer 2, Layer 3, Layer 2/3 ou algo diferente e mais flexível do que temos tradicionalmente, pois agora podemos programar como os fluxos que passam pelos switches serão tratados.

No modelo chamado Open SDN temos um controller centralizado que pode também se comunicar com aplicações através da interface northbound (NBI) como estudamos anteriormente. Os dispositivos de rede devem suportar o openflow, pois o controller vai utilizá-lo como SBI.

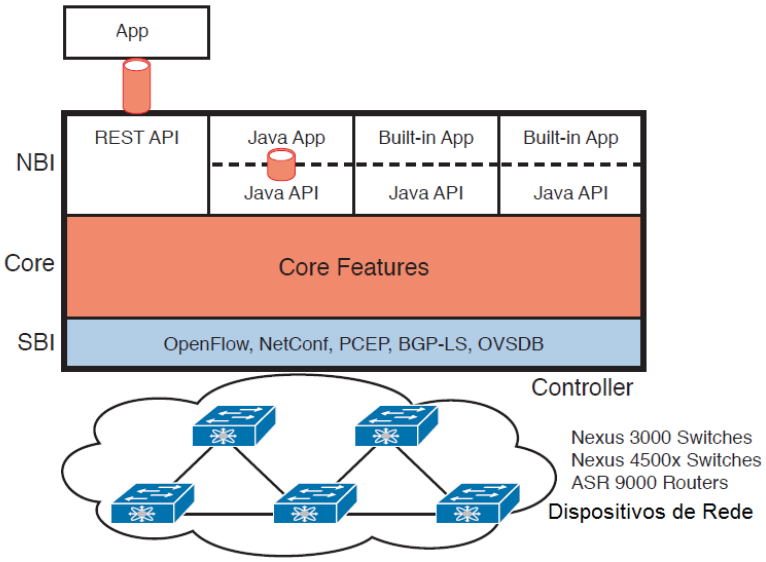
Após o início do SDN algumas empresas começaram a trabalhar em conjunto para desenvolver um controller SDN aberto (open source SDN controller) e mais ou menos em 2010 nasceu o OpenDaylight SDN controller (https://www.opendaylight.org).

O OpenDaylight ou ODL foi um projeto iniciado pelo Linux Foundation e suporta uma variedade de SBIs, tais como OpenFlow, NetConf, PCEP, BGP-LS e OVSDB.

Devido ao projeto do ODL ser composto por diversos fabricantes ele suporta muitas SBIs além do OpenFlow.

Você pode ver mais sobre o projeto em “OpenDaylight.org”.

Abaixo segue o modelo do controller ODL. Tanto o App externo como o Java App pode ser desenvolvido por qualquer fabricante, desenvolvedor ou até você mesmo!



4 Software-Defined Access e sua Arquitetura

4.1 Introdução

O modelo tradicional do SDN normalmente é chamado de Imperativo (Imperative), pois a controladora vai dizer TUDO o que os dispositivos de rede devem fazer para o encaminhamento de um pacote na rede, por exemplo.

Os dispositivos de rede não participam do processo de decisão, tudo é feito pela controladora que envia os passos que esses dispositivos devem tomar para a decisão de encaminhamento.

O que acontece se a controladora cai nesse modelo?

Simples, o dispositivo final fica sem saber o que fazer... você terá uma indisponibilidade na rede!

No modelo escolhido pela Cisco e vários outros fabricantes para aplicar o conceito de rede programável foi o "Declarativo" ou "Declarative".

Nesse modelo a rede continua tendo uma controladora, porém os dispositivos de rede não são mais "zumbis" totalmente comandados por ela.

Os roteadores e switches continuam tendo a capacidade de fazer todas as funções de rede, por isso mesmo que a controladora fique indisponível por qualquer motivo, a rede continuará funcionando.

Talvez novas configurações ou alterações na rede não possam ser implementadas sem a controladora, mas o fluxo de informações continuará seguindo normalmente mesmo sem o "cérebro" da rede.

Por isso, além do termo SD Access ou SDA, você vai ouvir também o termo "Intent Based Network" ou IBN, pois via controladora o administrador de rede vai colocar sua "intenção" e isso será traduzido em configurações que serão replicadas para os dispositivos de rede.

Isso é realizado através de alguns conceitos que vamos estudar a seguir:

- Os problemas de uma Rede Tradicional
- Underlay
- Overlay
- Fabric
- VXLAN
- LISP
- Segurança e QoS
- Anycast Gateway

Vamos lá estudar os conceitos e depois "montar o quebra cabeças"!

4.2 Redes Tradicionais e Seus Problemas

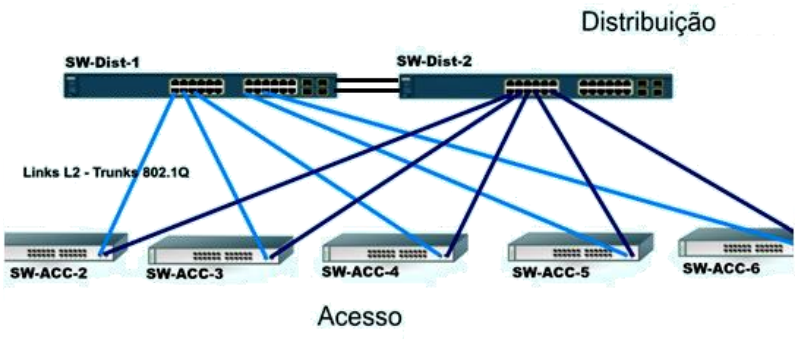
Falamos até o momento das dificuldades operacionais das redes tradicionais, porém elas têm outros desafios e dificuldades herdadas a muito tempo atrás e que nos "assombra" até os dias de hoje.

As redes tradicionais têm em sua estrutura fundamental um protocolo antigo que é o Ethernet.

A base do Ethernet seja na versão que for (fast, giga, 10giga) é a mesma e traz consigo um problema clássico nas redes que são os "Broadcasts".

Juntamente com os broadcasts temos os protocolos para minimizar seus efeitos que são da família do Spanning Tree, os quais trazem para as redes tradicionais de qualquer porte uma complexidade natural quando temos que aumentar ou escalar a rede.

Para minimizar esses efeitos outras tecnologias foram criadas, aumentando mais ainda a complexidade da rede. Por exemplo, empilhamento de switches, link aggregation, VSL e VPC.



Outras coisas complexas em redes tradicionais são políticas de QoS e de Segurança.

A administração dessas políticas normalmente envolve configurações extensas e complexas, realizadas dispositivo a dispositivo.

Em uma LAN, normalmente essas políticas de segurança e QoS são aplicadas em VLANs, as quais são as diversas sub-redes da LAN, porém se quisermos que essa política seja a mesma para vários dispositivos precisamos colocá-los em uma mesma VLAN.

Isso significa que muitas vezes para manter a estrutura precisamos estender os domínios de broadcasts e cruzar as distribuições em camada-2, aumentando mais ainda os efeitos dos broadcasts na rede.

Você já conhece bem essas dificuldades técnicas e de configuração, pois se está seguindo a trilha da certificação CCNA, desde o primeiro capítulo até agora administrou essas dificuldades.

Vários protocolos, cada um com seus comandos, administração descentralizada, ou seja, configurando um a um cada dispositivo... e por aí vai...

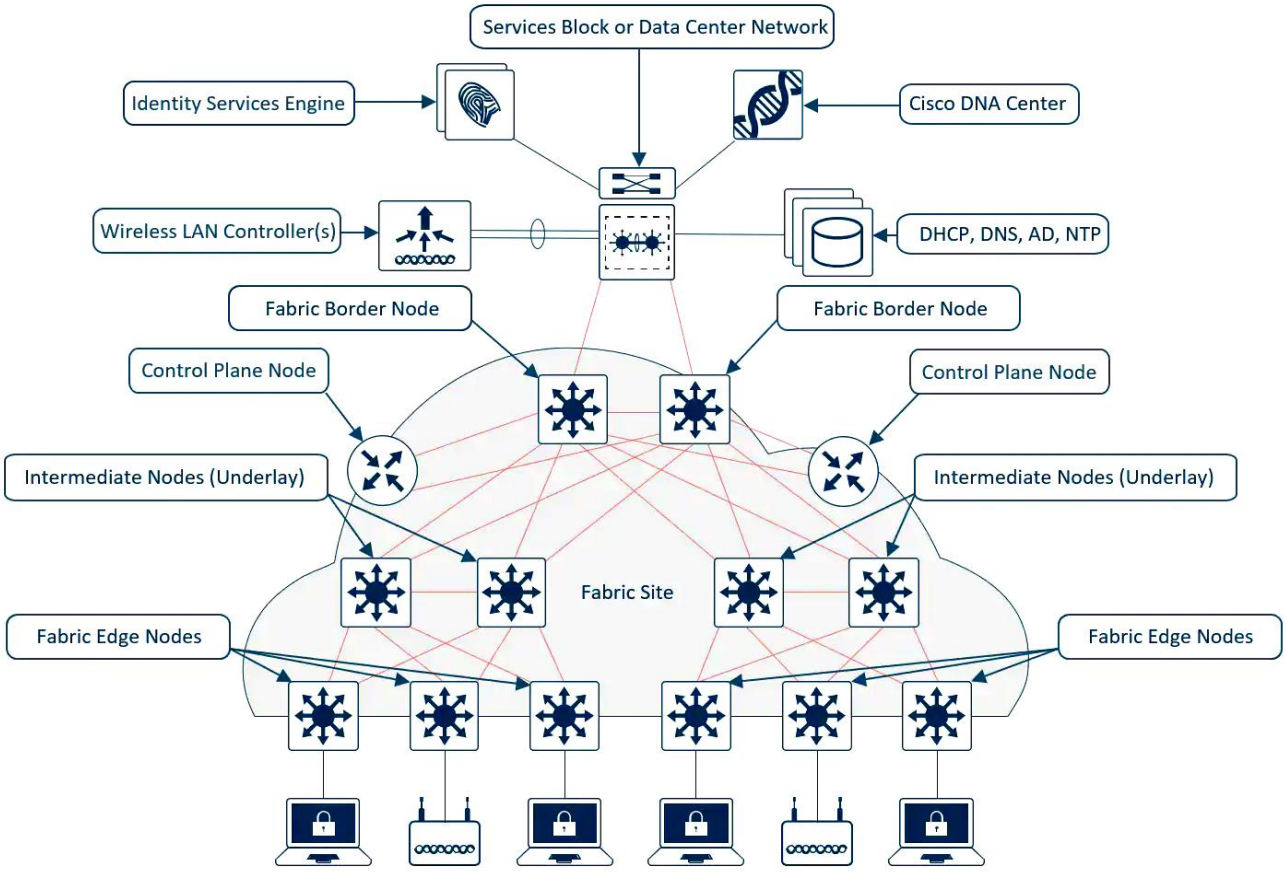
Guarde isso na sua mente e no final, quando formos juntar as peças do SD Access vamos comparar essa nova filosofia com as redes tradicionais!

4.3 Topologia e Elementos do Cisco SDA

O SDA está dividido em quatro camadas:

1. **Physical Layer**: essa camada trata dos dispositivos como switches, roteadores, access points, modelos e funções de cada dispositivo dentro da arquitetura.
2. **Network Layer**: camada composta pelo Underlay e Overlay.
3. **Controller Layer**: são basicamente o DNA Center e o ISE (Identity Services Engine), que em suma é a interface onde vamos inserir configurações e buscar informações.
4. **Management Layer**: trata da interface e gerenciamento de toda topologia SDA, tanto via interface gráfica via Web (GUI) como através de APIs e REST.

A topologia que a Cisco utiliza em seu acesso definido por software ou SDA (Software Defined Access) está representada na figura a seguir.



- **Fabric**: é a topologia lógica que o Overlay vai criar utilizando os túneis ou VXLAN. Podemos dizer que é a rede do SDA de forma genérica.
- **Fabric Edge Node**: switches que tem a função de conectar os endpoints e Access Points ao Fabric (rede SDA). Servem como Anycast Gateway para os endpoints.
- **Control-Plane Node**: normalmente um par de roteadores responsáveis por mapear a que switch de borda (Edge Node) cada endpoint pertence utilizando o LISP (Locator ID Separation Protocol). São responsáveis pelo "roteamento" e "fechamento" de uma VXLAN entre endpoints.
- **Fabric Wireless Controller**: WLC específica para o SDA que tem a função de configurar e controlar os APs contidos no fabric. Pode estar locado fora do Fabric, por exemplo, em um Data Center.
- **Fabric Border Nodes**: tem a função de conectar redes externas, tais como Data Center e Internet ao Fabric.

- **Intermediate Nodes (Underlay)**: são dispositivos que tem a função de fazer o transporte de pacotes entre Edge Nodes, Control Plane Nodes e Fabric Border Nodes. São encarados como dispositivos de "transporte" e não precisam ser necessariamente dispositivos Cisco, pois são transparentes para a comunicação final entre os endpoints.
- **Cisco DNA Center**: é a própria controladora SDN da Cisco que fornece um ponto único de contato para configuração de toda a rede.
- **Identity Services Engine (ISE)**: trata da identificação dos usuários, controle de acesso e definições dinâmicas de políticas para os endpoints.

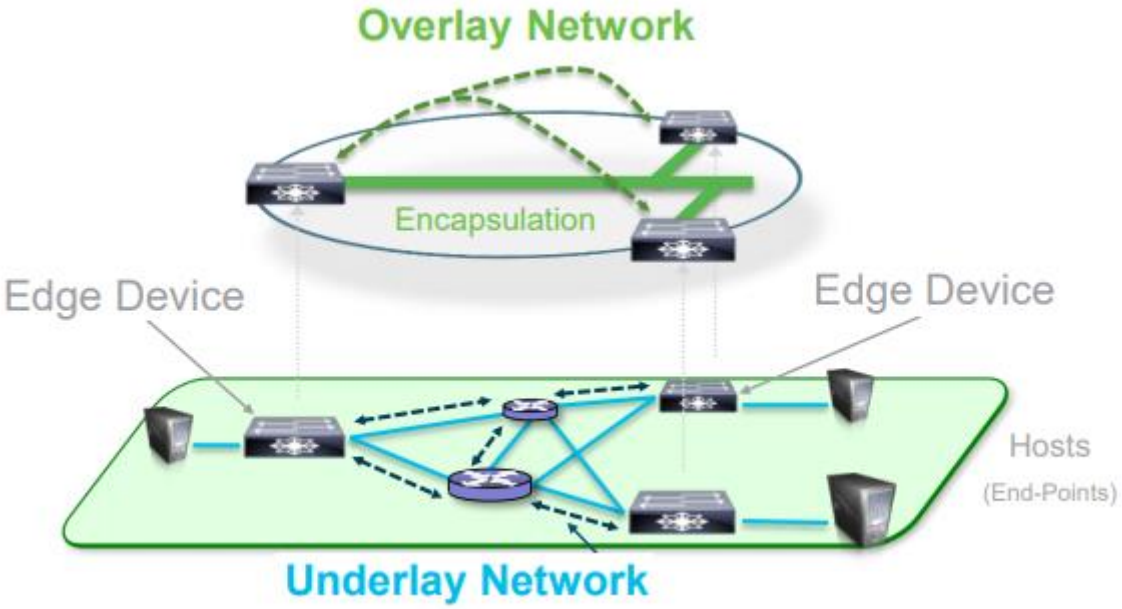
Note que a estrutura SDA que estamos mostrando acima faz parte da solução fornecida pelo DNA Center, por isso alguns dispositivos obrigatoriamente devem ser suportados pela controladora.

Dispositivos como Edge Nodes, Fabric WLC, Access Points, Control Plane Node e Border Nodes devem ser compatíveis com a versão de DNA Center a ser utilizada.

Apenas os Intermediate Nodes e servidores que prestam serviços como AD, DSN, DHCP, NTP, etc., conforme mostrado na topologia, não necessariamente precisam ser Cisco e nem compatíveis com a controladora utilizada na solução SDA chamada DNA Center.

4.4 Underlay

O Underlay da arquitetura SD Access é toda infraestrutura física e lógica que vai servir como base para o tráfego dos pacotes IP dos endpoints através das VXLANs (túneis de camada-2 que vamos estudar em breve) que serão criadas entre os Edge Nodes ou Edge Device.



São roteadores, switches, cabeamento e tudo mais que sirva para o transporte de informações que serão passadas via Overlay, conforme representado na figura anterior.

A principal função do Underlay é prover a conectividade fim a fim em camada-3 que vai possibilitar a criação dos túneis que serão a base do Overlay.

Portanto, o Underlay pode ser uma infraestrutura legada, ou seja, já existente tanto em camada-2 quanto em camada-3 ou então uma estrutura nova, criada do zero para a implantação do SDA.

Com uma infraestrutura existente você precisará uniformizar apenas os dispositivos que precisam ser obrigatoriamente da arquitetura SDA da Cisco que são:

- Edge nodes
- Access Points
- Fabric WLC
- Control-plane Nodes
- Border Nodes

Os demais dispositivos de transporte ou Intermediate Nodes que fazem parte do Underlay podem permanecer sem alteração desde que eles garantam a conectividade IP entre os dispositivos citados anteriormente.

Essa topologia pode ser tanto L2 como L3, porém uma topologia L2 continua trazendo para a rede física (underlay) todos aqueles problemas conhecidos com relação ao Ethernet, Broadcast, STP, políticas de QoS, segurança etc. Basicamente o Underlay continua sendo administrado da forma convencional.

Recomenda-se não utilizar o DNA Center para administrar os dispositivos do Underlay, pois como eles não são compatíveis com a solução isso pode causar danos à rede e até mesmo disrupções.

Existe no site da própria Cisco uma lista de compatibilidade de dispositivos e versão de sistema operacional (Cisco IOS ou IOS-XE) entre a versão de DNA Center e os equipamentos de rede.

Recomenda-se fazer uma análise bem detalhada para não haver transtornos no momento da implantação do SDA em redes pré-existentes.

No caso de uma infraestrutura nova (greenfield design ou "design do zero") a recomendação é que ela já seja toda criada utilizando links L3 entre os dispositivos do Underlay, assim como escolher TODOS os dispositivos suportados pelo DNA Center.

Seguem algumas diretrizes e constatações sobre o underlay em uma nova implantação:

- Utilizar apenas dispositivos compatíveis com o SDA da Cisco e DNA Center.
- Utilizar apenas switches Layer 3.
- Utilizar o IS-IS como protocolo de roteamento.
- Todos os links entre switches (sejam eles portas únicas ou EtherChannels) devem ser portas roteadas (L3).
- Com essa configuração o STP/RSTP/MSTP não serão mais necessários no Underlay e o protocolo de roteamento fará a escolha do melhor caminho com base na tabela de roteamento.
- Agora a camada de acesso com seus switches fará o papel do default gateway para os endpoints, lembrando que em redes convencionais o gateway normalmente está nos switches de distribuição. Isso é feito através de Anycast gateway (estudaremos mais tarde).
- Além disso, essa nova arquitetura elimina o uso de protocolos de proteção de primeiro salto (FHRP), por exemplo, HSRP e VRRP.

Além disso, a Cisco disponibiliza o recurso chamado "Zero-touch provisioning" ou provisionamento automático, onde o dispositivo vai alocar um IP designado no pool do DNA Center e buscar sua configuração automaticamente, sem nenhuma intervenção humana.

Dessa maneira toda a rede será configurada e administrada pelo DNA Center.

Um detalhe interessante sobre o Underlay é que independente da solução ser uma rede pré-existente ou nova será necessário um protocolo de roteamento interno ou IGP para fazer com que os switches das pontas ou Edge Nodes consigam se comunicar entre si.

Porém, não será necessário anunciar as redes dos endpoints, pois essas sub-redes serão administradas via LISP utilizando os roteadores chamados Control-plane Nodes.

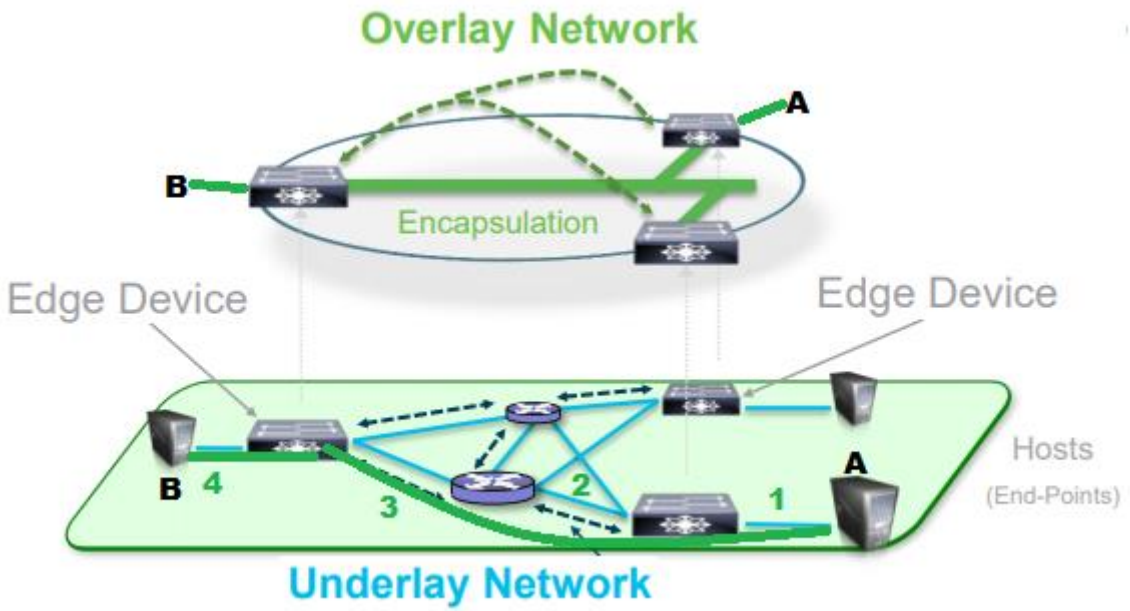
4.5 Fabric e Overlay

O Overlay é a abstração de rede física em uma rede lógica, chamada também de "Fabric", a qual basicamente é formada através de túneis ou VXLANs.

A diferença do Overlay para o Underlay é que ele pode seguir a topologia que você configurar, sem a necessidade de conexões físicas, pois tudo é realizado em um Plano ou Plane superior que esconde as complexidades da rede dos usuários finais.

Dentro do SDA o Overlay pode ser dividido em três Planes:

- **SDA Data Plane**: que é o tráfego entre os endpoints realizados pelas VXLANs (túneis criados entre os Edge Nodes).
- **SDA Control Plane**: responsável pelo mapeamento e roteamento dos endpoints utilizando o LISP.
- **SDA Policy Plane**: responsável por toda a Segurança (utilizando CTS ou Cisco Trust Sec) e QoS do tráfego entre os endpoints.



Note na figura anterior o exemplo onde o Host A deseja se comunicar com o Host B.

Analisando o Underlay, o pacote do Host A (1) sai de um switch (que é um Edge Node), passa por um roteador interno (2) do Underlay e finalmente chega ao segundo switch (3 | Edge Node de Destino) onde é encaminhado ao Host B (4).

Mas para os Hosts A e B esse pacote nem saiu da mesma LAN, pois o Overlay cria um túnel utilizando uma VXLAN, o qual simula uma conexão local como se ambos os servidores estivessem na mesma LAN ou VLAN.

Na realidade o pacote do Host A é tunelado (colocado dentro de uma VXLAN) no primeiro switch e quando chega no switch de destino simplesmente é desencapsulado (removido do túnel) e enviado para o Host B.

Mas você deve estar se perguntando: Como o switch do Host A consegue saber onde está o Host B para criar o túnel e enviar as informações se as redes dos endpoints não são anunciadas no Underlay?

A resposta está no LISP e nos Control-Plane Nodes!

Na verdade, o Overlay utiliza o LISP (que vamos estudar melhor posteriormente) para fazer a descoberta e localização de endpoints, função essa que está no Control Plane do SDA.

Resumidamente, a cada endpoint novo na rede que um Edge Node (switch de acesso) descobre é também enviada uma mensagem para os Control-Plane Nodes informando os dados desse endpoint.

Dessa maneira, toda vez que dois hosts precisam se comunicar os Edge Nodes solicitam aos Control-Plane Nodes a localização desse endpoint de destino para criação do túnel ou VXLAN entre os dispositivos.

Portanto, cada endpoint no Overlay terá seu identificador dentro do Fabric, chamado EID ou Endpoint Identifier.

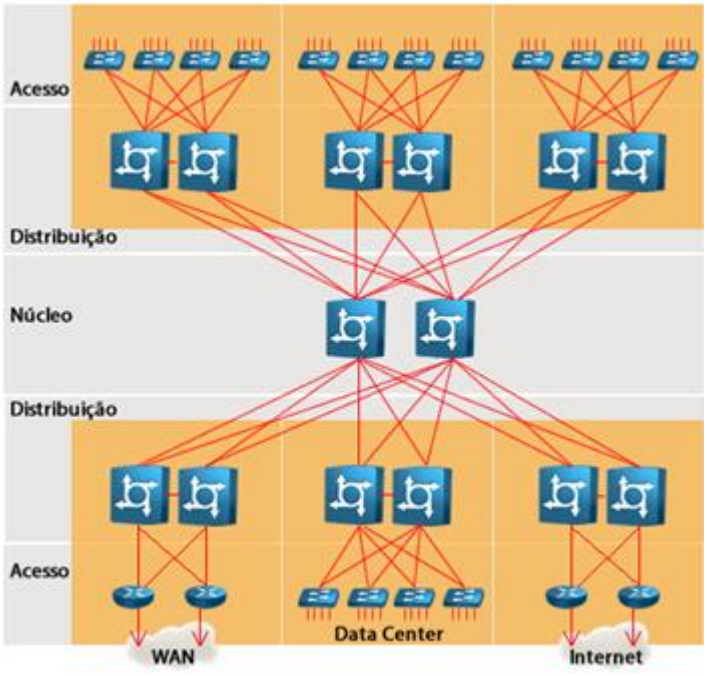
Assim como cada Edge Node terá também um identificador chamado Routing Locators (RLOCs), assim os Control-Plane Nodes sabem como encontrar onde está um determinado Endepoint e informar com qual Edge Node de destino uma VXLAN (Túnel) deve ser fechado para que a comunicação entre os endpoints possa ser estabalecida.

4.6 LISP: Locator ID Separation Protocol

Como já citado anteriormente o **LISP** ou "**Locator ID Separation Protocol**" será responsável no SDA por fazer com que os Edge Nodes consigam determinar onde um Host ou Endpoint de destino está situado dentro do Fabric.

Em uma rede tradicional quem faz isso são os protocolos de roteamento, mais especificamente um IGP como o OSPF ou EIGRP quando estamos na rede Interna de uma empresa.

Com os protocolos de roteamento TODOS OS DISPOSITIVOS L3 necessitam conhecer como alcançar TODAS as sub-redes onde temos endpoints, sejam LANs ou VLANs, lembra disso certo?



Em redes de pequeno porte isso não é um problema, porém quando falamos de um Campus com centenas de dispositivos e dezenas de mil sub-redes o cenário começa a mudar, pois os dispositivos da Distribuição e Core precisarão conhecer essas redes e normalmente não podem ser "caixas pequenas", pois elas precisam ter "poder de processamento e memória" para tudo isso, concorda?

Agora imagine se os dispositivos de distribuição e core de uma rede tradicional não precisassem conhecer as redes dos usuários, ou seja, as sub-redes das LANs e VLANs onde estão computadores e servidores da empresa.
Se pegássemos a topologia da figura anterior, teríamos apenas algumas poucas sub-redes entre os switches, pois nesse desenho temos apenas 30 switches e 4 roteadores nela certo?

Olhando assim, esses switches de Core que normalmente precisam ser "parrudos" nem precisariam de tanto poder assim...

No SDA com o LISP é isso que acontece, pois quem precisa saber as sub-redes e VLANs dos usuários finais, seja o tipo de endpoint que for, são apenas os Edge Nodes.

O Underlay que conecta esses Edge Nodes não precisa mais conhecer essas sub-redes, pois elas ficam escondidas dentro do túnel que é formado pelas VXLANs, as quais transportam os pacotes dos Endpoints.

Portanto, as sub-redes dos Endpoints estão "escondidas" dentro do túnel criado com as VXLANs.

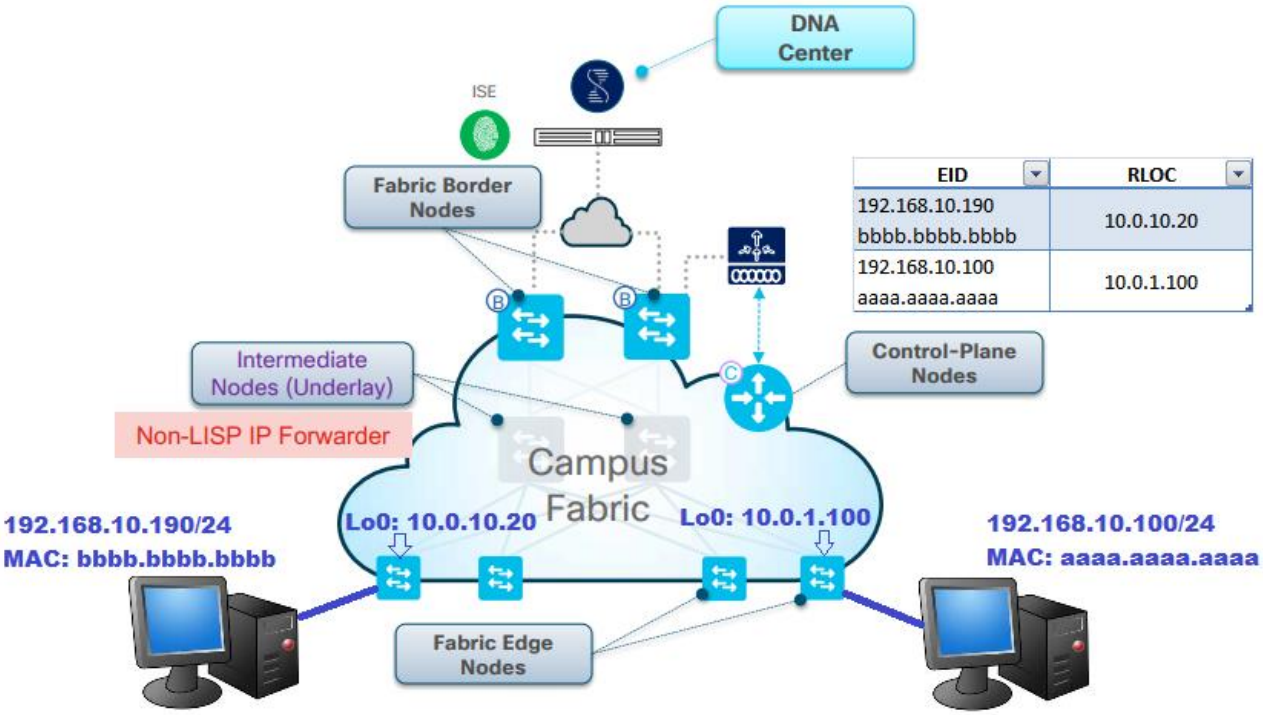
O LISP tem a função de mapear os endpoints com um identificador chamado EID ou Endpoint Identifier, assim como mapear a que Edge Node esse endpoint está vinculado.

Os Edge Nodes terão também seu identificador chamado Routing Locators (RLOCs).

Isso tudo ficará armazenado nos Control-Plane Nodes, pois dessa maneira eles conseguem saber como encontrar onde está um determinado Endepoint e informar com qual Edge Node de destino uma VXLAN (Túnel) deve ser fechado para que a comunicação entre os endpoints possa ser estabelecida.

Os Control-Plane Nodes, portanto, servem como "LISP Map-Servers", ou seja, fazem o mapeamento entre os EIDs (endpoints) e RLOCs (Edge Devices), permitindo que os diversos endpoints se comuniquem através do Fabric.

Veja exemplo na figura a seguir.



Note que o endpoint com endereço IP 192.168.10.100 e MAC aaaa.aaaa.aaaa está conectado ao Switch (que é um edge node) que tem endereço de gerenciamento configurado na Loopback 0 como 10.0.1.100.

Portanto, esse switch enviará para os control-plane nodes essa informação, a qual será armazenada vinculando o EID do endpoint (192.168.1.100 e seu MAC) ao RLOC do Edge Node (10.0.1.100).

Caso o computador com IP 192.168.10.190 queira se comunicar com o 192.168.10.100, o seu Edge Node fará uma consulta ao Control-plane node, que é o LISP Server para os Edge Nodes, o qual responderá com o RLOC do switch onde o IP 192.168.10.100 está conectado.

Com isso o switch de origem pode montar o túnel VXLAN que fará a comunicação no Data Plane do Fabric SDA entre os dois endpoints.

Você pode estar se perguntando agora: E as redes externas ou a Internet que está fora do LISP e do Fabric do SDA?

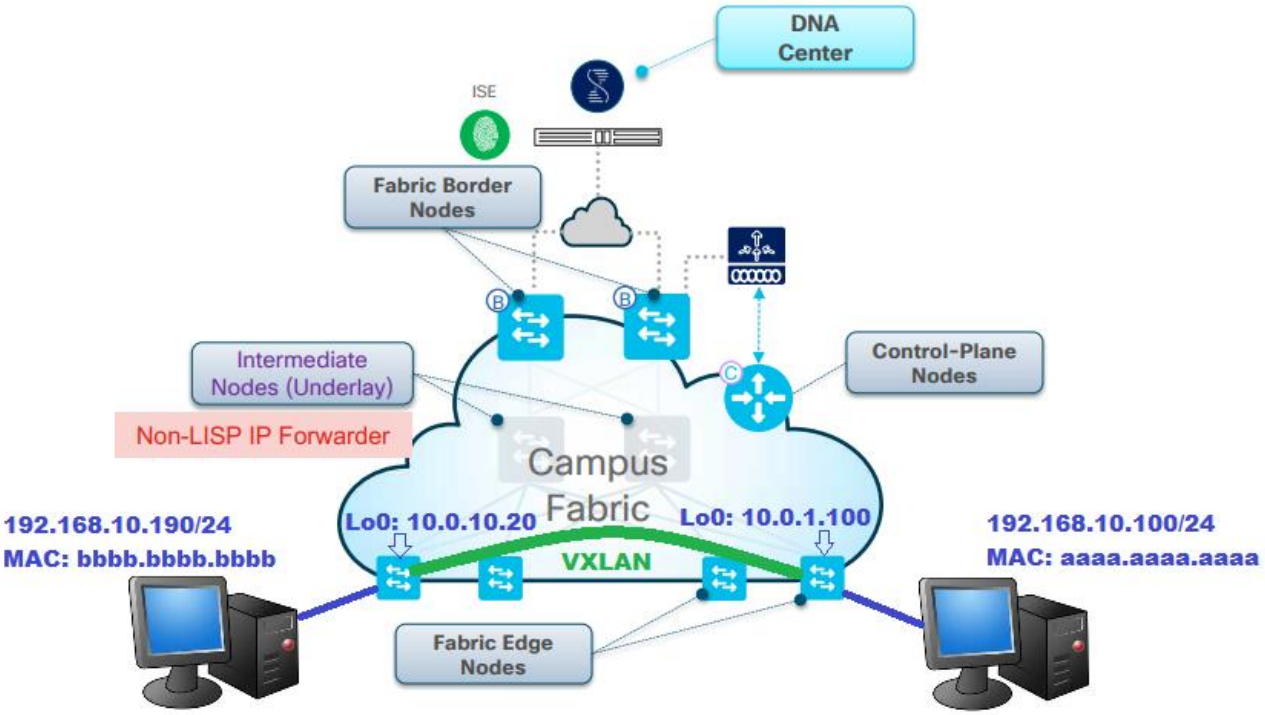
Aí os Fabric Border Nodes cuidam de traduzir o mundo externo para dentro do Fabric e vice versa. Neles são mapeadas as redes externas e juntamente com os Control-Plane Nodes toda e qualquer requisição de encaminhamento será completada.

4.6.1 Endereçamento do Underlay versus Overlay

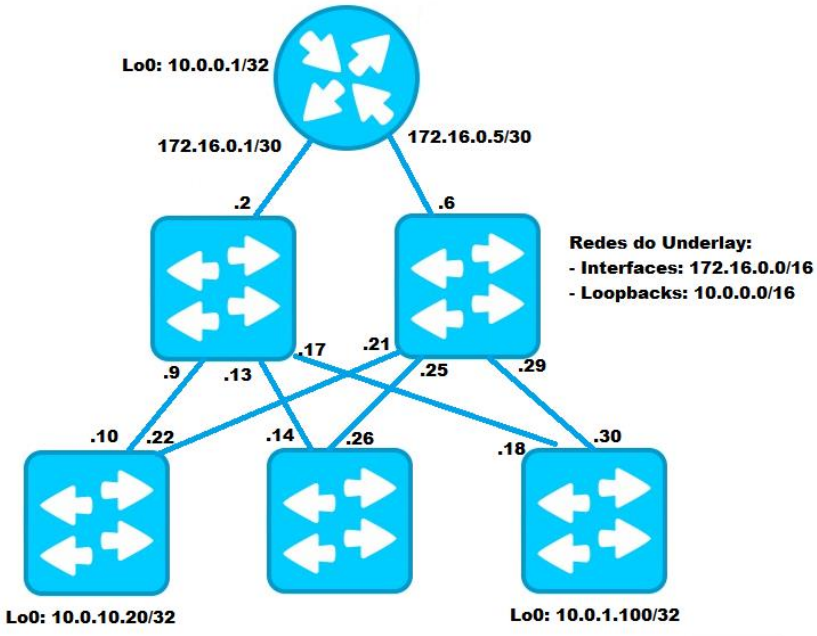
Lembre-se que o endereçamento do Underlay é um outro espaço de endereçamento do Overlay, por isso mesmo não será necessário o dispositivo do Underlay conhecer os endereços dos Endpoints.

Vamos usar o exemplo já feito anteriormente para reforçar o conceito.

Portanto, se o PC 192.168.10.100 quer enviar pacotes para o PC 192.168.10.190, seu Edge Node solicita a localização do IP 192.168.10.190 ao control-plane e cria uma VXLAN para o RLOC aprendido que é 10.0.10.12. Portanto, o IP de origem do túnel será 10.0.1.100 e de destino 10.0.10.20.

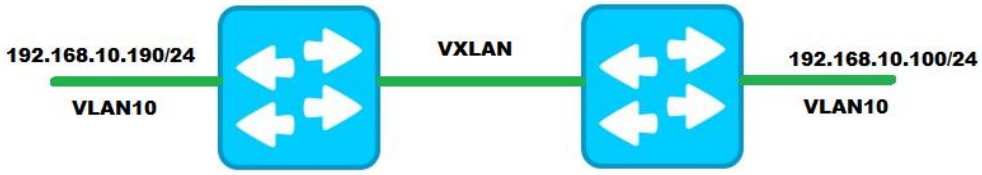


Veja exemplo abaixo onde o Underlay utiliza a rede 172.16.0.0/16 e 10.0.0.0/16 para seu endereçamento e o Underlay a rede 192.168.0.0/16 para endereçar as VLANs dos Endpoints.



Note na figura anterior que o Underlay tem uma tabela de roteamento pequena e tem que garantir que todos os Edge Nodes e demais dispositivos do Fabric do SDA se comuniquem.

Já na figura abaixo, a visão dos endpoints é que eles estão conectados a dois switches, em camada-2 a uma mesma VLAN, por exemplo, a VLAN 10. Quem cria essa visão no encaminhamento ou data plane é o túnel VXLAN criado entre os Edge Nodes.



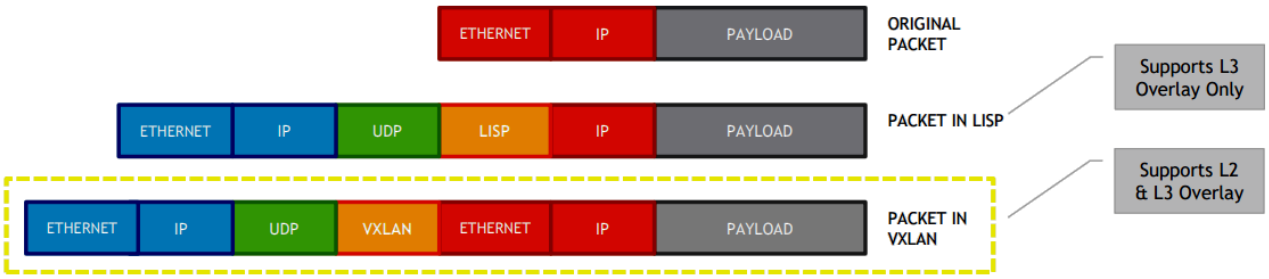
Essa é a visão da rede criada pelo Overlay ou Fabric do SDA.

4.7 VXLAN: Virtual Extensible LAN

O próprio LISP já possui a possibilidade de tunelamento, porém ele suporta apenas um Overlay em L3, ou seja, Endpoints comunicando apenas via IP.

Mas o que mais interessa em uma LAN e uma infraestrutura de um Campus é a possibilidade de comunicação tanto em camada-2 como camada-3, por isso mesmo o LISP é utilizado para determinar a localização dos Endpoints (control plane) e o Data Plane, ou seja, a comunicação de dados do Frabric do SDA é feita utilizando uma VXLAN.

Portanto, como já estudamos anteriormente, quando um endpoint dentro do Fabric (ou simplesmente Rede) do SDA precisa enviar um frame ou quadro para outro endpoint situado em outro Edge Node da rede, o switch de entrada (ou Edge Node de entrada) encapsula o quadro e envia essa informação utilizando um túnel VXLAN até o switch ou Edge Node de destino.



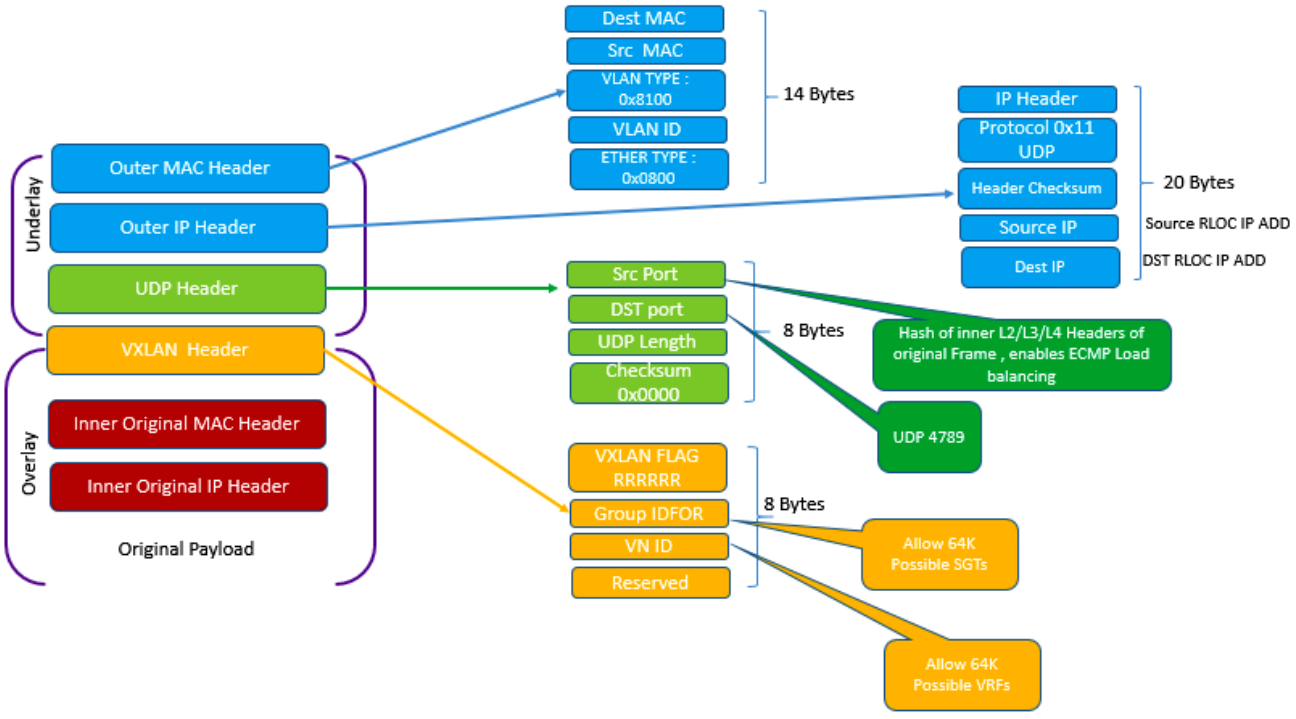
Note que o quadro original é encapsulado dentro de um novo pacote IP, o qual tem como origem o RLOC do Edge node de origem (normalmente o endereço IP de uma interface loopback de gerenciamento) e o destino terá o RLOC (ou IP de gerenciamento) do Edge node de destino.

Já os dados do endpoint estão encapsulados dentro de um datagrama UDP onde está o cabeçalho da VXLAN e dentro pacote original do endpoint, conforme mostrado na figura anterior em "PACKET IN VXLAN" (pacote em destaque).

Características das VXLANs no SDA da Cisco:

- Chamada de VXLAN-GPO (Group Policy Option).
- Suportam segmentação tanto em Layer 2 (VLAN) como em Layer 3 (VRF) utilizando VNI ou VN-ID (VXLAN Network Identifier).
- O cabeçalho da VXLAN transporta o MAC original do endpoint dentro do payload ("MAC in IP").
- Suporta o uso de "Security Group-Tags" (SGTs) para implementações de segurança e também políticas de QoS (Security and QoS Policy).

Abaixo segue o cabeçalho completo de um pacote tunelado via VXLAN, chamado também de "VXLAN-GPO Header", pois, como já citado, ele suporta MAC-in-IP com VN-ID (VRFs para segregar em L3) e Group ID (utilizados para segurança e QoS).



Os dispositivos que formam os túneis VXLAN podem ser chamados também de **VTEP**s ou "**Virtual Tunnel End Points**" que normalmente é identificado pelo endereço de loopback que será utilizado pelo Underlay para formar os túneis.

Aqui no Fabric do SDA chamamos os VTEPs de Fabric Edge Nodes, os quais podem ser chamados no LISP de Ingress Tunneling e Egress Tunneling Router.

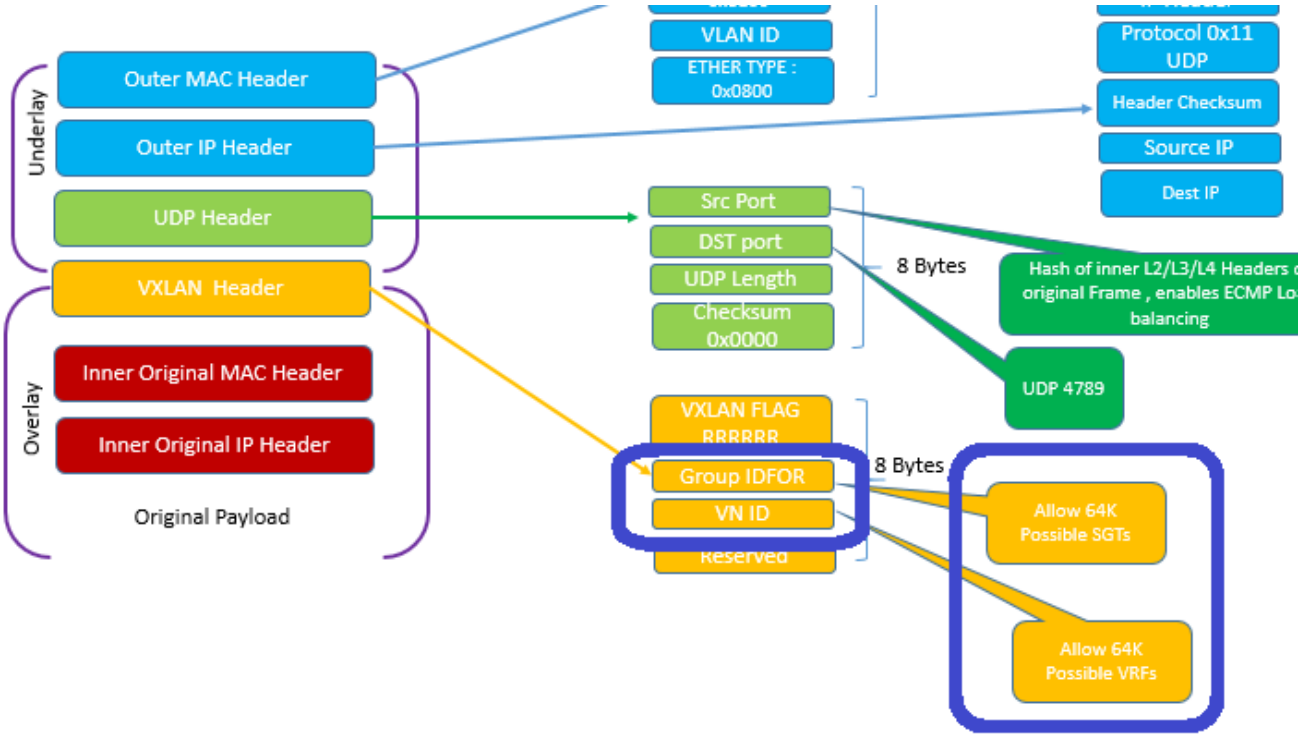
4.8 Policy Plane: Segurança e QoS

Para segurança e QoS o SDA utiliza o conceito do trustSec com SGTs (Security Group Tag) e VN-ID ou VNI (VXLAN Network Identifier).

Porém, o termo SGT no SDA foi transformado para "**Scalable Group Tag**", pois ele vai além da segurança como era utilizado no trustSec.

No SD-Access temos opção de fazer "macro segmentação" (macrosegmentation) utilizando o VNI e "micro segmentação" (microsegmentation) utilizando o SGT.

Essas informações são passadas no cabeçalho de cada túnel VXLAN criado, veja imagem a seguir.



A macro segmentação é feita com a criação de VNs ou **Virtuais Networks** e essas redes virtuais são segmentadas em camada-3, portanto são redes ou sub-redes isoladas entre si e precisarão de roteamento ou regras específicas para se comunicarem.

É como se criássemos VRFs (virtual routing and forwarding) para separar as redes virtuais que estão dentro de um mesmo ambiente físico.

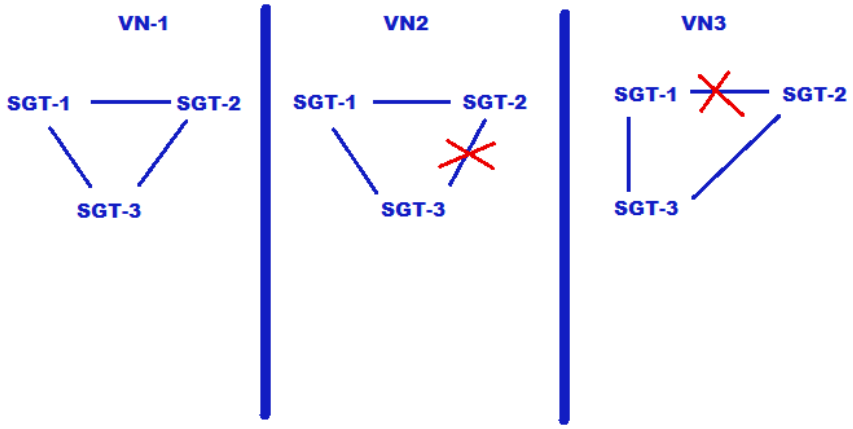
A micro segmentação é feita através dos SGTs e podemos comparar com as diversas VLANs que temos dentro do mesmo ambiente de rede segmentando nossas LANs.

Na prática os SGTs podem ser classificados utilizando endereços IP, nome de usuário, setores da empresa, grupo do Active Directory, tipo de dispositivo etc.

Com os SGTs criados, via DNA Center e ISE o administrador pode definir quais SGTs podem se comunicar com outras SGTs, definindo as políticas desses grupos.

Esta parte da implementação depende mais do ISE (Identity Services Engine) do que do próprio DNA Center.

Por exemplo, um ambiente pode ter três empresas ocupando a mesma infraestrutura física, portando teremos três VN-IDs (VN1, 2 e 3) para segregar cada uma delas.



Porém, dentro de cada empresa temos vários setores que separamos por VLANs, tipo de usuário e seja lá como for. Vamos utilizar VLANs para facilitar o entendimento.

Para cada VLAN temos políticas (policy) de comunicação entre elas, portanto vamos criar vários SGTs para identificar essas VLANs e definir como elas se comunicam.

Por exemplo, na imagem anterior dentro da VN1 todas as VLANs se comunicam entre si, porém na VN2, que é uma rede diferente, as VLAN 2 e 3 não podem ter acesso entre si.

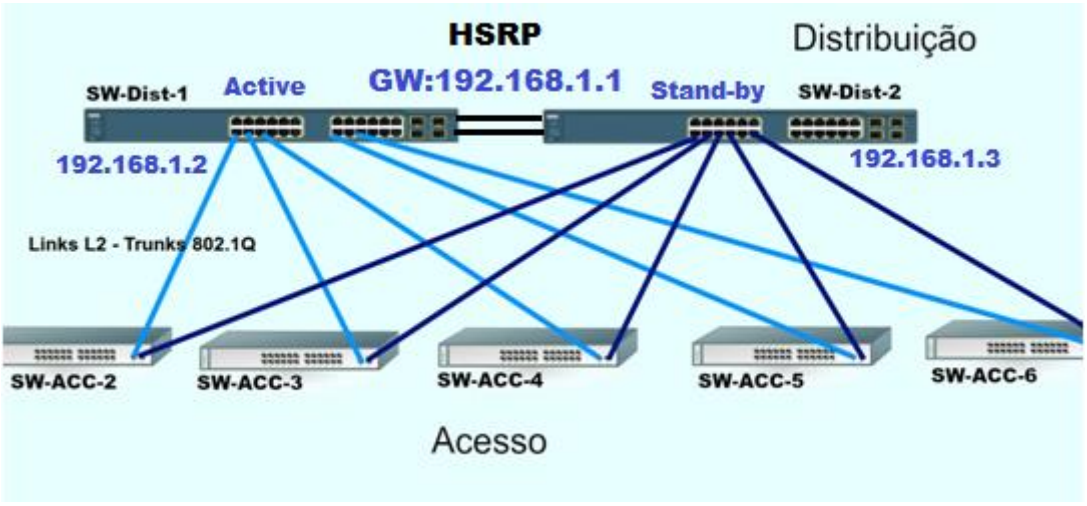
Esse mesmo conceito pode ser utilizado na classificação de QoS e priorização de tráfego, que juntamente com outras ferramentas de classificação de tráfego permitem a implementação de QoS de maneira muito mais simples em um Fabric SDA que em uma infraestrutura normal.

Se você está seguindo a trilha do CCNA já configurou ACLs no capítulo de Tópicos de Segurança e viu como pode ser complexa essa implementação... o mesmo ocorre com o QoS.

Porém, no SDA você vai configurar essas políticas (sua intenção) no DNA Center, o qual vai interagir com o ISE e "traduzir" essas políticas nas configurações necessárias para todos os dispositivos envolvidos no fabric.

4.9 Default Gateway: SDA versus Redes Tradicionais

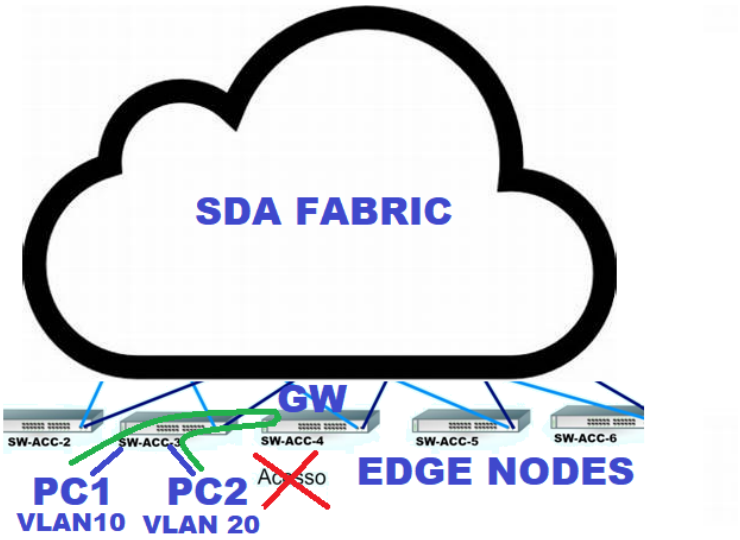
Em uma rede tradicional o default gateway que configuramos nos endpoints está situado nos switches de distribuição, sendo que por motivos de redundância utilizando um protocolo de redundância em primeiro salto como HSRP ou VRRP, veja imagem a seguir.



Portanto, na figura anterior os computadores e demais endpoints serão configurados com o endereço 192.168.1.1, o qual estará ativo no switch SW-Dist-1 e caso ele fique indisponível o switch SW-Dist-2 assume essa função, pois ele está como stand-by para o HSRP.

Mas vamos lembrar de um detalhe, no SD-Access os switches de distribuição agora estarão no Underlay e não serão mais acessíveis pelos endpoints, portanto o gateway precisa estar nos switches de acesso, os quais são os Edge Nodes no SDA, certo?

Mas o que aconteceria se escolhermos um dos switches como gateway? Por exemplo o switch SW-Acc-4? Todo tráfego da rede teria que ser encaminhado para ele para haver comunicação entre diferentes VLANs, mesmo que os endpoints estejam no mesmo switch, conforme imagem abaixo.



Se pensarmos bem isso não faz muito sentido, você fazer tráfego entre VLANs e hosts que estão no mesmo Edge Node como na figura anterior.

Para resolver esse problema e eliminar a necessidade de protocolos como HSRP e VRRP o SDA utiliza o conceito de "Anycast Gateways".

Cada VLAN ou sub-rede de usuário criada no Fabric terá uma interface VLN criada nos switches que conectam os endpoints (usuários) com um endereço IP de Anycast.

Esse endereço é o mesmo em todos os switches e é utilizado para fazer o roteamento local entre VLANs, assim como para que os endpoints acessem todas as demais sub-redes do Fabric.

Portanto, quando um endpoint migrar de um switch para outro na rede nada precisa ser feito, pois o endereço do gateway será sempre o mesmo, facilitando o Roaming dos dispositivos (mudança de edge node).

E isso tudo é transparente ao Underlay porque toda essa comunicação está sendo realizada através de túneis criados com as VXLANs.

4.10 Juntando as Peças do SD Access

Vamos resumir o que é o SD-Acces.

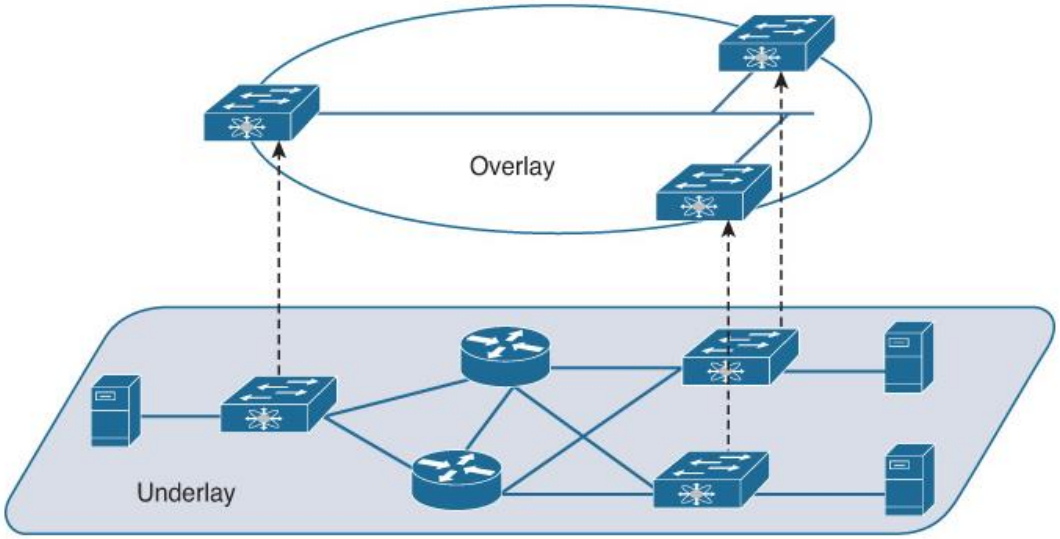
A rede física com seus equipamentos passa a se tornar um Underlay (camada inferior) que tem a função de transportar os pacotes entre Edge Nodes e demais elementos do Fabric.

Essa estrutura física não precisa e nem vai conhecer as redes dos usuários (endpoints), pois essas redes e sub-redes estão em outro espaço de endereçamento (IP ou Address Space).

Uma vez montado o Underlay dificilmente precisaremos configurá-los novamente.

Essa rede física pode ser tanto L2 como L3, porém o mais recomendado é uma rede L3.

O Overlay (camada superior) é como os usuários finais (endpoints) enxergam a rede, ou seja, é uma abstração lógica da rede realizada via software.



O Fabric é um sinônimo do Overlay, termo que representa a Rede Lógica criada com os túneis VXLAN e vista pelos usuários finais.

Portanto, em uma rede que utiliza o SD-Access, quando dois servidores, hosts, computadores ou endpoints quaisquer quiserem se comunicar é como se eles estivessem conectados à rede superior, sem a complexidade da rede inferior do desenho anterior.

Na prática é criada uma VXLAN entre os dispositivos para que os endpoints se comuniquem e o protocolo LISP é utilizado para mapear esses dispositivos na rede.

4.11 Controller e Management Planes

Até o momento nós estudamos como funciona o fluxo de informações, ou seja, como os dados entre dois endpoints são enviados através do Fabric ou da Rede SDA utilizando LISP e VXLANs.

Mas o "SD" do nome "SD-Access" significa "Software Defined", portanto temos que estudar também dois "Planes" muito importantes aqui:

- **Controller Plane**: onde reside a controladora SDN da solução Cisco DNA.
- **Management Plane**: como é a interface de gerenciamento, ou seja, a comunicação Northbound e Southbound na solução Cisco DNA.

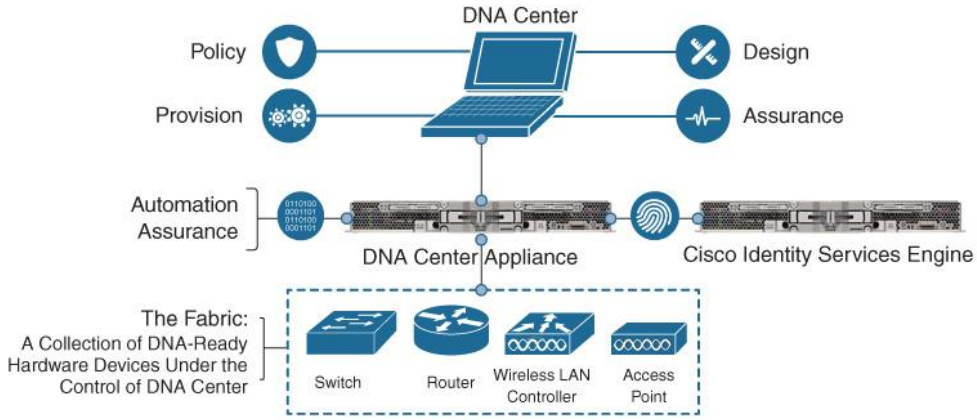
Vamos estudar cada um dos planes a seguir.

4.11.1 Controller Plane

A solução DNA tem como controller o DNA Center ou DNAC, o qual atualmente é apenas fornecido como um appliance físico, ou seja, não tem a disponibilidade de um DN Center virtualizado.

Ele pode ser montado em duas versões:

1. Stand-alone, ou seja, apenas um appliance ou
2. Em cluster, sendo que são necessários três appliances para montar o cluster.



Além disso, juntamente com o DNAC o controller plane utiliza o Cisco ISE para realizar as tarefas de identificação e políticas na rede, porém o administrador não fará nenhuma configuração direta no ISE, tudo é realizado via DN Center utilizando APIs em REST.

Diferentemente do DNAC o ISE pode ser adquirido em forma de appliance físico ou virtualizado.

O DNA Center está dividido em duas partes:

- **NCP** (Network Control Platform)
- **NDP** (Network Data Platform)

O NCP é responsável pela comunicação e configuração dos dispositivos do Fabric do SDN.

Isso pode ser realizado utilizando diversas formas, por exemplo, através de comandos do CLI, SNMP e NETCONF/YANG.

Já o NDP é utilizado principalmente para implementar o conceito de "Network Assurance", ou seja, para monitoração e troubleshooting do Fabric.

O NDP tem a função de coletar informações dos dispositivos através de protocolos como SNMP, Syslog, Netflow ou através de Streaming Telemetry (disponível em em dispositivos mais novos apenas).

Essas informações coletadas pelo NDP são enviadas para o "Assurance Engine", o qual tem a função de monitorar e informar sobre o estado da rede de forma proativa.

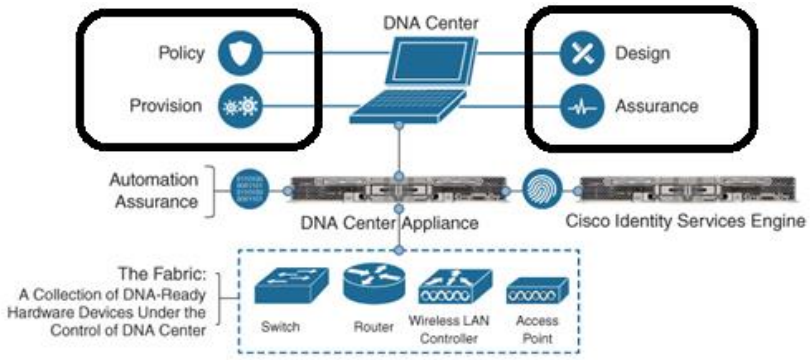
Portanto, se você instalar uma rede totalmente nova com base no SD-Access e DNA Center você não precisará mais entrar em cada dispositivo para fazer a configuração ou monitoração, toda essa interface será feita via DNA Center, economizando toneladas de comandos e tempo de execução de tarefas operacionais.

4.11.2 Management Plane

O DNA Center fornece uma interface Web para gerenciamento do sistema como um todo, assim como você pode interagir utilizando APIs com base em REST.

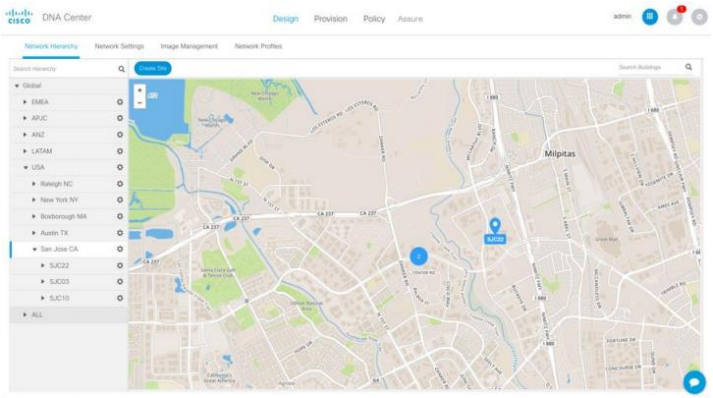
Na realidade, a interface web do DNA Center já utiliza APIs em REST para funcionar, por isso mesmo podemos utilizar esse mesmo princípio para gerenciar com APIs externos a plataforma.

Existem quatro workflows que o DNA Center já tem preparado para auxiliar na tarefa de implantar, manter, operar e resolver problemas de rede de forma mais simples e automatizada. Eles são acessíveis via interface web.

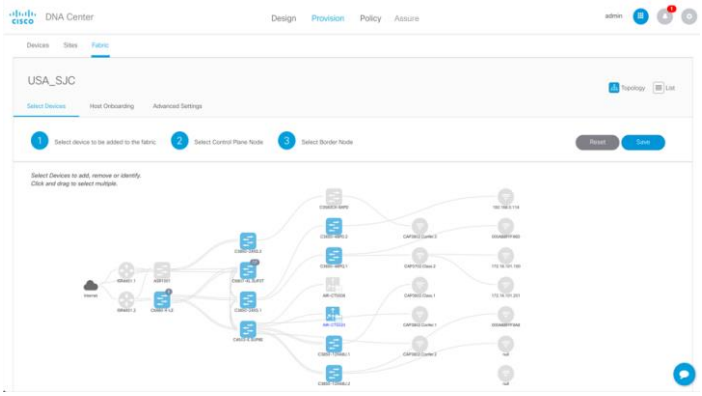


Utilizando a mesma figura do tópico anterior podemos notar que existem quatro ícones saindo do DNA Center:

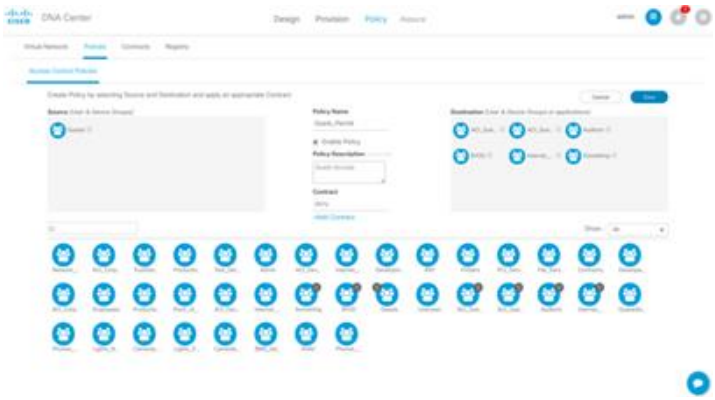
- **Design**: você vai encontrar nesse workflow as opções de Hierarchy (Sites, Geographic Location, Building ID, Floor number), Network Settings (IP Polls, DNS, DHCP, WAN QoS, Wireless), Image Management, Network Profiles e Authentication Template.



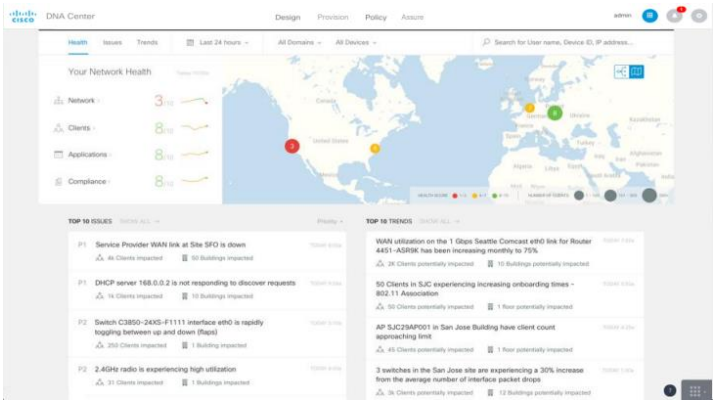
- **Policy**: esse workflow traz as opções das políticas de macro segmentação ou Macrosegmentation (L3 através das Virtual Networks – VNs) e micro segmentação ou Microsegmentation (L2 através do SGTs ou Scalable Group Tags).



- **Provision**: o terceiro workflow é onde realmente vamos descobrir os elementos de rede e definir "quem fala com quem". temos as opções Devices, Fabric e Services.



- **Assurance**: o quarto e último workflow cuida da monitoração e troubleshooting, definição "muito simples" para o poder do que esse workflow traz, porém por enquanto esse é o fundamental. Temos as opções de Health Dashboard, Trends and Insights e Manage. Aqui temos o recurso de "360-Degree Contextual Insights" que permite você analisar em detalhes a saúde da rede.



4.12 Resumo das Soluções SDN e SD-Access Cisco

Esse tópico deveria estar no começo desse capítulo, mas decidi inverter porque fica mais fácil de entender depois de ter visto como funciona o SD-Access para o DNA Center.

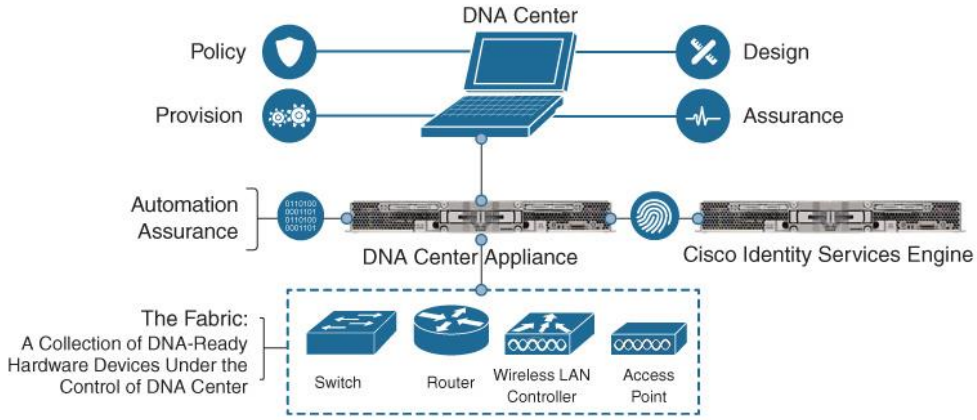
Onde essas soluções de SDN e SD-Access são implementadas na prática? Resposta: Cisco DNA (Cisco Digital Network Architecture), ACI (Application Centric Infrastructure) e SD-WAN (Cisco Software Defined WAN).

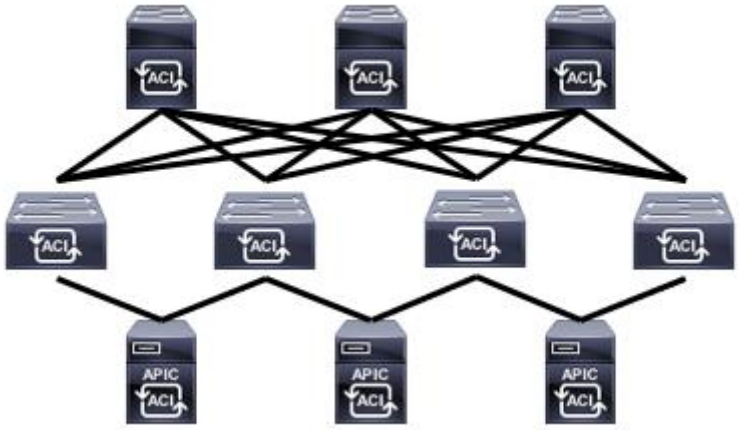
A base do funcionamento das soluções SD Access para Data Center e Campus é a mesma: Controllers, VXLAN, LISP...

Porém, a configuração e elementos internos do software variam um pouco, pois o ACI é mais voltada para soluções de Data Centers e o DNA para ambiente corporativo.

Veja abaixo uma comparação entre essas duas soluções.

- **Controladora**: ACI utiliza o APIC e o DNA o DNA Center ou DNAC.
- **Arquitetura**: ACI utiliza Spine-and-Leaf e o DNA normalmente Underlay em três camadas.



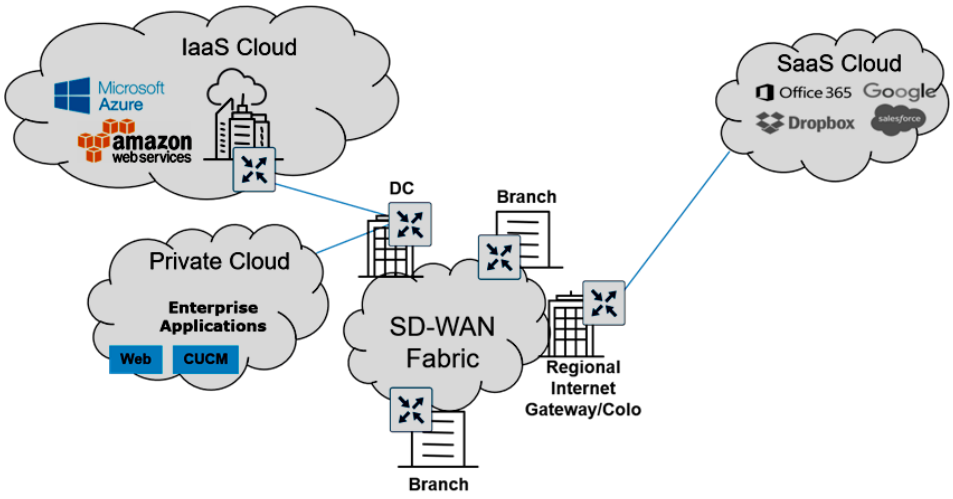


- **Número de controladoras**: ambas as soluções podem utilizar uma controladora ou três em cluster, porém para o DNA Center dependendo da versão existe apenas opção de uma controladora.
- **Utilização**: ACI é utilizado em Data Center e DNA em Empresas (Campus, WAN e Branch Networks).

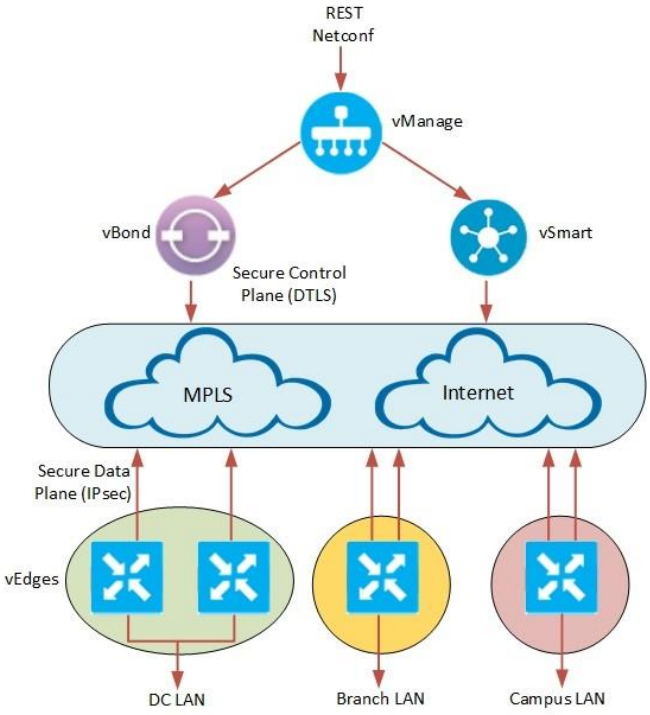
Outra solução SDN da Cisco é o SD-WAN ou Software Defined WAN.

O conceito do SD-WAN é parecido com o que estudamos sobre Underlay e Overlay, porém na WAN o Underlay são os diversos links WAN e de Internet fornecidos por um Service Provider (Provedor de Serviços de Telecom).

Já o Overlay serão túneis L3 protegidos pelo protocolo IPSec que formarão o Fabric do SD-WAN(Rede lógica do SD-WAN), veja figura a seguir.



Na topologia SD-WAN temos os seguintes componentes:



- **vManage**: interface gráfica de Gerenciamento utilizada para monitorar e configurar a solução Cisco SD-WAN.
- **vBond**: responsável pela autenticação inicial dos vEdges e orquestra a conectividade entre vSmart, vManage e vEdge. Podemos dizer que é o "Controller SD-WAN".
- **vSmart**: responsável por manter uma conexão segura com cada roteador e distribuir informações de rotas e políticas por meio do OMP. Também orquestra a conectividade segura do plano de dados entre os roteadores, enviando informações de chave de criptografia. Também tem a função de controller.
- **vEdge**: pode ser um dispositivo em hardware ou oftware utilizado para conectar de forma segura diversos pontos da empresa através da WAN. São os dispositivos responsáveis pelo tráfego, segurança, qualidade de serviço (QoS), e muito mais. São como os Edge Nodes do SDA.

Citamos aqui as três principais soluções SDN e SD-Acces Cisco para que você tenha conhecimento, porém o grande foco para quem está na trilha do CCNA é o DNA Center.

Além disso, todas as soluções permitem o uso de APIs e fornecem possibilidade de uso de automações e programabilidade através de REST e outras ferramentas.

Vamos tratar desse assunto no próximo tópico.

5 REST-based APIs, CRUD, Verbos HTTP e Data Encoding

5.1 Introdução

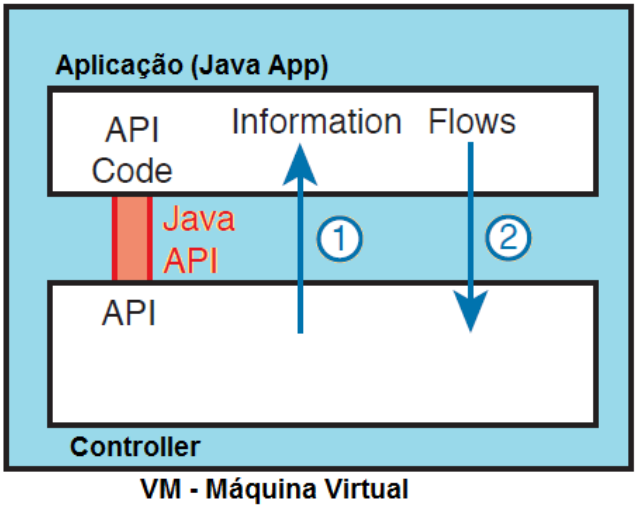
Uma **API** ou "**Application Programming Interface**" é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web.

Em tradução para o português "**Interface de Programação de Aplicativos**".

Portanto, uma API permite que você crie uma comunicação direta com uma aplicação sem conhecimento ou intervenção dos usuários, pois elas funcionam através da comunicação de diversos códigos, definindo comportamentos específicos de determinado objeto em uma interface.

Por exemplo, uma controladora de rede utiliza Java em sua API, então quaisquer fabricantes ou desenvolvedores podem escrever aplicativos (apps) utilizando as regras desse API para trocar informações com esse controller dentro do mesmo sistema operacional.

Essa aplicação pode coletar informações sobre os dispositivos de rede (1) e até programar como os fluxos serão encaminhados na rede entre os dispositivos, alterando as informações das tabelas de encaminhamento dos dispositivos (2), por exemplo.



O **REST** (**Representational State Transfer**) permite que os aplicativos em diferentes hosts utilizem mensagens HTTP para transferir dados entre eles.

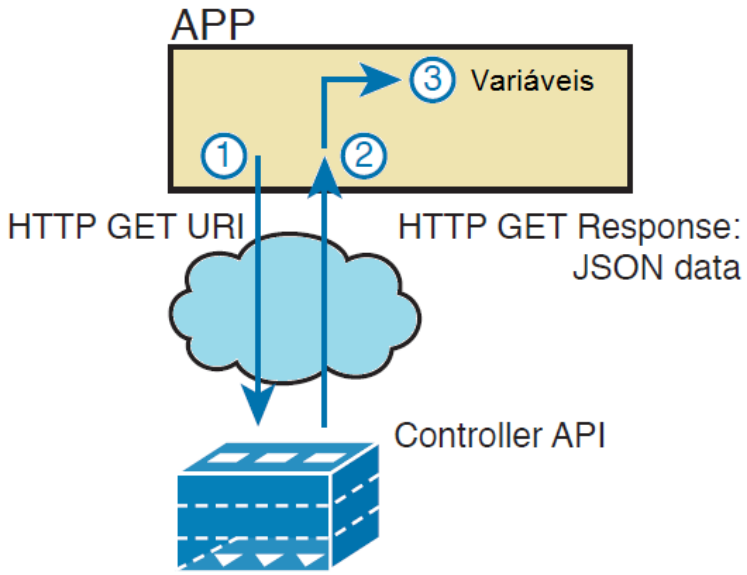
Por exemplo, o controller está em uma determinada VM e as aplicações em outra VM, portanto nesse exemplo os aplicativos precisam enviar e receber mensagens pela rede IP, por isso precisam utilizar APIs RESTful.

Com o REST tudo começa com um aplicativo enviando um HTTP GET (1) solicitando uma **URI** (Uniform Resource Identifier ou Identificador de Recursos Universal) específica.

Esse HTTP GET é como um HTTP GET normal, igual ao que utilizamos para solicitar páginas da Internet.

Só que ao invés de solicitar uma página da web, essa URI identifica um objeto específico no controller, normalmente uma estrutura de dados que a aplicação precisa para processar uma determinada informação.

O URI pode identificar um objeto que é a lista de interfaces que um dispositivo possui e o estado operacional de cada uma dessas interfaces.



Utilizando o API do REST o controller recebe esse GET da aplicação, processa e responde com outra mensagem HTTP GET (2) contendo o objeto solicitado pelo aplicativo (3).

Em maioria dos APIs que utilizam o REST será solicitado e enviado estruturas de dados ao invés de páginas de web como você receberia em um browser, normalmente essa resposta contém nomes e valores de variáveis, tudo em um formato que pode ser facilmente utilizado por um programa de computador.

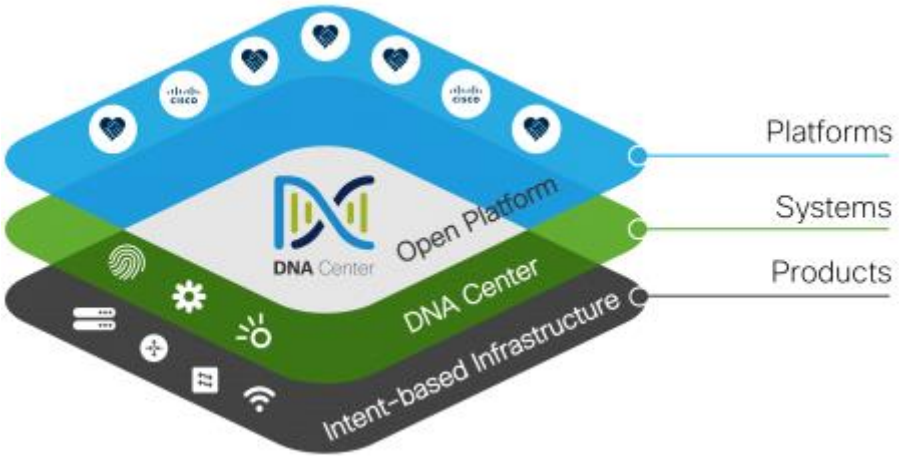
Os formatos mais comuns utilizados em redes programáveis são o **JSON** (JavaScript Object Notation), **XML** (eXtensible Markup Language) e **YAML** (YAML Ain't Markup Language).

Vamos estudar mais a fundo esses assuntos nos tópicos a seguir.

5.2 Automação e Programabilidade no DNA Center

Se você entrar na interface gráfica do DNA Center encontrará um **quinto workflow** chamado "**Platform**" que possibilita a automação e mostra como funciona sua API.

Podemos dividir a solução Cisco DNA em três camadas para entender melhor a automação e programabilidade dela: **Products**, **Systems** e **Platform**, conforme imagem a seguir.



A camada inferior dos produtos representa os dispositivos que fazem parte do Underlay, que forma a infraestrutura física do Fabric SDA.

Depois temos a própria controladora DNA Center e o ISE, ou seja, representa a infraestrutura que controla todo o sistema e as demais interações que ele pode ter.

E a camada chamada **Platform** ou **Plataformas** está no topo de tudo isso e é quem possibilita toda a automação e programabilidade da solução Cisco DNA.

Veja que no capítulo anterior exploramos essa camada com os quatro workflows que a interface gráfica (GUI) fornece.

Porém isso não é tudo, pois podemos através da API do DNA Center explorar muito mais todas essas capacidades e criar aplicações próprias para customizar a operação, monitoração e resolução de problemas da Infraestrutura como um todo.

Isso tudo pode ser realizado através de "**REST APIs**".

Por exemplo, você pode escrever scripts utilizando Python ou utilizando uma ferramenta de automação como o Ansible (vamos estudar posteriormente) para buscar informações sobre o status dos dispositivos de rede ou dos clientes e com base nessas informações configurar scripts para que ações sejam tomadas.

A ação pode ser desde mandar um e-mail até enviar um arquivo para que uma configuração seja aplicada ao dispositivo e resolver automaticamente uma determinada situação.

Nesse novo mundo de automação e programabilidade você agora tem inúmeras possibilidades e pode customizar conforme as necessidades da empresa.

Inclusive outras aplicações comerciais podem ser elaboradas utilizando o API que o DNA Center disponibiliza e ser vendida para empresas que usam a solução.

5.3 REST e RESTfull APIs

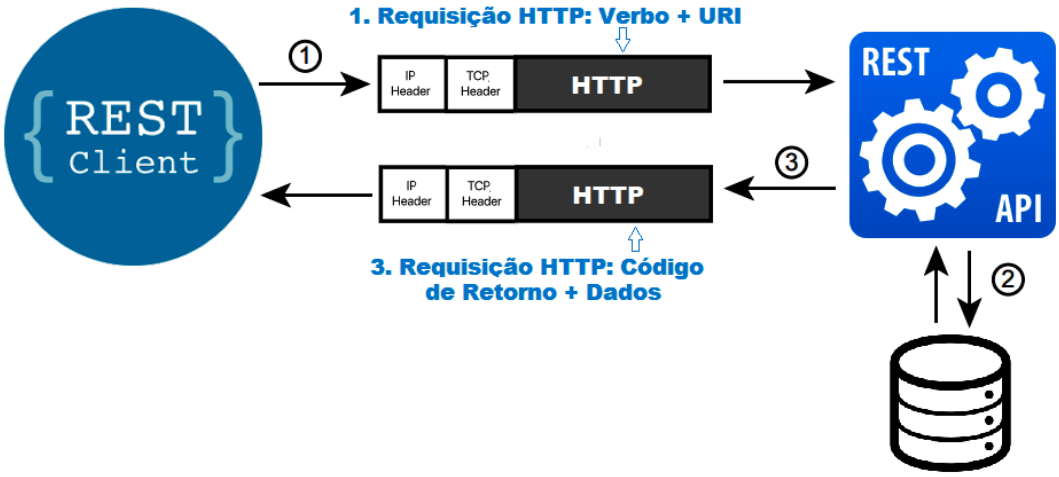
No mundo da programação, o **REST** ou "**Representational State Transfer**" tem o objetivo de definir características fundamentais para o desenvolvimento de aplicações Web.

Ele foi pensado como uma abstração da arquitetura da web, ou seja, um conjunto de princípios e definições necessários para a criação de um projeto com interfaces bem definidas.

E como já estudamos, a utilização da arquitetura REST permite a comunicação entre aplicações utilizando mensagens do HTTP, portanto um API RESTfull funciona praticamente igual a um site da internet (Website).

A definição acima nos permite dizer que aplicações desenvolvidas em REST são cliente/servidor, onde um cliente faz a requisição (Client API Call) e o servidor REST responde.

Veja exemplo na imagem a seguir, onde o API do cliente faz uma requisição (1) ao servidor que tem a API em REST, o servidor recebe a mensagem e processa a solicitação (2), monta a resposta e envia para o cliente (3).



Note que a requisição é realizada utilizando os Verbos ou Métodos HTTP (HTTP Verbs ou HTTP Methods) e uma URI (Uniform Resource Identifier ou "Identificador Uniforme de Recurso").

A resposta vai conter um código de retorno do HTTP e os dados solicitados formatados em um determinado padrão, por exemplo, em JSON ou XML.

Mas o que contém nessa resposta? As variáveis que você disponibilizou via API, por exemplo, você pode ter solicitado a lista de todas as interfaces em shutdown que estão situadas nos Edge Nodes do Fabric criado com o DNA Center.

Por exemplo, você deseja obter a lista de dispositivos que o DNA Center possui pode utilizar o Postman (vamos estudar como utilizar posteriormente) e enviar uma solicitação GET para a seguinte URI: https://10.0.1.10/dna/intent/api/v1/network-device

A resposta parcial que você obterá será como exemplo abaixo:


```
{
    "response": [
        {
            "memorySize": "3735220224",
            "family": "Wireless Controller",
            "type": "Cisco 3504 Wireless LAN Controller",
            "macAddress": "50:61:bf:57:2f:00",
            "softwareType": "Cisco Controller",
            "softwareVersion": "8.8.111.0",
            "deviceSupportLevel": "Supported",
            "platformId": "AIR-CT3504-K9",
            "reachabilityFailureReason": "",
            "series": "Cisco 3500 Series Wireless LAN Controller",
            "serialNumber": "FCW2218M0B1",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/></status>",
```


Nesse exemplo o IP de gerenciamento é 10.0.1.10 da controladora para acesso web.

Essas requisições e respostas não ficam armazenadas no servidor, isso quer dizer que ele não guardará o estado, ou seja, a cada requisição uma nova resposta será montada.

Isso porque o REST trabalha com o conceito Stateless, além disso, note que a interface é uniforma (Uniform Interface), baseada em camadas (Layered) e seus componentes possuem uma hierarquia.

Além disso, uma resposta pode ser armazenável ou não em cachê (Cacheable), porém isso deve ser explicitado na resposta para que o cliente possa saber que pode reutilizar a informação em outro momento.

Nesse exemplo anterior fizemos uma coleta de informações do DNA Center, porém você também pode inserir informações via API e o que define o que será feito são os Verbos HTTP, que vamos estudar a seguir.

5.4 CRUD e Verbos HTTP

O CRUD e os verbos HTTP (Verbs ou Methods) são a maneira que você vai interagir ou conversar via REST API com o DNA Center, pois aqui estão as "ações" que você vai fazer via API.

CRUD é o acrônimo das palavras "**Create, Read, Update and Delete**", as quais são as quatro principais funções para interagir com aplicações e seus bancos de dados.

A função de cada uma das quatro operações do CRUD é:

- **Create**: permite que o cliente crie ou adicione novas entradas.
- **Read** (Retrieve): permite que o cliente leia, recupere ou veja entradas existentes.
- **Update**: permite que o cleinte atualize ou edite entradas existentes.
- **Delete** (Destroy): permite que o cliente remova entradas existentes.

Trazendo para o mundo do DNA Center, você pode utilizar a API RESTfull dele e utilizar o Create para criar uma nova localidade na sua topologia, a qual terá um nome que será armazenado no banco de dados do DNA Center como uma variável.

Com um READ você poderá ver tudo o que você criou e caso tenha digitado o nome de algum site ou andar errado pode utilizar o conceito de Update para alterar essa variável.

Na prática utilizamos o HTTP para espelhar as ações do CRUD, ou seja, utilizamos o conceito do Request/Reply do HTTP para enviar/solicitar informações do cliente para o servidor, recebendo por consequência uma resposta.

Abaixo seguem alguns verbos ou métodos HTTP:

- **OPTIONS**: Busca os métodos http válidos e outras opções.
- **GET**: Busca uma entrada.
- **HEAD**: Busca apenas o header de uma entrada.
- **PUT**: Atualiza uma entrada.
- **POST**: Cria uma entrada.
- **DELETE**: Remove uma entrada.
- **PATCH**: Atualiza parcialmente uma entrada.

Tudo isso é transportado via TCP porta 80 para o HTTP ou porta 433, no caso do HTTPS.

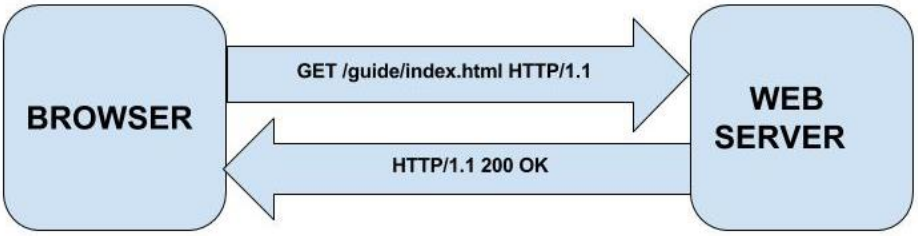
Abaixo segue uma correlação entre o CRUD e os verbos HTTP mais importantes para RESTfull APIs:

- Create: POST
- Read: GET
- Update: PATCH, PUT
- Delete: DELETE

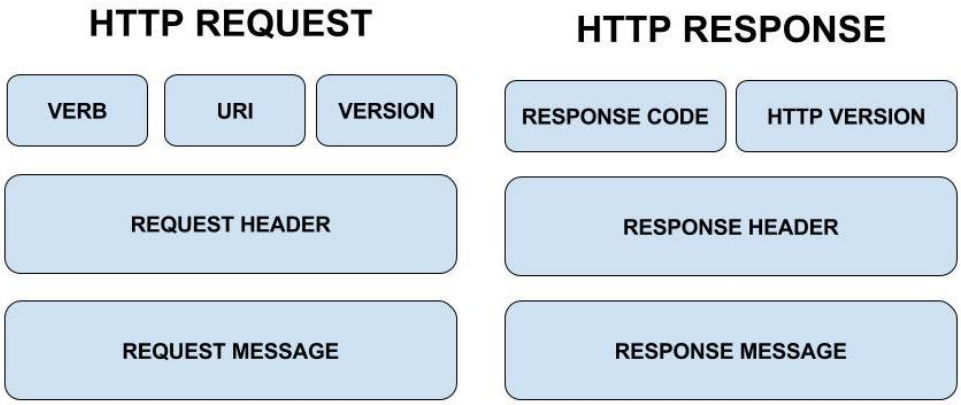
Portanto, se você criar uma API ou for utilizar um cliente REST como o Postman para buscar uma informação do DNA Center você deve utilizar um GET (Read no CRUD), para criar ou inserir novas variáveis utilize um POST (Create no CRUD), para atualizar uma variável utilize o PUT (Update no CRUD) e para remover uma variável utilize o DELETE (Delete no CRUD).

5.5 Troca de Mensagens: HTTP Client/Server

Portanto, uma API que utiliza REST terá um cliente gerando uma requisição HTTP (HTTP Request), a qual será enviada para um servidor REST, recebida, processada e o servidor enviará uma resposta HTTP para o cliente solicitante (HTTP Response).



Os verbos HTTP (que correspondem ao CRUD) são enviados na requisição, assim como outros parâmetros como a URI, cabeçalhos (headers) do HTTP e o corpo da mensagem (body ou Request Message).



Já na resposta do servidor você receberá um código de resposta (Response Code), os headers e a resposta da mensagem (Body ou Response Message).

5.5.1 Estrutura da URI e Requisições REST

Lembra desse exemplo abaixo? Nele a gente desejava obter a lista de dispositivos que o DNA Center através do Postman e enviou um **HTTP GET** para a seguinte URI: https://10.0.1.10/dna/intent/api/v1/network-device

Nesse exemplo utilizamos um verbo HTTP para leitura, que foi o GET, para requisitar informações sobre um determinado recursos e obter suas variáveis.

Na realidade utilizamos a versão segura do HTTP que é o HTTPS e o endereço do host (hostname) do DNA Center é 10.0.1.10.

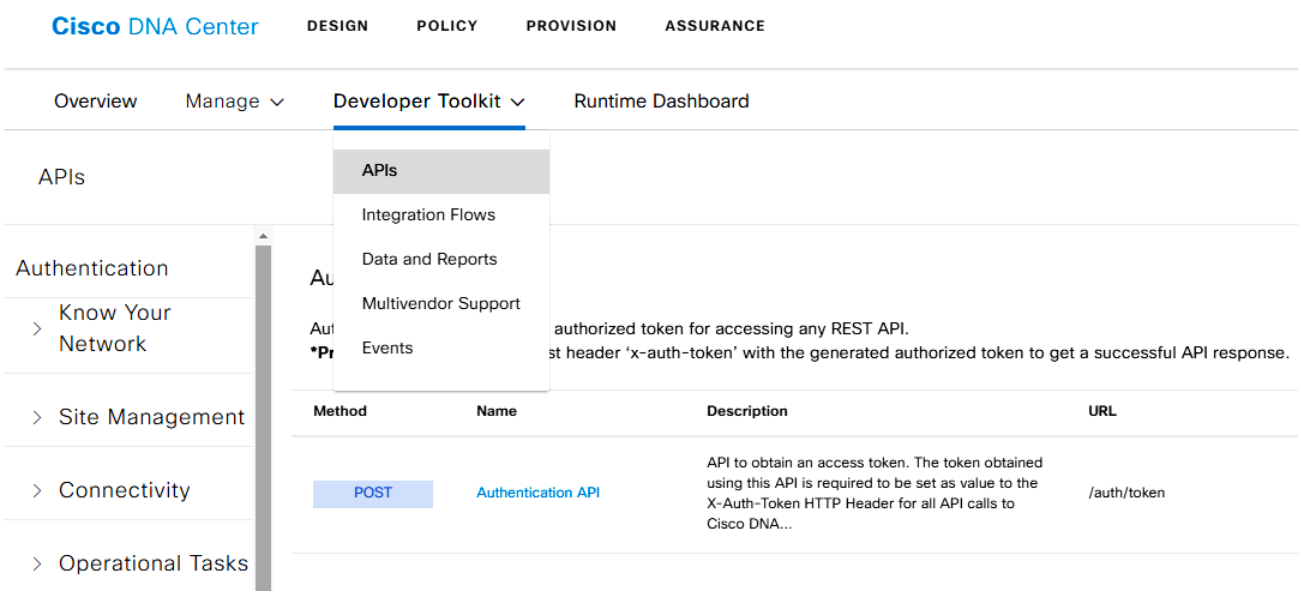
Depois temos o recurso (resource ou entrada ou variável) buscado logo após o endereço do DNA Center, o qual poderia ser um nome de URL também, pois esse servidor pode estar configurado em uma entrada do serviço de DNS da empresa.

Cada recurso dentro da API do DNA Center é identificado de forma hierárquica através de uma URI, que em nosso exemplo foi "dna/intent/api/v1/network-device".

Essa parte da URI também é chamada de "resource path" ou caminho do recurso.

O início da URI "dna/intent/api/v1/" é um valor padrão definido pela versão de DNA Center que você estiver conectado e a parte final "/network-device" é o recurso específico definido na API Server.

Esses recursos podem ser encontrados dentro do DNA Center na opção "**Platform>Developer Toolkit>APIs**", onde existe uma lista dos verbos e recursos suportados pelo sistema, veja imagem abaixo.



Cada recurso disponibilizado pela API do DNA Center terá uma URI específica e novos recursos são adicionados e você poderá ter acesso, pois o DNA Center é um sistema em evolução.

Você também pode utilizar após a URI um ponto de interrogação e passar outros parâmetros (parameters) na mensagem de requisição. Veja exemplo abaixo:

- https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device?macAddress=f8:7b:20:67:62:80

Na mensagem foi solicitada a lista de dispositivos através da utilização do recurso "/network-device", porém queríamos os dados apenas do dispositivo que tem o endereço MAC f8:7b:20:67:62:80 utilizando o "ponto de interrogação" seguido do sinha de "igual" e o endereço buscado: "**?macAddress=f8:7b:20:67:62:80**".

Isso poderia ser utilizado em uma API Cliente para verificar se existe um dispositivo com esse endereço MAC, por exemplo.

Se você tivesse que buscar manualmente esse dispositivo, teria que entrar um a um via SSH ou Telnet, dar um "show version" e buscar se o MAC f8:7b:20:67:62:80 está nessa resposta.

Você pode também o "&" (e comercial) para concatenar uma requisição com vários parâmetros, por exemplo, você quer encontrar um dispositivo pelo IP e MAC poderia utilizar a URI abaixo: https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device?macAddress=f8:7b:20:67:62:80&managementIPAddress=10.1.2.3

Esses parâmetros de busca podem ser chamados de "**Query Parameters**".

5.5.2 Response Status Codes

Ao se comunicar com um REST API você vai receber um código que representa o status daquela requisição em específico, o qual será enviado pelo servidor como resposta a uma requisição de um cliente.

Essas mensagens são muito importantes para sabermos se a requisição foi ou não bem sucedida, veja os códigos abaixo:

Código	Descrição
200	Sucesso
400	Parâmetros inválidos
404	Não Encontrado
401	Não autenticado
403	Não autorizado
409	Conflito
413	Tamanho do corpo da requisição é maior que o esperado
500	Erro interno no servidor

Por exemplo, ao enviar uma requisição a um servidor REST e receber a mensagem de 200 (OK) quer dizer que o que você procurava foi encontrado e você terá os dados solicitados.

Se você receber ao invés disso uma mensagem 401 na repostas é porque você não está autenticado para obter as informações solicitadas.

5.5.3 Cabeçalho do HTTP ou Headers

O cabeçalho do HTTP traz detalhes e meta dados (meta-data), sendo utilizado para definir alguns parâmetros importantes na comunicação com o servidor.

As principais informações são:

- "**Content-Type**": define o tipo de formato que teremos no corpo da mensagem, por exemplo, podemos ter o tipo como "application/json", indicando que utilizaremos o JSON como formato dos dados.
- "**Accept**": define o formato que deve ser utilizado na resposta respectivamente, por exemplo, "application/json".
- "**Authorization**": campo que fornece as informações de autenticação, ou seja, as credenciais para uso da API.
- "**Date**": traz as informações de data e hora das mensagens.

Existem vários métodos de autenticação que podem ser utilizado em REST APIs, desde abertos e sem autenticação até os mais complexos utilizando certificados digitais.

Normalmente você encontra na Internet APIs abertos para consulta (GET), porém a realização de alterações de parâmetros não são permitidas.

O **Basic Authentication** é um esquema de autenticação bem simples especificado no protocolo HTTP, o qual será utilizado para fazer a autenticação no DNA Center em breve ainda nesse curso.

Nesse tipo de autenticação o cliente envia uma requisição com o header (Authorization) que contém a palavra **Basic** e o nome de usuário e senha, separados por dois pontos (:) no formato de base64.

Por exemplo, para autorizar o usuário devnetuser com senha cisco123!, o client enviaria na requisição, o seguinte header:

- Authorization: Basic YWRtaW46R3JhcGV2aW5lMQ==

Como o formato base64 é muito fácil de ser decodificado, a basic authentication só é recomendada quando são utilizados outros complementos de segurança, como por exemplo, o protocolo HTTPS.

5.5.4 Dados ou Body

Os dados estão contidos no corpo (body) da mensagem HTTP e será utilizado para inserir dados (POST, PUSH e PATCH) ou então para verificar a resposta de uma requisição (GET).

Veja abaixo a resposta parcial de um comando GET enviado a um Cisco DNA Center solicitando informações sobre seu inventário de dispositivos.


```
{
    "response": [
        {
            "memorySize": "3735220224",
            "family": "Wireless Controller",
            "type": "Cisco 3504 Wireless LAN Controller",
            "macAddress": "50:61:bf:57:2f:00",
            "softwareType": "Cisco Controller",
            "softwareVersion": "8.8.111.0",
            "deviceSupportLevel": "Supported",
            "platformId": "AIR-CT3504-K9",
            "reachabilityFailureReason": "",
            "series": "Cisco 3500 Series Wireless LAN Controller",
            "serialNumber": "FCW2218M0B1",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/></status>",
```

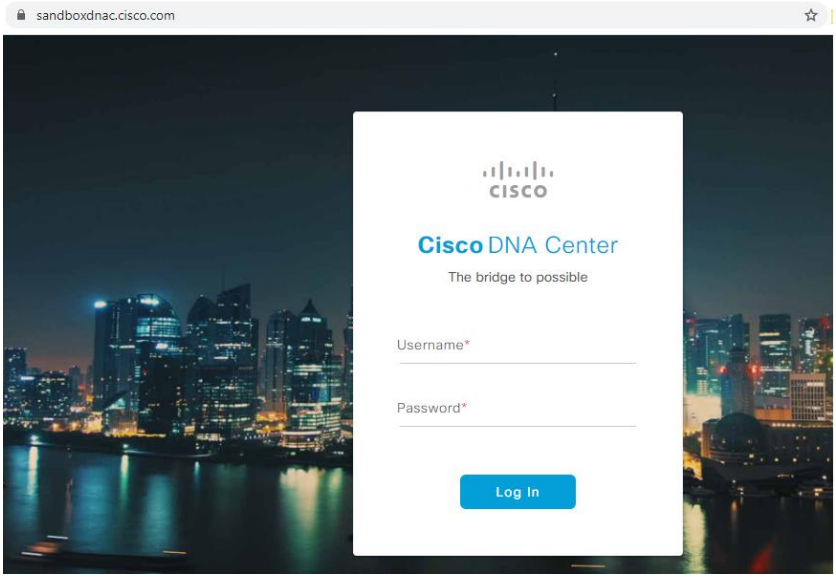

Normalmente os dados estão formatados em JSON ou XML, os quais serão estudados ainda nesse curso.

5.6 Conectando-se ao DevNet Always-On Sandbox

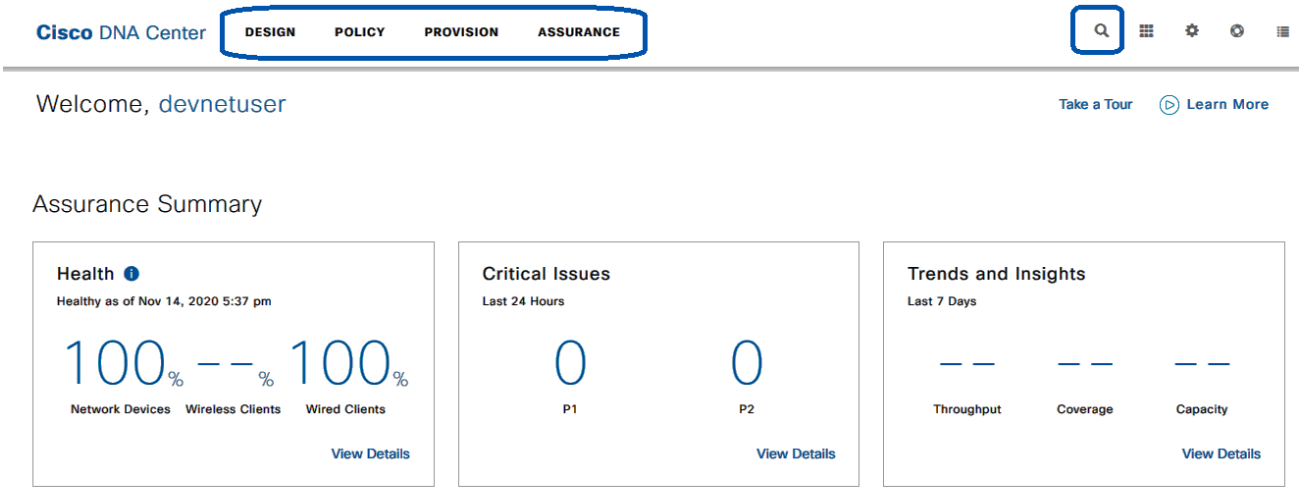
Existe um DNA Center com alguns dispositivos disponibilizado pela Cisco chamado "DevNet Always-On Sandbox" que você pode utilizar para visualizar tudo o que falamos até o momento, tanto via interface gráfica como através de um REST API.

Para acessar a interface gráfica basta abrir seu browser e utilizar a seguinte URL e dados de acesso:

- https://sandboxdnac.cisco.com
- Nome de usuário: **devnetuser**
- Senha: **Cisco123!**



Após o login você entrará na tela inicial do DNA Center e terá acesso aos quatro principais Workflows. Para acessar as opções e informações sobre a API do DNA Center vá na lupa de pesquisa e digite API.

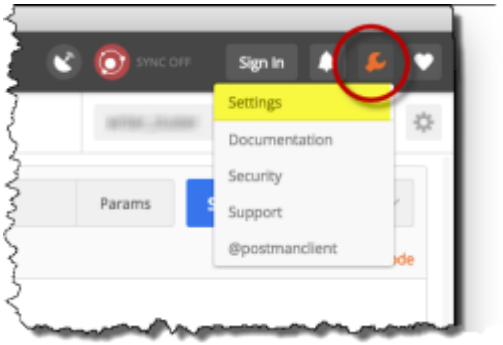


Para acessar o DNA Center utilizando REST vamos utilizar o Postman, o qual vamos ensinar como instalar e configurar a seguir.

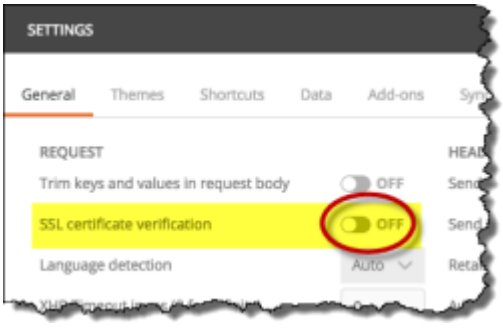
5.6.1 Instalando e Configurando o Postman

Vamos aqui aprender a acessar das duas formas, porém vamos utilizar o Postman (gratuito) para o estudo do REST.
Para baixar o Postman vá na página oficial (https://www.getpostman.com/) faça o dowload, instale o aplicativo, inicialize-o e realize os seguintes ajustes:

1. Procure o ícone de ajustes das configurações (Settings), clique nele.



2. Na sessão chamada "Request" coloque a opção "SSL certificate verification" em OFF.



5.6.2 Gerando a Chave de Autenticação com o Sandbox

O próximo passo é gerar a chave de autorização (Authorization Token) que será necessário para enviar as mensagens e coletar informações do API do DNA Center seguindo os passos abaixo:

1. Inicie o Postman se ainda não o tiver feito.

2. Crie uma nova solicitação.

Certifique-se de que a guia **Create New** esteja selecionada na tela de abertura do Postman e clique no botão **Request** na seção **Building Blocks**.

O Postman vai mostrar uma caixa de diálogo, dê um nome e uma descrição à sua nova solicitação. Salve-a em uma coleção nova ou existente.

3. Defina o tipo de solicitação para **POST**.

Para novas solicitações, o Postman define o tipo de solicitação padrão como **GET**, por isso você precisa alterar esta configuração para **POST** para esta solicitação específica.

4. Forneça o **URL** de solicitação REST.

Insira o URL de solicitação no campo de texto próximo ao menu **request-type**, substituindo o endereço IP pelo nome de domínio **sandboxdnac.cisco.com** para utilizar o "DevNet Always-On Sandbox".

No Cisco DNA Center versão 1.2.6 e superior, o URL deve ter o seguinte formato:
https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token

Para versões anteriores do Cisco DNA Center, o URL deve ter o seguinte formato:
https://sandboxdnac.cisco.com/api/system/v1/auth/token

Para localizar o número da versão do Cisco DNA Center, escolha **About Cisco DNA Center** no menu em formato de "engrenagem" em qualquer página da interface de usuário do Cisco DNA Center.

5. Escolha **Basic Auth**.

Na guia **Authorization** abaixo do campo URL, vá no menu Type e selecione **Basic Auth**.

O Postman vai fornecer um formulário de login, digite um nome de usuário e senha do DevNet Always-On Sandbox:

Nome de usuário: **devnetuser**
Senha: **Cisco123!**

6. Clique em Enviar.

Postman vai enviar a solicitação ao Cisco DNA Center e mostrar uma resposta.

Se seu login for bem-sucedido, o Cisco DNA Center retornará uma resposta 200 OK e a resposta formatada em JSON semelhante ao seguinte exemplo:

```
JSON
 {
   "Token": "eyJ0eXA ... O5uEMR-tc"
 }
```

Esta mensagem em JSON vai trazer a mensagem "Token" e seu valor associado, sendo que este valor é o próprio token de autorização.

A chave é o valor que está entre aspas duplas ("), basta copiar esse valor para fazer uma requisição REST ao DNA Center, porém essa chave pode expirar e se você receber alguma mensagem desse tipo (relacionada a chave) posteriormente, basta usar o mesmo script para gerar uma nova chave.

5.6.3 Exemplo de Busca de Informações no DNA Center via REST

Com a chave de autorização gerada vamos agora fazer um exemplo de coleta de informações via REST utilizando o Postman.

Vamos coletar a lista de dispositivos (device inventory ou inventário de dispositivos) que estão conectados ao Fabric do Sandbox seguindo os passos abaixo:

1. Altere o tipo de de requisição no Postman para GET.
2. Entre com a URL "https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device" (sem as aspas).
3. Adiciona uma autenticação do tipo "X-Auth-Token" no cabeçalho da mensagem HTTP.
    a. Clique em Headers.
    b. No campo chamado "key" digite "X-Auth-Token" (sem as aspas).
    c. No campo "value" insira o Token que obtivemos na etapa anterior sem as aspas.
4. Envie a requisição.

Se tudo deu certo você vai receber um HTTP 200 OK e a mensagem parecida com a que será mostrada abaixo formatada conforme o JSON.


```
{
   "response": [
      {
         "memorySize": "NA",
         "family": "Switches and Hubs",
         "hostname": "cat_9k_1",
         "macAddress": "f8:7b:20:67:62:80",
         "serialNumber": "FCW2136L0AK",
         "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/></status>",
         "deviceSupportLevel": "Supported",
         "collectionStatus": "Managed",
         "apManagerInterfaceIp": "",
         "tagCount": "0",
         "tunnelUdpPort": null,
         "waasDeviceMode": null,
         "upTime": "89 days, 10:04:42.26",
         "lastUpdateTime": 1605375343707,
         "collectionInterval": "Global Default",
         "softwareType": "IOS-XE",
         "softwareVersion": "16.6.4a",
         "associatedWlcIp": "",
         "bootDateTime": "2020-08-17 07:31:43",
         "errorCode": null,
         "errorDescription": null,
         "interfaceCount": "0",
         "lastUpdated": "2020-11-14 17:35:43",
         "lineCardCount": "0",
         "lineCardId": "",
         "locationName": null,
         "managementIpAddress": "10.10.22.66",
         "platformId": "C9300-24UX",
         "reachabilityFailureReason": "",
         "reachabilityStatus": "Reachable",
         "series": "Cisco Catalyst 9300 Series Switches",
         "snmpContact": "",
         "snmpLocation": "",
         "roleSource": "AUTO",
         "type": "Cisco Catalyst 9300 Switch",
         "location": null,
         "role": "ACCESS",
         "instanceUuid": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
         "instanceTenantId": "5dc444d31485c5004c0fb20b",
```

```
            "id": "21335daf-f5a1-4e97-970f-ce4eaec339f6"
        },
        {
            "memorySize": "NA",
            "family": "Switches and Hubs",
            "hostname": "cat_9k_2",
            "macAddress": "f8:7b:20:71:4d:80",
            "serialNumber": "FCW2140L039",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/></status>",
            "deviceSupportLevel": "Supported",
            "collectionStatus": "Managed",
            "apManagerInterfaceIp": "",
            "tagCount": "0",
            "tunnelUdpPort": null,
            "waasDeviceMode": null,
            "upTime": "89 days, 10:02:57.15",
            "lastUpdateTime": 1605375314870,
            "collectionInterval": "Global Default",
            "softwareType": "IOS-XE",
            "softwareVersion": "16.6.4a",
            "associatedWlcIp": "",
            "bootDateTime": "2020-08-17 07:33:14",
            "errorCode": null,
            "errorDescription": null,
            "interfaceCount": "0",
            "lastUpdated": "2020-11-14 17:35:14",
            "lineCardCount": "0",
            "lineCardId": "",
            "locationName": null,
            "managementIpAddress": "10.10.22.70",
            "platformId": "C9300-24UX",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "series": "Cisco Catalyst 9300 Series Switches",
            "snmpContact": "",
            "snmpLocation": "",
            "roleSource": "AUTO",
            "type": "Cisco Catalyst 9300 Switch",
            "location": null,
            "role": "ACCESS",
            "instanceUuid": "3e48558a-237a-4bca-8823-0580b88c6acf",
            "instanceTenantId": "5dc444d31485c5004c0fb20b",
            "id": "3e48558a-237a-4bca-8823-0580b88c6acf"
        }
    ],
    "version": "1.0"
}
```


Na mensagem acima o DNA Center retornou a lista com os detalhes dos dispositivos que estão no Fabric.

Nesse exemplo temos dois switches da família Catalyst 9300.

Vamos estudar em breve como interpretar esses dados recebidos no formato JSON.

Portanto, dessa maneira que a comunicação em REST é realizada, porém podemos coletar e inserir informações, ou seja, fazer tudo o que estudamos relativo ao CRUD.

Utilizamos um exemplo de inserção de dados com o PUT e a geração da chave de autorização e agora utilizamos um GET para buscar a lista de dispositivos.

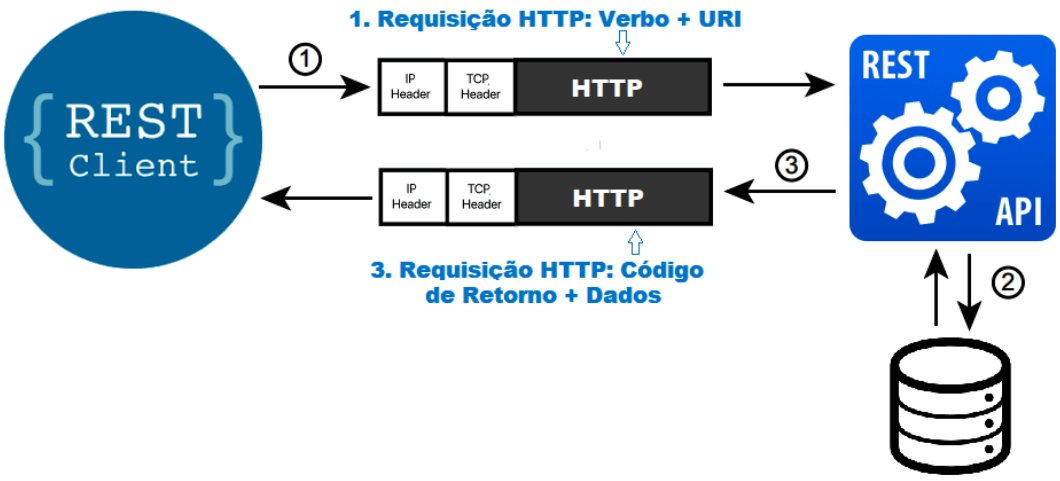
Esses dados podem ser inseridos em um banco de dados e trabalhados por uma API para gerar informações valiosas e necessárias para a operação do sistema.

Você pode navegar pela API do DNA Center disponibilizado no Sandbox e utilizar o Python para criar sua própria API, por exemplo, porém essa é uma tarefa que está além dos objetivos desse curso que é apresentar os conceitos de automação e programabilidade.

5.7 Linguagens de Programação e Serialização de Dados

Basicamente o que estudamos até o momento já nos permite ter uma noção do que é a "Programabilidade" oferecida pelo DNA Center e APIs em REST ou RESTfull APIs.

Ao invés de você entrar na CLI e entrar com comandos isolados, podemos fazer uma interface de comunicação com o DNA Center e fazer uma comunicação direta entre APIs, ou seja, "duas máquinas" conversando entre si.



No lado do cliente (Client API) você pode utilizar diversas linguagens de programação para criar sua própria API ou até mesmo alguns "scripts" para facilitar desde a implantação, operação e troubleshooting do sistema.

Isso é feito normalmente na área de redes (Networking) com linguagens como Python, porém várias outras linguagens de programação suportam a "serialização de dados", tais como PHP e Ruby.

Portanto, você não precisa utilizar a interface gráfica do DNA Center para operar e manter sua rede, você pode ter sua própria aplicação!

Porém, essa troca de informação deve ser estruturada, pois devemos seguir os princípios do CRUD e do REST, por isso mesmo os dados trocados entre o DNA Center e as aplicações estão formatadas em dados serializados, mais especificamente o utilizando o formato JSON.

Serializar os dados (Data Serialization) é um processo utilizado para converter uma estrutura de dados ou um objeto em um formato que possa ser armazenado ou transferido entre aplicações distintas.

Portanto, o processo de serialização é independente da aplicação justamente para permitir que um dado serializado em uma plataforma deve poder ser deserializada por qualquer outra, ou seja, garantindo a comunicação entre aplicações.

Existem vários formatos para serialização de dados, sendo que os mais utilizados em programabilidade de redes são:

- **JSON** ou JavaScript Object Notation
- **XML** ou eXtensible Markup Language
- **YAML** ou YAML Ain't Markup Language

Vamos estudar um pouco dos três principais formatos de serialização de dados a seguir.

5.8 Noções Básicas de XML e YAML

Por muito tempo o XML dominou a web quando o assunto era serialização de dados e resumidamente, o XML serve para definir como um determinado conteúdo vai ser visualizado na tela ou como os dados serão distribuídos, sendo que essa codificação interna é feita pelo uso de marcadores ou tags conforme exemplo a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <response>
      <family>Switches and Hubs</family>
      <hostname>Catalyst_01</hostname>
      <interfaceCount>41</interfaceCount>
      <lineCardCount>2</lineCardCount>
      <macAddress>f8:7b:20:61:10:40</macAddress>
      <managementIpAddress>10.1.1.10</managementIpAddress>
      <role>ACCESS</role>
      <serialNumber>FCW1040L1AK</serialNumber>
      <series>Cisco Catalyst 9300 Series Switches</series>
      <softwareType>IOS-XE</softwareType>
      <softwareVersion>16.6.1</softwareVersion>
      <type>Cisco Catalyst 9300 Switch</type>
      <upTime>20 days, 21:01:11.32</upTime>
 </response>
</root>
```

Por exemplo, se você quiser ler o endereço IP de gerenciamento desse switch basta procurar pela tag "managementIpAddress":

- <managementIpAddress>10.1.1.10</managementIpAddress>

Seu armazenamento é feito em texto, permitindo que a leitura seja feita por diferentes aplicativos.

Já o YAML, acrônimo para YAML Ain't Markup Language, e no início do seu desenvolvimento significava Yet Another Markup Language (Mais outra linguagem de marcação) para distinguir seu propósito centrado em dados no lugar de documentos marcados.

O YAML foi projetado para "ser legível aos seres-humanos", veja exemplo abaixo de arquivo em YAML que pode ser utilizado no Ansible para SSH em dispositivos com Cisco IOS.

```
---
  - name: Configure SSH
    hosts: all
    gather_facts: no
    connection: local

    tasks:
      - debug:
          msg: "The SSH crypto key generate will fail because of netmiko/ntc
timeouts. Comment that out after it becomes no longer necessary."

    # this will fail the first time. When it's run once to get your keys, just
comment it out.
    # coming soon - global_delay_factor will be available, but not yet.
      - ntc_config_command:
          connection: ssh
          platform: "{{ platform }}"
          port: "{{ port }}"
          commands:
            - ip domain-name mycooldomain.net
            - crypto key generate rsa modulus 2048
          host: "{{ inventory_hostname }}"
          username: "{{ username }}"
          password: "{{ password }}"
          #global_delay_factor: 4
```

```
# this will configure ssh parameters
  - ntc_config_command:
      connection: "{{ connection }}"
      platform: "{{ platform }}"
      port: "{{ port }}"
      commands:
        - ip ssh version 2
        - line vty 0 {{ max_vty }}
        - transport input ssh telnet
      host: "{{ inventory_hostname }}"
      username: "{{ username }}"
      password: "{{ password }}"
```

Mesmo sem saber nada de Ansible ou do YAML podemos ler algumas coisas que estudamos dentro desse curso sobre Tópicos de Segurança, por exemplo, analise os comandos que estão abaixo da linha "- ntc_config_command".

Veja que foi definido um nome de domínio (ip domain-name mycooldomain.net), depois criada a chave de criptografia (crypto key generate rsa modulus 2048) e em seguida temos o usuário/senha para acesso SSH (username e password).

Note uma coisa importante aqui, tanto o usuário como a senha são "variáveis" que o Ansible vai ler de outro arquivo de configuração que é definido em outras etapas de configuração.

Vamos estudar mais sobre ferramentas de automação e sobre o JSON na sequência do curso, pois o foco de interpretação de formato é o próprio JSON!

5.9 Interpretando Dados Codificados em JSON

O **JavaScript Object Notation** ou **JSON** tenta equilibrar a legibilidade humana e a linguagem de máquina através regras de notação.

Essas regras permitem que maioria dos humanos consigam ler dados JSON e interpretar as estruturas de dados definidas pelo formato JSON.

Ao mesmo tempo, os dados JSON tornam mais fácil para os programas converterem texto formatado em JSON em variáveis, tornando-o muito útil para a troca de dados entre aplicativos usando APIs.

Você pode tanto ler informações (Read), por exemplo, utilizando um GET, como escrever informações (Write) através de um PUT conforme já estudamos até aqui.

Você precisa saber a sintaxe do JSON para poder escrever corretamente as requisições e também interpretar os dados recebidos pelo API do DNA Center e outros APIs que utilizem o JSON como formato de dados, abaixo segue um resumo do mais importante:

- Utilize chaves ("curly braces" "{}") para delimitar objetos e parênteses ("square brackets" "[]") para delimitar matrizes (arrays).
- Os dados são escritos aos pares de chaves e valores (key/value ou nome/valor).
  Um par "key/value" é formado por uma chave ou key (uma string escrita entre aspas duplas ou "quotation marks" > ""), seguida por dois pontos (colon ":") e seguida de um valor. Exdemplo: "interface":"GibabitEthernet 0/0"
- Cada chave ou key deve ser única.
- Os valores podem ser do tipo string, número (number), objeto (object), matriz (array), boleano (boolean) ou nulo (null).
- Se houver mais de um par "key/value" dentro de um objeto eles devem ser separados com vírgulas (comma).

Você vai encontrar na prática arquivos escritos com as chaves na horizontal ou então linha a linha, porém, não importa como esteja escrito desde que obedeça às regras citadas acima.

Veja exemplos abaixo de representações de diversos tipos de dados em JSON.

1. Representando o ano de 2020

"ano": 2020

Um par nome/valor deve ser representado pelo nome entre aspas duplas, seguido de dois pontos, seguido do valor.

Os valores podem possuir apenas 3 tipos básicos: numérico (inteiro ou real), booleano e string.

Os valores do tipo string devem ser representados entre aspas.

2. Representando um número real

"altura": 1.72

3. Representando uma string

"site": "www.dltec.com.br"

4. Representando um número negativo

"temperatura": -5

5. Representando um valor booleano

"solteiro": true

A partir dos tipos básicos, é possível construir tipos complexos: array e objeto. Como já estudamos arrays são delimitados por colchetes, com seus elementos separados entre vírgulas.

6. Array de Strings

["PR", "SC", "RS"]

7. Matriz de Inteiros

```
[
 [1,2],
 [-2,15],
 [1650,0]
]
```

Os objetos são especificados entre chaves e podem ser compostos por múltiplos pares nome/valor, por arrays e por outros objetos.

Desta forma, um objeto JSON pode representar, virtualmente, qualquer tipo de informação, veja mais no próximo exemplo onde vamos mostrar os dados sobre a personagem Luke Skywalker da série de filmes "Star Wars".

8. Objeto

```
{
 "name": "Luke Skywalker",
 "height": "172",
 "mass": "77",
 "hair_color": "blond",
 "skin_color": "fair",
 "eye_color": "blue",
 "birth_year": "19BBY",
 "films": ["A Vingança dos Sith", "Uma Nova Esperança", "O Império Contra-Ataca", "O Retorno do Jedi", "O Despertar da Força", "Os Últimos Jedi", "A Ascensão Skywalker"],
 "gender": "male"
}
```

Poderíamos também representar o array com o nome "films" no objeto de outra maneira, veja abaixo:

```
{
 "name": "Luke Skywalker",
 "height": "172",
 "mass": "77",
 "hair_color": "blond",
 "skin_color": "fair",
 "eye_color": "blue",
 "birth_year": "19BBY",
 "films": [
        "A Vingança dos Sith",
        "Uma Nova Esperança",
        "O Império Contra-Ataca",
        "O Retorno do Jedi",
        "O Despertar da Força",
        "Os Últimos Jedi",
        "A Ascensão Skywalker"
 ],
 "gender": "male"
}
```

Outra forma de representar os objetos pode ser em apenas uma linha:

- {"cor": "verde", "peso": "50", "unidade": "Kg"}

É possível representar mais de um objeto ou registro de uma só vez. Um exemplo é apresentado a seguir onde dois filmes são representados em um array.

9. Array de objetos

```
[
  {
   "titulo": "Uma Nova Esperança",
   "resumo": "Dezenove anos depois, Luke compra dois dróides, R2-D2 e C-3PO ...",
   "genero": ["aventura", "ação", "ficção"]
  },
  {
   "titulo": "O Império Contra-Ataca",
   "resumo": "Três anos depois, a Aliança Rebelde mudou-se para o...",
   "genero": ["aventura", "ação", "ficção"]
  }
]
```

Outro exemplo importante é a palavra-chave "null", a qual pode ser utilizada para a representação de valores nulos, veja abaixo.

10. Representando um valor nulo

```
"site":null
```

Ainda sobre valores nulos ou não preenchidos você pode encontrar outras notações, isso porque tudo depende como está configurado o banco de dados da API.

Por exemplo, se você listar uma interface no DNA Center que não tem o campo description configurado você terá a seguinte resposta:

- "description": "",

Note que o dispositivo de rede, seja um roteador ou switch, não traz um "null" como resposta e sim duas aspas sem nada dentro, pois o tipo de valor dessa chave é uma string.

O mesmo pode acontecer com um objeto ou array, por exemplo, um objeto vazio pode vir representado como "{}" e um array vazio como "[]".

Por exemplo, veja o item "taskId":{}, ele quer dizer que a key taskId é um objeto, porém não tem nenhum objeto definido dentro dessa chave como valor.

5.10 Exemplos de JSON Recebidos pelo DNA Center

Por exemplo, vamos interpretar arquivo abaixo escrito em JSON pelo DNA Center listando a quantidade de dispositivos conectados no Fabric utilizando a URL:

- "https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device/count"

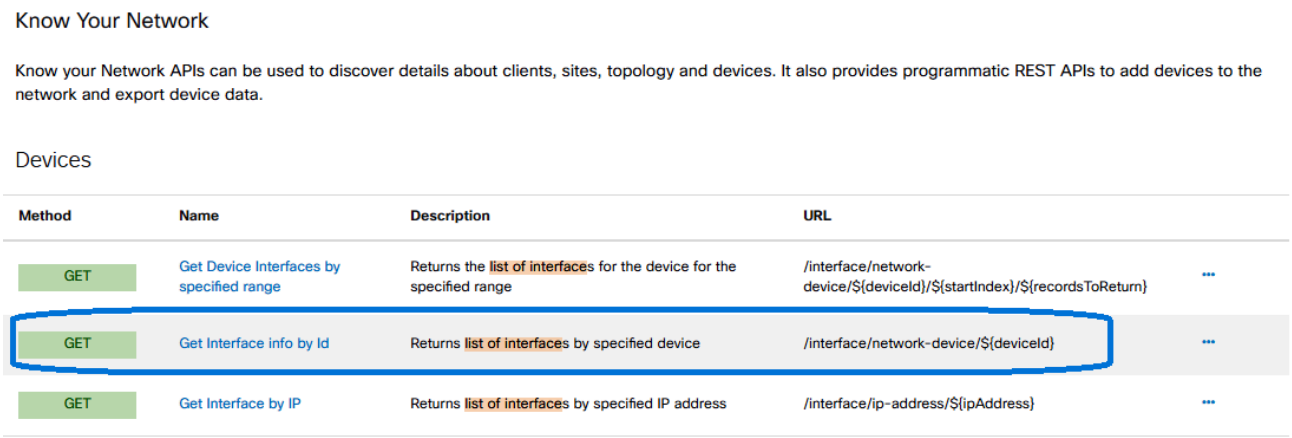
A resposta no momento da coleta utilizando o Sandbox Allways On da Cisco foi a seguinte:

```
{
    "response": 2,
    "version": "1.0"
}
```

Note que o DNA Center nos enviou como resposta apenas um objeto com duas chaves (Keys): response e version, sendo que o valor da chave que traz a quantidade de dispositivos foi o número dois, ou seja, temos dois dispositivos conectados. Já a chave version trouxe o valor "1.0".

Vamos analisar mais um exemplo onde queremos descobrir quantas portas estão UP em um switch conectado ao DNA Center. Para isso, precisamos saber o Device ID, o qual está na lista de dispositivos que extraímos no tópico sobre "Exemplo de Busca de Informações no DNA Center" e verificar como buscar essa informação na API do DNA Center.

Fazendo uma busca na API do DNA Center por "list of interface" encontramos as seguintes opções:



Portando a URL será:

- https://sandboxdnac.cisco.com/dna/intent/api/v1/interface/network-device/21335daf-f5a1-4e97-970f-ce4eaec339f6?status=up

Na URL utilizaremos o Device ID "21335daf-f5a1-4e97-970f-ce4eaec339f6" e a opção "status=up" para procurar apenas valores onde a chave (key) "status" esteja igual a "up", ou seja, "status":"up". Veja a saída abaixo e tente encontrar quantas interfaces estão UP no total:


```
{
    "response": [
        {
            "pid": "C9300-24UX",
            "vlanId": "0",
            "ifIndex": "50",
            "macAddress": "00:00:00:00:00:00",
            "status": "up",
            "adminStatus": "UP",
            "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
            "portName": "Loopback0",
            "mediaType": null,
            "speed": "8000000",
            "duplex": null,
            "interfaceType": "Virtual",
            "ipv4Address": "10.2.2.3",
            "ipv4Mask": "255.255.255.0",
            "isisSupport": "false",
            "mappedPhysicalInterfaceId": null,
            "mappedPhysicalInterfaceName": null,
            "nativeVlanId": null,
            "ospfSupport": "false",
            "portMode": "routed",
            "portType": "Service Module Interface",
            "serialNo": "FCW2136L0AK",
            "voiceVlan": null,
            "lastUpdated": "2020-11-16 17:38:07.565",
            "series": "Cisco Catalyst 9300 Series Switches",
            "description": "",
            "className": "LoopbackInterface",
            "instanceUuid": "4ca556c0-c836-4d42-b3cb-78e04fdfa300",
            "instanceTenantId": "5dc444d31485c5004c0fb20b",
            "id": "4ca556c0-c836-4d42-b3cb-78e04fdfa300"
        },
        {
```

```
        "pid": "C9300-24UX",
        "vlanId": "0",
        "ifIndex": "55",
        "macAddress": "00:00:00:00:00:00",
        "status": "up",
        "adminStatus": "UP",
        "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
        "portName": "Loopback1",
        "mediaType": null,
        "speed": "8000000",
        "duplex": null,
        "interfaceType": "Virtual",
        "ipv4Address": "192.168.11.5",
        "ipv4Mask": "255.255.255.0",
        "isisSupport": "false",
        "mappedPhysicalInterfaceId": null,
        "mappedPhysicalInterfaceName": null,
        "nativeVlanId": null,
        "ospfSupport": "false",
        "portMode": "routed",
        "portType": "Service Module Interface",
        "serialNo": "FCW2136L0AK",
        "voiceVlan": null,
        "lastUpdated": "2020-11-16 17:38:07.565",
        "series": "Cisco Catalyst 9300 Series Switches",
        "description": "",
        "className": "LoopbackInterface",
        "instanceUuid": "772d3d8e-b60a-44ac-a134-51bf0bd8417b",
        "instanceTenantId": "5dc444d31485c5004c0fb20b",
        "id": "772d3d8e-b60a-44ac-a134-51bf0bd8417b"
    },
    {
        "pid": "C9300-24UX",
        "vlanId": "1",
        "ifIndex": "8",
        "macAddress": "f8:7b:20:67:62:81",
        "status": "up",
        "adminStatus": "UP",
        "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
        "portName": "TenGigabitEthernet1/0/1",
        "mediaType": "100/1000/2.5G/5G/10GBaseTX",
        "speed": "1000000",
        "duplex": "FullDuplex",
        "interfaceType": "Physical",
        "ipv4Address": null,
        "ipv4Mask": null,
        "isisSupport": "false",
        "mappedPhysicalInterfaceId": null,
        "mappedPhysicalInterfaceName": null,
        "nativeVlanId": "1",
        "ospfSupport": "false",
        "portMode": "access",
        "portType": "Ethernet Port",
        "serialNo": "FCW2136L0AK",
        "voiceVlan": null,
        "lastUpdated": "2020-11-16 17:38:07.565",
        "series": "Cisco Catalyst 9300 Series Switches",
        "description": "link to host hostA",
        "className": "SwitchPort",
        "instanceUuid": "68e382c9-57f9-44c7-81a0-c7987e11d101",
```

```
        "instanceTenantId": "5dc444d31485c5004c0fb20b",
        "id": "68e382c9-57f9-44c7-81a0-c7987e11d101"
    },
    {
        "pid": "C9300-24UX",
        "vlanId": null,
        "ifIndex": "36",
        "macAddress": "f8:7b:20:67:62:80",
        "status": "up",
        "adminStatus": "UP",
        "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
        "portName": "TenGigabitEthernet1/1/1",
        "mediaType": "SFP-10GBase-CX1",
        "speed": "10000000",
        "duplex": "FullDuplex",
        "interfaceType": "Physical",
        "ipv4Address": "10.10.22.66",
        "ipv4Mask": "255.255.255.252",
        "isisSupport": "false",
        "mappedPhysicalInterfaceId": null,
        "mappedPhysicalInterfaceName": null,
        "nativeVlanId": null,
        "ospfSupport": "true",
        "portMode": "routed",
        "portType": "Ethernet Port",
        "serialNo": "FCW2136L0AK",
        "voiceVlan": null,
        "lastUpdated": "2020-11-16 17:38:07.565",
        "series": "Cisco Catalyst 9300 Series Switches",
        "description": "P2P link cs3850",
        "className": "SwitchPort",
        "instanceUuid": "d7fc6810-1ad7-4502-bb4c-46063909cf78",
        "instanceTenantId": "5dc444d31485c5004c0fb20b",
        "id": "d7fc6810-1ad7-4502-bb4c-46063909cf78"
    },
    {
        "pid": "C9300-24UX",
        "vlanId": "1",
        "ifIndex": "49",
        "macAddress": "f8:7b:20:67:62:c7",
        "status": "up",
        "adminStatus": "UP",
        "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
        "portName": "Vlan1",
        "mediaType": null,
        "speed": "1000000",
        "duplex": null,
        "interfaceType": "Virtual",
        "ipv4Address": "10.10.22.97",
        "ipv4Mask": "255.255.255.240",
        "isisSupport": "false",
        "mappedPhysicalInterfaceId": null,
        "mappedPhysicalInterfaceName": null,
        "nativeVlanId": null,
        "ospfSupport": "true",
        "portMode": "routed",
        "portType": "Ethernet SVI",
        "serialNo": "FCW2136L0AK",
        "voiceVlan": null,
        "lastUpdated": "2020-11-16 17:38:07.565",
```

```
            "series": "Cisco Catalyst 9300 Series Switches",
            "description": "",
            "className": "VLANInterfaceExtended",
            "instanceUuid": "b359bd4d-9f68-4d1b-adfc-eda2f9c45e78",
            "instanceTenantId": "5dc444d31485c5004c0fb20b",
            "id": "b359bd4d-9f68-4d1b-adfc-eda2f9c45e78"
        }
    ],
    "version": "1.0"
}
```


Nós temos aqui um objeto, o qual é um array que dentro dele temos mais cinco objetos, portanto temos cinco interfaces UP nesse switch.

Você conseguiu entender a resposta acima? Parabéns! Você já está dominando o assunto, mas...

... se não conseguiu não se desespere, é mais simples que você pensa.

A primeira chave aberta representa o objeto recebido que tem o nome "response", o qual inicia com um parêntese, por isso sabemos que esse objeto é um array ou uma matriz.

Dentro desse array temos que contar as chaves abertas e fechadas que estão separadas por vírgulas:


```
[ # INÍCIO DO ARRAY
        { # INÍCIO DO OBEJTO
            "pid": "C9300-24UX", # PRIMEIRA CHAVE
            "vlanId": "0",
            "ifIndex": "50",
            "macAddress": "00:00:00:00:00:00",
            "status": "up",
            "adminStatus": "UP",
            "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
            "portName": "Loopback0",
            "mediaType": null,
            "speed": "8000000",
            "duplex": null,
            "interfaceType": "Virtual",
            "ipv4Address": "10.2.2.3",
            "ipv4Mask": "255.255.255.0",
            "isisSupport": "false",
            "mappedPhysicalInterfaceId": null,
            "mappedPhysicalInterfaceName": null,
            "nativeVlanId": null,
            "ospfSupport": "false",
            "portMode": "routed",
            "portType": "Service Module Interface",
            "serialNo": "FCW2136L0AK",
            "voiceVlan": null,
            "lastUpdated": "2020-11-16 17:38:07.565",
            "series": "Cisco Catalyst 9300 Series Switches",
            "description": "",
            "className": "LoopbackInterface",
            "instanceUuid": "4ca556c0-c836-4d42-b3cb-78e04fdfa300",
            "instanceTenantId": "5dc444d31485c5004c0fb20b",
            "id": "4ca556c0-c836-4d42-b3cb-78e04fdfa300" # ÚLTIMA DO CHAVE
        } # FIM DO OBJETO , # VÍRGULA SEPARANDO O SEGUNDO OBJETO
```

Como temos cinco aberturas e fechamentos de chaves dentro do array podemos concluir que temos cinco interfaces, as quais trazem várias informações sobre elas, conforme definido pela API do DNA Center.

Por exemplo, a chave "interfaceType" traz o tipo de interface, parâmetro que poderia ser utilizado para extrair informações específicas sobre portas físicas ou virtuais presentes nesse switch.

Essa é a base da programabilidade, ou seja, utilizar essas informações para extrair ou inserir informações de forma automatizada, sistemática e organizada, eliminando arquivos pessoais e atividades braçais intermináveis.

6 Impacto da Automação no Gerenciamento das Redes

6.1 Introdução

Vamos começar nossa conversa sobre automação com um exemplo...

Suponha que você precisa aplicar políticas de segurança que foram requisitadas pela empresa em todos os dispositivos, ou seja, roteadores, switches, firewalls, APs e demais dispositivos de redes necessitam ser atualizados ou reconfigurados.

Nessa empresa temos 1000 pontos com 5000 dispositivos de rede no total.

Considerando a administração de uma rede tradicional, normalmente você precisará nesse caso elaborar uma ou várias configurações em scripts para aplicar via CLI, ou seja, entrar via SSH ou Telnet para realizar essa configuração.

Normalmente cada tipo de equipamento vai necessitar de uma configuração diferente, pois temos portas diferentes e com várias nomenclaturas, sem contar que se sua empresa tiver vários fabricantes teremos ainda comandos diferentes.

Resumindo: caos total, muito trabalho braçal, muita probabilidade de erro... e por aí vai...

Um dos maiores problemas em uma rede tradicional é o crescimento dos custos de TI para operações de rede, pois como a operação manual torna quase impossível manter uma operação enxuta.

Em redes tradicionais até 95 por cento das alterações de rede são realizadas manualmente, resultando em custos operacionais 2 a 3 vezes maiores do que o custo da rede.

E como já estamos cientes as mudanças manuais levam a erros de configuração e inconsistências na rede.

Além disso, expandir a rede em larga escala normalmente é um processo problemático, assim como a solução de problemas ficam cada vez mais complexa.

Por isso mesmo a necessidade de automatizar processos sempre foi uma necessidade na área de operação de redes.

Esse processo de "**automação de redes**" ou "**network automation**" não é novo, porém nos últimos anos tornou-se uma necessidade latente e tem se desenvolvido de forma muito rápida.

No próximo tópico vamos estudar o que é automação de rede e como ele pode impactar na operação.

6.2 O que é Automação de Rede?

Automação de rede é o processo de automação da configuração, gerenciamento, testes, implantação, operação e manutenção de dispositivos físicos e virtuais em uma rede.

Com tarefas e funções de rede diárias automatizadas e processos repetitivos controlados e gerenciados automaticamente, a disponibilidade do serviço de rede melhora.

A automação de rede é uma metodologia na qual um software (programa ou API) configura, provisiona, gerencia e testa automaticamente os dispositivos de rede.

É usado por empresas e prestadores de serviços para melhorar a eficiência e reduzir erros humanos e despesas operacionais.

As ferramentas de automação de rede oferecem suporte a funções que variam desde o mapeamento básico de rede e descoberta de dispositivo, até fluxos de trabalho mais complexos, por exemplo, tais como gerenciamento de configuração e provisionamento de recursos de rede.

Portanto, podemos gerenciar desde o provisionamento, implantação, alterações e até mesmo a manutenção da rede com automações.

6.3 Tipos de Automação de Rede

A automação pode ser empregada em qualquer tipo de rede, incluindo redes locais (LANs), redes de longa distância (WANs), data center, redes em nuvem e redes sem fio.

Resumindo, qualquer recurso de rede controlado por meio da interface de linha de comando (CLI) ou de uma interface de programação de aplicativo (API) pode ser automatizado.

A automação de rede orientada por script emprega linguagens de script e programação para executar tarefas, de preferência aquelas com estruturas de configuração e procedimentos consistentes.

Linguagens de programação de código aberto como Python e Ruby são as mais utilizadas quando falamos em automação de rede nos dias de hoje.

6.4 Ferramentas de Automação de Rede

Existem várias categorias de interfaces, plataformas e protocolos usados para executar automação de rede baseada em script ou baseada em software.

O CLI é o veículo mais tradicional para implantação de automação de rede.

Embora disponível gratuitamente, testado pelo tempo e altamente personalizável, requer conhecimento na sintaxe CLI.

Uma variedade de ferramentas de código aberto, tais como Ansible, Chef e Puppet, oferecem estruturas de automação de rede.

Essas ferramentas geralmente oferecem uma biblioteca de comandos ou fluxos de trabalho comuns que podem ser facilmente repetidos.

Vamos estudar mais sobre essas ferramentas de automação em breve ainda nesse curso.

6.5 Benefícios da automação de rede

A automação de rede tem três benefícios principais.

- **Maior eficiência**: Ao automatizar funções em dispositivos de rede o tempo de execução das tarefas é drasticamente reduzido.
- **Menor probabilidade de erro humano**: As tarefas manuais estão sujeitas a erros humanos, sendo que ao configurar uma tarefa dentro de uma automação significa que ela só precisa ser inserida corretamente uma vez e repetidamente utilizada da mesma forma.
- **Redução de custos operacionais**: Esse benefício está intimamente ligado aos dois itens anteriores.

Ao eliminar certas tarefas manuais relacionadas ao provisionamento de dispositivos de rede e gerenciamento de rede, as empresas podem operar com maior velocidade e agilidade.

Por exemplo, o provisionamento automatizado pode evitar que um engenheiro de rede tenha que viajar para uma nova filial para "subir a rede", permitindo assim que os funcionários desse local trabalhem mais rapidamente e custos com deslocamento sejam evitados.

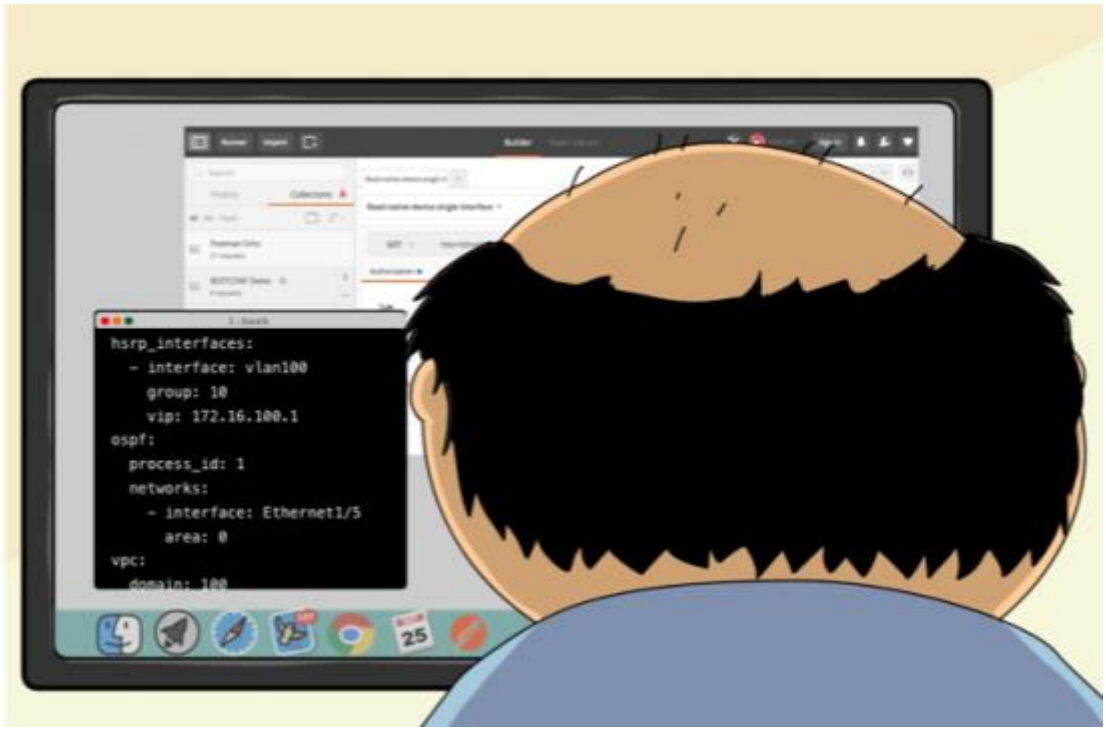
7 Puppet, Chef e Ansible na Automação de Redes

7.1 Introdução

Sendo bem direto, nesse capítulo vamos tratar de como gerenciar as configurações dos dispositivos de rede, principalmente os que não são administrados pelo DNA Center ou então em uma estrutura onde nem exista ainda a solução DNA implementada.

Não é preciso do DNA Center para iniciar a automação da sua rede, principalmente para gerenciar a configuração dos dispositivos ou realizar o "**Configuration Management**".

Esse é o princípio que estão falando cada vez mais em "**Infrastructure as a Code**" ou "**Network as a Code**", ou seja, a rede gerenciada como uma filosofia **DevOps**, que em redes a Cisco chama de **NetDevOps**.



Eu tirei essa figura de um blog da Cisco (fonte: *https://blogs.cisco.com/developer/what-does-network-as-code-mean*) porque quando comecei a estudar o assunto me senti como esse velhinho careca contemplando códigos e a tela do Postman...

... um verdadeiro Dinossauro da TI! (rsrsrs)

Mas, se assim como eu, você ficar assustado no início: "NÃO TENHA MEDO"!

Pois esse é um início de mais uma jornada que vamos trilhar juntos!

Nesse capítulo do curso vamos estudar os principais problemas do gerenciamento tradicional de configurações e como ferramentas de automação como Puppet, Chef e Ansible podem ajudar nessa tarefa.

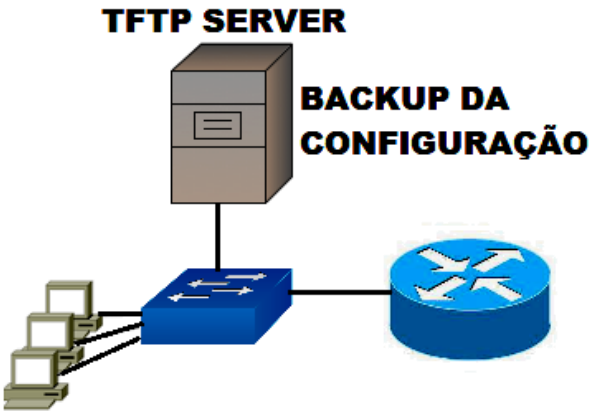
7.2 Desafios no Gerenciamento da Configuração

Já estudamos esse assunto distribuído nesse curso: "*Redes Convencionais e Seus Problemas*".

E, apesar disso, vamos agora listar mais alguns problemas relacionados à configuração...

Em redes convencionais é muito comum que a configuração dos equipamentos de rede esteja dentro das memórias de inicialização ou NVRAM dos dispositivos.

Algumas vezes existe backup (espera-se que exista!) e normalmente está no computador dos administradores de rede... Muito poucas vezes existe um "controle de versão" das configurações e raramente existe "template de configuração".



Outro desafio em redes tradicionais é a alteração em alguma configuração ou parâmetro que precisa ser realizada em diversos equipamentos ao mesmo tempo.

Normalmente esse tipo de alteração da configuração é realizada através do acesso dispositivo a dispositivo e no estilo "copy&paste", ou seja, o responsável pelas alterações copia um script e cola na CLI dos dispositivos de rede que são acessados remotamente via Telnet ou SSH.

Outro problema é chamado de "configuration drift", que é um desvio na configuração padrão estabelecida previamente para um determinado dispositivo.

Por exemplo, existem dois ou mais responsáveis pelas configurações de rede e em um determinado momento uma alteração precisa ser feita para corrigir um determinado problema.

O responsável do dia insere um comando a mais do que o padrão porque com o planejado a interface apresenta uma mensagem de aviso, por isso altera a configuração que está na documentação.

Em redes convencionais isso é muito comum e pode gerar problemas sérios no decorrer do tempo, pois coisas mais graves podem acontecer devido a esses desvios nas configurações.

Você saberia dizer outro problema no gerenciamento das configurações de uma rede? Provavelmente sim, pois essa história é antiga e não morre nos pontos citados acima!

7.2.1 Gerenciamento da Configuração e "Network as a Code"

"**Infrastructure as Code**" (IaC) ou "**Network as a Code**" (NaC) são termos utilizados para definir uma nova maneira de encarar os desafios da configuração em uma Rede.

Isso vem do impacto positivo que o DevOPs e Cloud trouxeram para a administração de ambientes extremamente complexos, implementando automações e processos bem definidos.

Portanto, por que não repetir em ambientes de Rede?

Podemos utilizar três princípios que tratam dos problemas de gerenciamento de configurações de maneira tradicional com essa nova visão de "**Network as a Code**":

1. Store Network Configurations in Source Control

Basicamente isso quer dizer: "**Armazenar as configurações dos dispositivos de rede em um local centralizado (Controle de Origem ou Source Control)**".

Ou seja, trocar o armazenamento local ou até mesmo em servidores TFTP das configurações em texto (.txt), por um local centralizado (chamado controle de origem ou source control).

Além disso, utilizar um controle de origem para as configurações que já forneça controle de versão.

Com isso você já força a implementação de "**templates de configuração**", ou seja, uma configuração padronizada e uniforme para toda a rede.

Um exemplo de repositório é o GitHub, onde as configurações podem ser armazenadas e baixadas, assim como cada alteração pode ser monitorada em detalhes.

2. Source Control is the Single Source of Truth

Com nossas configurações de rede armazenadas em segurança no controle de origem, devemos tratar o controle de origem como a única fonte da verdade sobre como a rede deve ser configurada.

Trazendo isso para nosso mundo de roteadores e switches com Cisco IOS, atualmente a "fonte da verdade" sobre as configurações de rede são as que estão armazenadas nas "running-configs" desses dispositivos.

E da forma que as redes são tradicionalmente administradas, nem sempre a configuração original elaborada na fase de implantação (deploy) é a que temos na memória corrente dos dispositivos, pois não há um controle de versão ou de alteração preciso a ponto de sempre termos documentada a versão mais atual da configuração dos dispositivos de rede.

Com esse controle de configuração podemos sempre garantir que o que temos nos dispositivos de rede é o que também temos dentro do controle de origem, o que facilita e muito resolução de problemas e implementação de novas estruturas de rede.

3. Deploy Configurations with Programmatic APIs

O princípio final da Rede como Código refere-se a como as configurações que estão no controle de origem são realmente aplicadas aos próprios dispositivos de rede.

Conforme mencionado no princípio anterior, a configuração manual por engenheiros de rede pode gerar desvios e erros nas configurações.

O "toque humano" no gerenciamento da configuração da rede está no projeto, desenvolvimento e teste da rede, porém, a implementação real do "código" deve ser feita por meio de APIs, ou seja, automatizando o processo.

Em algumas vezes até mesmo os testes podem ser automatizados!

A seguir vamos estudar três APIs que podem ser utilizados como auxiliares no gerenciamento da configuração, principalmente para automatizar o envio das configurações e busca de informações dos dispositivos de rede.

7.2.2 Templates de Configuração

A configuração de um roteador ou switch via CLI tem base em comandos e suas variáveis de entrada, assim como estudamos os pares "key/value" na interpretação dos dados recebidos no formato JSON, correto?

Por exemplo, para configurar o nome de um roteador precisamos entrar em modo de configuração global e digitar o comando "hostname", seguido de um espaço e um nome:

```
Router(config)#hostname R1
```

Portanto, se quiséssemos criar um template de configuração bastaria utilizar como chave o comando "hostname" e o valor ou variável seria o nome do roteador, o qual poderia ser definido conforme um padrão da empresa (que pode ser automatizado também).

E isso pode ser feito para a configuração completa do dispositivo, sendo que os valores podem ter como origem uma ou mais tabelas ou arquivos padrões que são preenchidos por um ou mais setores da empresa.

Essa fonte de entrada (input) pode ser facilmente convertida em um arquivo em formato JSON ou YAML ou XML e servir como entrada dos valores para o template de configuração do dispositivo de rede.

Por exemplo, um novo escritório remoto de uma empresa vai ser ativado e nele por padrão temos um roteador e um ou mais switches conectados.

Os dados de entrada para a configuração desse novo site, como nome, setores que vão gerar as VLANs, nomes dos usuários para criação de ramais telefônicos, etc... podem ser preenchidos em uma planilha, a qual servirá como input para gerar um template de configuração dos dispositivos de rede.

Evitando que haja trabalho manual na elaboração da configuração e criação de scripts de configuração.

Isso pode ser feito de forma global, por exemplo, gerar a configuração completa de um switch ou então fragmentada, por exemplo, um template de configuração e ativação de uma ou algumas portas de um switch.

7.2.3 Monitoração da Configuração e Controle de Mudanças

Outra facilidade que a implementação da rede como código é acompanhar se as configurações realmente são as que foram planejadas, assim como quais alterações foram realizadas que fogem do padrão.

Isso porque a configuração original do dispositivo, aquele que deve estar em sua "running-config" deve estar armazenada no controle centralizado, facilitando a realização de auditorias de configuração.

Isso também pode ser feito de forma automatizada com as ferramentas que vamos estudar ainda nesse capítulo.

Você pode rodar um teste buscando a configuração corrente e fazendo uma comparação com a configuração que está armazenada no controle central.

As diferenças podem ser avaliadas e você normalmente pode automatizar o processo de padronização através das ferramentas de automação.

7.3 Ferramentas de Automação: Ansible, Puppet e Chef

Todas as três ferramentas emergiram como parte da transição de servidores baseados em hardware para servidores virtualizados, que aumentou o número de servidores e criou a necessidade de automação de software para criar, configurar e remover VMs.

Tanto o Ansible, o Puppet e o Chef são pacotes de software.

Eles possuem versões pagas e diferentes opções gratuitas que permitem que você baixe e aprenda sobre as ferramentas.

Um detalhe importante é que todas as três devem ser instaladas em uma máquina Linux, principalmente o "servidor".

Normalmente as que precisam de um agente instalado no cliente, tais como como puppet e chef, esse agente tem compatibilidade com diversos sistemas operacionais.

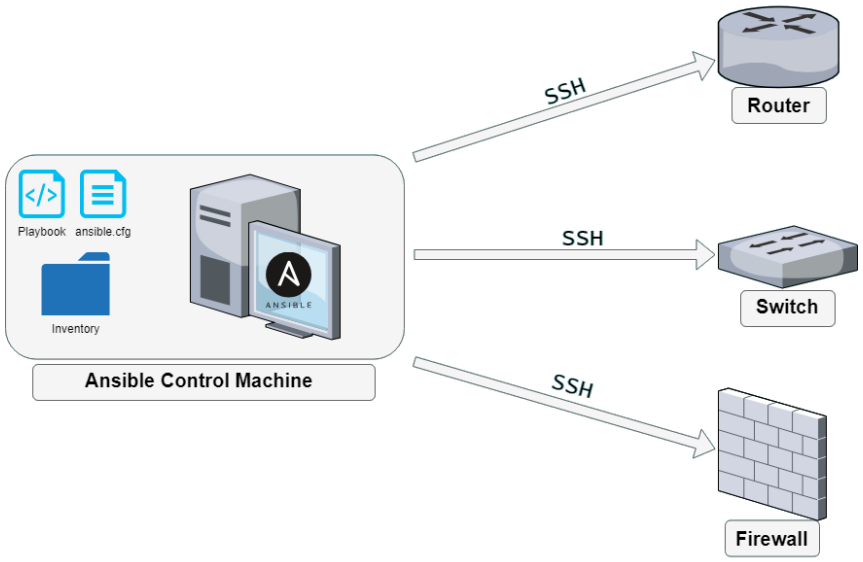
Porém aqui vamos ressaltar o uso dessas ferramentas na automação de Redes ou Network Automation, conforme citado no início, pois eles podem ser utilizados para automação de várias outras áreas da Infraestrutura de TI.

Além disso, vamos estudar também as principais características de cada uma dessas ferramentas de automação.

7.3.1 Ansible

O Ansible é uma ferramenta de automação de código aberto usada para configurar servidores, instalar software e executar uma grande variedade de tarefas de TI a partir de uma localização central.

É um mecanismo sem agente (agentless), onde todas as instruções são executadas a partir de uma máquina de controle (Ansible Control Machine) que se comunica com clientes remotos em SSH, embora outros protocolos também sejam suportados.



Para usar o Ansible você precisa de uma máquina Linux ou uma VM do Linux se seu computador Windows, sendo que você pode baixar o Ansible na página oficial em: "www.ansible.com".

Você pode usar a versão gratuita de código aberto ou usar a versão paga do servidor chamado "**Ansible Tower**".

Uma vez instalado, você cria vários arquivos de texto, como o seguinte:

- **Playbook**: É a forma pelo qual o Ansible consegue configurar uma política ou passos de um processo de configuração. Um playbook é composto por um conjunto de plays, os quais são compostos por um conjunto de tarefas ou tasks. Vamos estudar abaixo melhor o que isso significa.

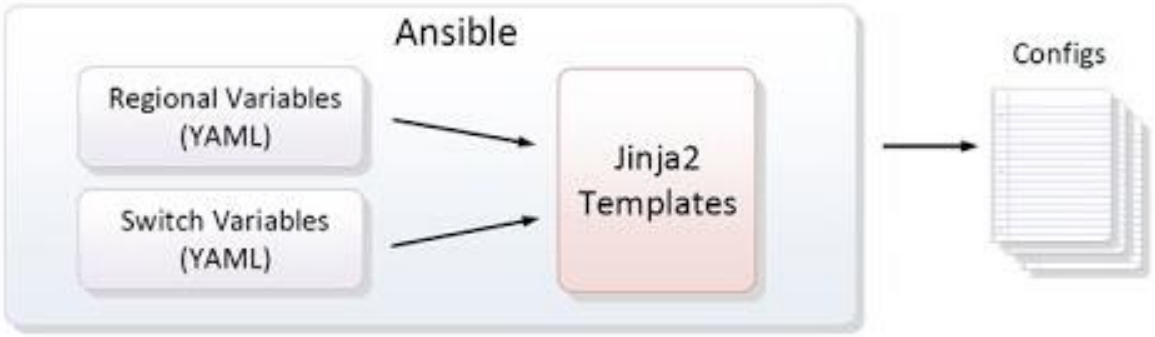
Uma tarefa ou Task é a menor unidade de trabalho, por exemplo, uma task pode ser uma ação como "Instalar um banco de dados", "Instalar um servidor web" ou "configurar o endereço IP em uma interface".

Já um play é composto de tarefas ou tasks, por exemplo, um determinado play pode ser "Preparar um banco de dados para ser usado por um servidor web" é composto das seguintes tarefas:

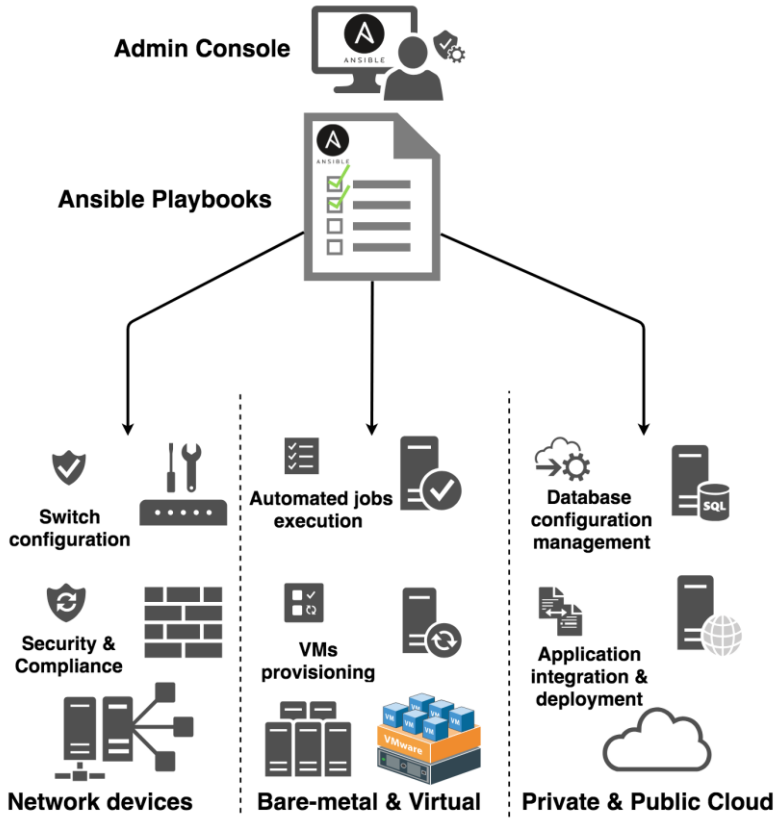
1) Instalar o pacote de banco de dados;
2) Definir uma senha para o administrador do banco de dados;
3) Criar um banco de dados;
4) Definir o acesso ao banco de dados.

Portanto, um exemplo de playbook poderia ser: "Prepare meu site com um backend de banco de dados", e os plays seriam 1) Configurar o servidor de banco de dados; e 2) Configurar o servidor web.

- **Inventory**: Esses arquivos fornecem a lista dos dispositivos e agrupamentos lógicos para configuração, por exemplo, a lista de roteadores, switches e servidores da rede.
- **Variables**: Usando YAML, um arquivo de variável pode listar variáveis que o Ansible substituirá dentro dos templates. Podemos ter variáveis específicas de um dispositivo, assim como variáveis gerais de um grupo.
- **Templates**: Usando a linguagem Jinja2, os templates ou "modelos" representam a configuração de um dispositivo através de variáveis (variables).



Depois de instalar e fazer a configuração básica do Ansible, um engenheiro de redes precisa criar e editar os vários arquivos dele, sendo que ao executar um Playbook é que realmente ele diz ao Ansible para executar os passos definidos da configuração.



Essas etapas podem incluir a configuração de um ou mais dispositivos através dos diversos arquivos configurados, portanto o computador com o Ansible instalado "empurra" (push) a configuração para o dispositivo, por isso mesmo é definido como um "push model".

Como em todas as ferramentas o Ansible pode fazer tanto o provisionamento da configuração quanto a sua monitoração, verificando se a configuração do dispositivo corresponde à configuração ideal que está definida dentro do controle central.

Porém, para fazer o monitoramento de configuração são necessários módulos extras em relação à instalação original do Ansible.

Quanto à forma como o Ansible funciona para gerenciar dispositivos de rede, ele usa uma arquitetura sem agente, ou seja, ele não precisa de nada instalado no dispositivo de rede a ser gerenciado.

Portanto, o Ansible pode acessar os dispositivos de rede utilizando recursos tradicionais de gerenciamento, tais como como SSH ou NETCONF para fazer alterações e extrair informações.

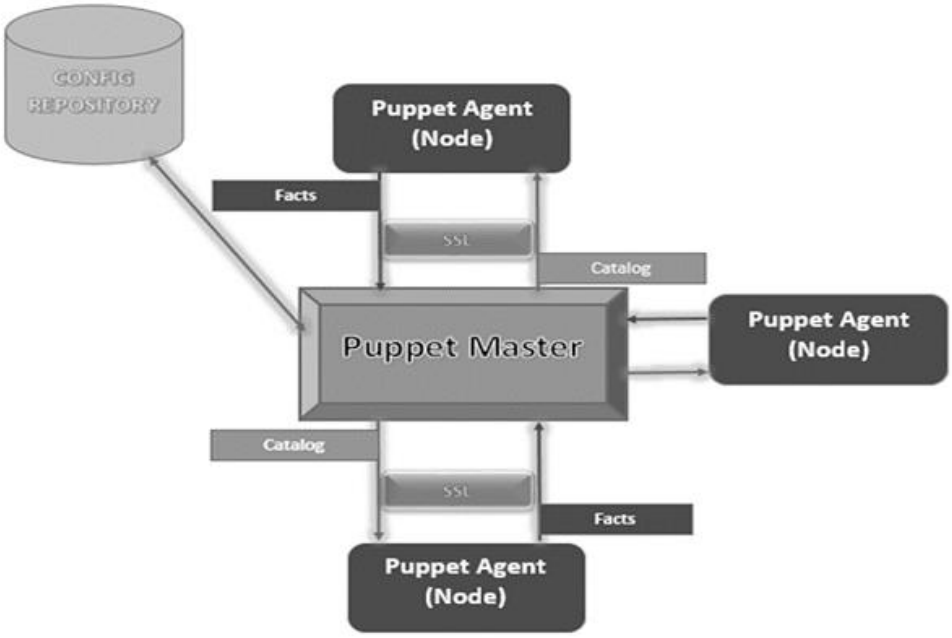
7.3.2 Puppet

O Puppet é uma ferramenta Open Source para gerenciamento de configuração que é utilizada para configurar, gerenciar e implantar principalmente servidores através de códigos.

A ideia central do Puppet é que se tenha a configuração centralizada em um único ponto, e essa configuração seja distribuída para diversos nós de uma rede.

Puppet utiliza uma linguagem declarativa que permite descrever quais configurações os servidores devem ter, através de uma linguagem simples e prática.

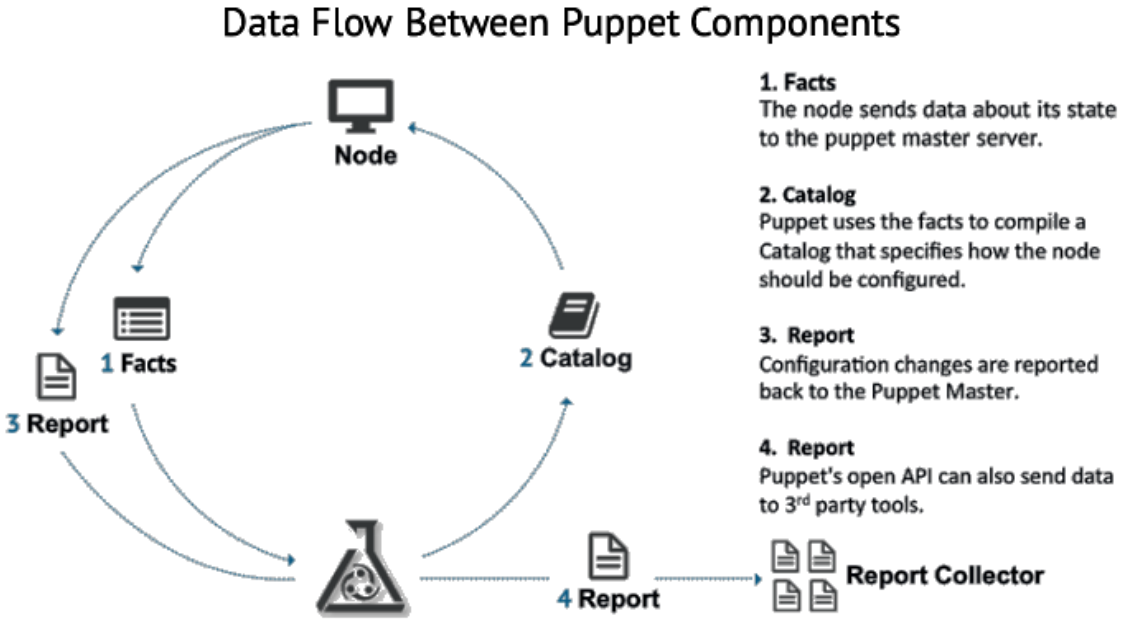
O Puppet geralmente é utilizado no modelo cliente/servidor, sendo que os clientes são chamados de nodes ou nós (dispositivo, servidor físico, máquina virtual onde roda um "Puppet Agent") e o servidor central é chamado de master ou "Puppet Master".



Componentes da arquitetura do Puppet:

- **Puppet Master**: servidor onde o Puppet está instalado. Dentro do servidor temos:
  - **Módulos ou Modules** que são compostos por:
    - **Manifests**: onde ficam os códigos de configuração dos clientes.
    - **Templates**: utiliza a linguagem DSL (Puppet domain-specific language) e permite a criação de modelos de configuração que podem ser preenchidos com as variáveis conforme necessidade de cada ambiente a ser configurado.
    - **Files**: arquivos que serão baixados pelos clientes com as diversas configurações.
  - **CA (Certificate Authority)**: certificado digital utilizado para garantir a segurança da comunicação e autenticar os diversos clientes da rede. Utilizado no SSL.
- **Puppet Client**: software instalado nos clientes que serão controlados pelo Puppet Master.

A execução do Puppet segue os passos abaixo:



- **Fact Collection**: os nodes possuem um agente instalado que permanece em execução e se conecta ao servidor central periodicamente, geralmente de 30 em 30 minutos. O agente envia informações ao servidor central sobre as configurações do nó, como tempo de atividade, sistema operacional, endereço IP, versões dos pacotes, entre outras.
- **Catalog Compilation**: o puppet master usa o **Fact Collection** fornecido pelo agente para compilar as informações sobre como cada nó deve ser configurado. Essas informações compiladas são chamadas de "**catálogo**" (Catalog), sendo que o catálogo é um documento que informa o estado desejado para cada recurso de um nó. O puppet master envia o catálogo para o agente.
- **Report**: cada agente envia um relatório ao puppet master, indicando todas as alterações que foram efetivadas, havendo divergências ou não.
- **Enforcement** : o agente aplica o catálogo nos nós.

É possível também executar o puppet sem a presença de um agente nos nodes, porém nesse cenário, a aplicação do catálogo é agendada na crontab ou disparada via Mcollective.

A codificação do ambiente na linguagem Puppet geralmente estará armazenada em um controle de versão, como o GIT.

No site oficial do Puppet você verá que ele suporta tanto uma arquitetura com agentes como sem agentes (agentless), assim como no caso do Ansible.

Para dispositivos de rede Cisco como roteadores e switches Cisco, os quais não suportam instalação de aplicativos, isso é um fato muito importante, pois eles podem ser gerenciados como um dispositivo externo pelo Puppet Master, mesmo sem um client instalado.

7.3.3 Chef

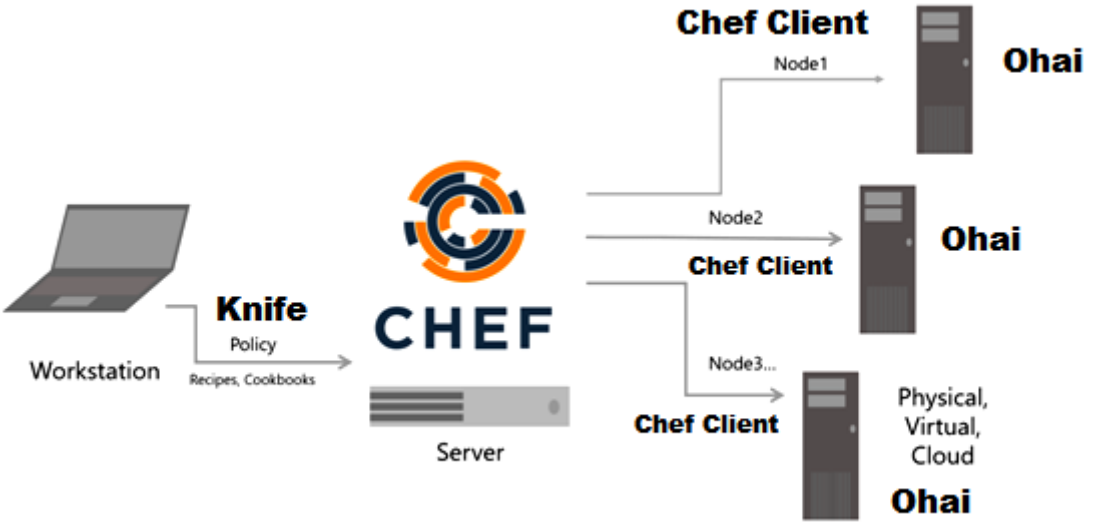
O Chef é uma ferramenta para criação e configuração automática de novos hosts através de um conjunto programável de tarefas, chamadas de as receitas ou Recipies.

Uma **Recipe** é a "receita" que executa ações específicas como instalar um pacote, criar um arquivo etc.

As tais receitas são escritas em uma DSL (Domain Specific Language), ou seja, uma linguagem criada para um determinado propósito, assim como no caso do Puppet.

Tanto o Chef como o Puppet utilizam a estrutura e sintaxe do Ruby para essa tarefa.

A arquitetura do Chef utiliza um "**Chef Server**", o qual é o servidor que possui as receitas e controla toda a estrutura de gerenciamento da configuração.

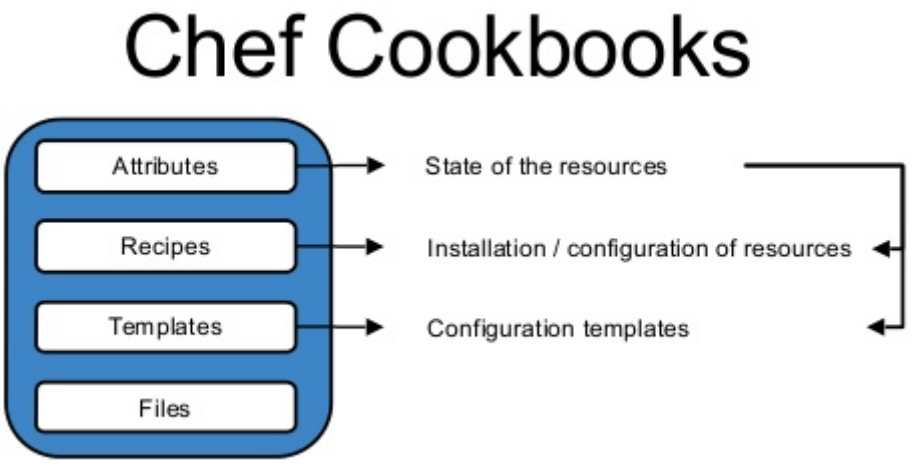


Dentro dos clientes temos um "**Chef Client**", ou seja, software instalado que faz o papel do cliente ou agente que cuida de "aplicar" as receitas nos diversos dispositivos gerenciados pelo Chef Server.

Além disso, você pode encontrar também o **Chef DK** ou Development Kit, o qual permite criar algumas receitas próprias e não apenas executar as já prontas.

Você também deve conhecer alguns conceitos importantes para se entender o Chef:

- **Knife** (faca): é o comando utilizado para comunicação entre as receitas e o servidor, ou seja, a receita é a instrução e a "faca" é o que faz a receita funcionar.
- **Ohai:** verifica o estado dos clientes (nós ou nodes).
- **Resource** (recurso): é algo que descreve um componente da infraestrutura (arquivo, modelo , pacote etc).
- **Recipe** (receita): é um agrupamento de recursos para prover a configuração de um servidor com determinadas características.
- **Cookbook** (livro de receitas): É um agrupamento de receitas.
- **Runlist**: lista de receitas que devem ser aplicadas em uma determinada ordem nos clientes.



Como o Chef requer um software no cliente e muitos dispositivos Cisco não oferecem suporte à instalação de aplicativos, provavelmente você encontrará na prática muito mais ambientes gerenciados pelo Ansible e Puppet quando falamos de gerenciamento de configuração de dispositivos Cisco.

7.3.4 Quadro Comparativo e Nomenclatura

	Ansible	Puppet	Chef
Linguagem	Python + YAML	Ruby	Ruby
Requisitos dos Dispositivos Monitorados	Agentless (sem agente)	Tradicionalmente precisa de um agente instalado	Precisa de um agente instalado
Gerenciamento Centralizado	Controller (qualquer computador pode ser a controladora)	Puppet Master	Chef Server
O que você cria?	Playbook/Roles	Manifest/Modules	Recipe/Cookbook
Comunicação Server/Agent	Não se aplica	TCP 8140 (Agent/status endpoint) TCP 8142 (Orchestrator e botão de "Run Puppet")	TCP 10.000 (Hartbeat) TCP 10.0002 (Comandos) TCP 10.0003 (API default)

8 Conclusão do Curso

Bem pessoal, chegamos ao final de mais um curso!

É muito importante que nesse ponto do curso você tenha domínio dos seguintes itens:

- Como a automação afeta o gerenciamento de uma rede
- A diferença entre as redes tradicionais e redes baseadas em controladoras
- Características das arquiteturas baseadas em controladoras e definidas por software (conceito de overlay, underlay e fabric)
- Separação do plano de controle e plano de dados (Data Plane e Management Plane)
- APIs para norte e sul (North-bound e South-bound)
- Entender a diferença do gerenciamento de dispositivos de forma tradicional e o gerenciamento de dispositivos via Cisco DNA Center
- Características das APIs baseadas em REST (CRUD, verbos HTTP e codificação de dados)
- Reconhecer os recursos dos mecanismos de gerenciamento da configuração utilizando Puppet, Chef e Ansible
- Interpretar dados codificados no formato do JSON

Lembre-se que esse curso conta também com vídeo aulas e questionários extras que estão dentro da trilha do CCNA para quem está se preparando para a prova de certificação ou então quer ter os conhecimentos de um profissional nível associado exigido pela Cisco.