Zero Touch Provisioning (ZTP) para ISPs: Implantação Automatizada e Padronização de Equipamentos de Rede

Pedro Rodrigues

RNP

Educação, Pesquisa
e Inovação em Rede

Quem sou eu?

- Bacharel Interdisciplinar em Ciências e Tecnologia pela UFBA
- Analista de Redes no Ponto de Presença da RNP na Bahia (PoP-BA/RNP)
- Principais atividades no PoP-BA:
  - Conectividade IP de clientes da RNP na Bahia e na Rede Metropolitana de Salvador (REMESSA)
  - Operação de servidores e serviços do PoP-BA

Ementa

- Introdução ao Zero Touch Provisioning (ZTP)
- Protocolos e mecanismos envolvidos no provisionamento automático
- Arquitetura de serviços para provisionamento automatizado
- Fluxo de funcionamento do ZTP
- Estruturação de templates de configuração (Jinja2)
- Padronização de configurações de rede
- Geração automatizada de configurações
- Conceitos básicos de containers
- Vantagens do uso de containers em ambientes de automação de rede
- Integração entre templates e inventário de dispositivos
- Implementação do ambiente de ZTP em containers
- Disponibilização de arquivos de boot e configuração
- Casos práticos de aplicação do ZTP
- Testes e validação do provisionamento

Introdução

- Como provisionamos hoje?
  - Acesso manual aos equipamentos
  - Configuração via CLI
  - Processo repetitivo

Introdução

- Isso ainda faz sentido?
  - Escala?
  - Padronização?
  - Tempo?

Introdução

- O que é Zero Touch Provisioning (ZTP)?
  - Em síntese, provisionamento automático de equipamentos de rede sem intervenção humana.
- Como funciona?
  - Equipamento ligado
  - Obtém rede via DHCP
  - Recebe configuração ou script de provisionamento.
    - Entra em operação
- Por que usar ZTP?
  - Provisionamento rápido
  - Redução de erros
  - Padronização
  - Escalabilidade

Introdução

- Onde isso já é usado?
  - ISP's
  - Redes acadêmicas
  - Redes corporativas
  - Data centers

Introdução

O ponto alto de uso do ZTP é que o equipamento não é configurado, ou seja, um terceiro acessa e o configura, ele se configura.

Como isso funciona?

RFC's relacionadas ao ZTP

O conceito de ZTP não está formalmente definido em uma RFC específica, mas é uma prática amplamente usada na área de redes, suportada por vários padrões e protocolos que podem estar relacionados a RFCs.

- DHCP (RFC 2131 e RFC 8415 para DHCPv6)
  - Usado para atribuir endereços IP e apontar para servidores de configuração ou provisionamento.

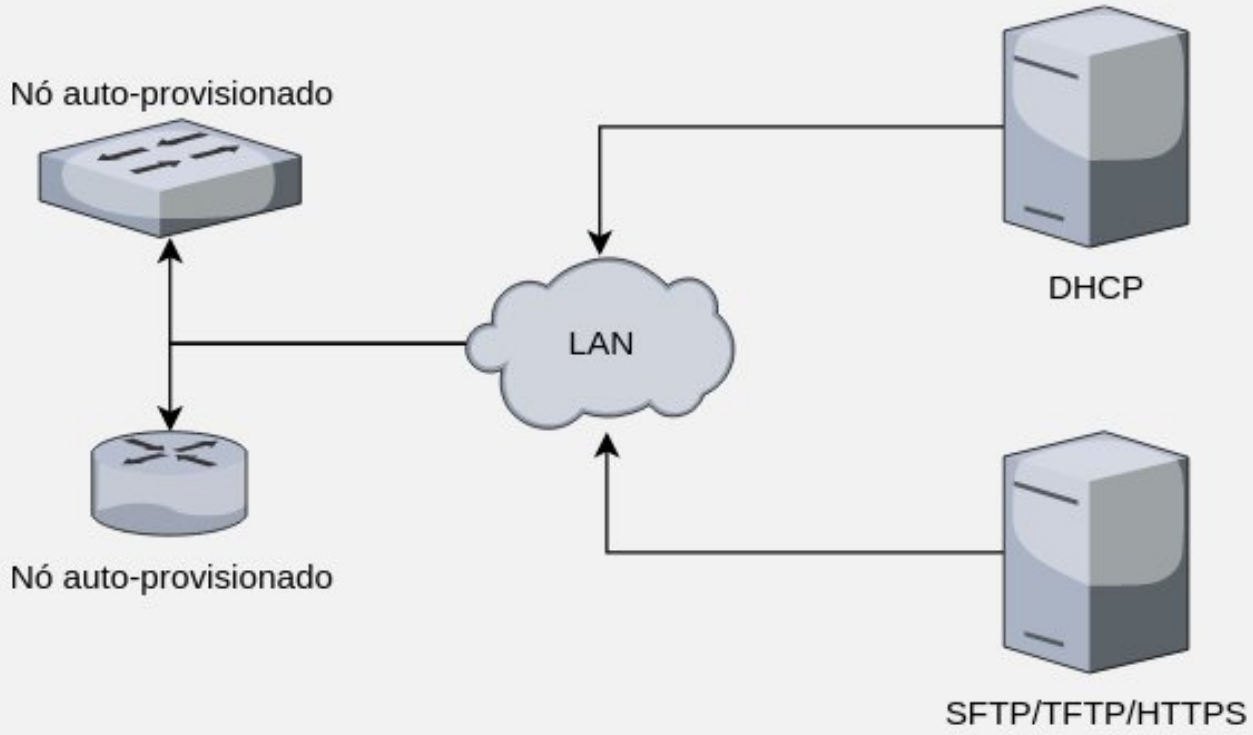
- TFTP (RFC 1350)
  - Permite transferir arquivos de configuração ou firmware para o dispositivo.
- HTTP/HTTPS (RFC 2616, RFC 9110)
  - Cada vez mais utilizado no lugar do TFTP por ser mais seguro e flexível.

- NETCONF (RFC 6241) e RESTCONF (RFC 8040)
  - Usados para provisionamento de configurações em dispositivos que suportam gerenciamento moderno.

- Bootstrap Protocol (BOOTP) - RFC 951
  - Um precursor do DHCP, ainda referenciado em alguns sistemas.

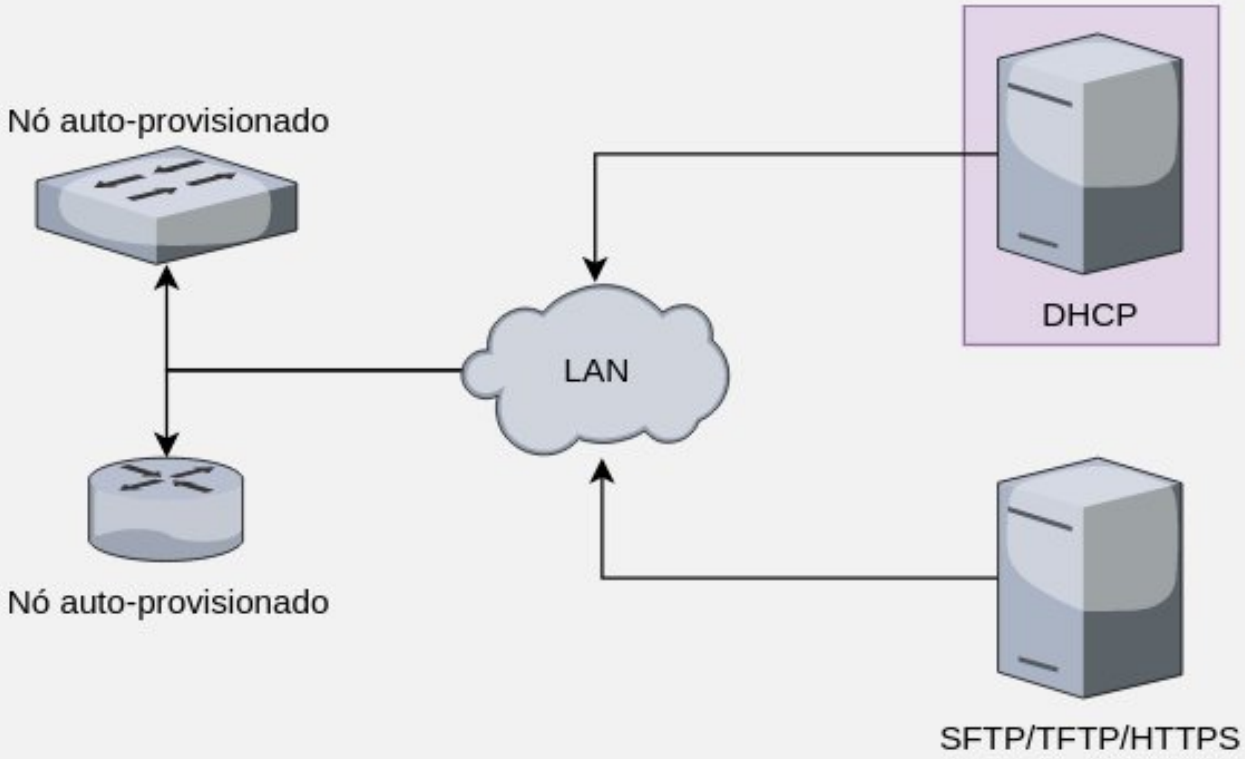
Arquitetura do serviço

- Múltiplos serviços integrados;
- Cada componente possui uma função.



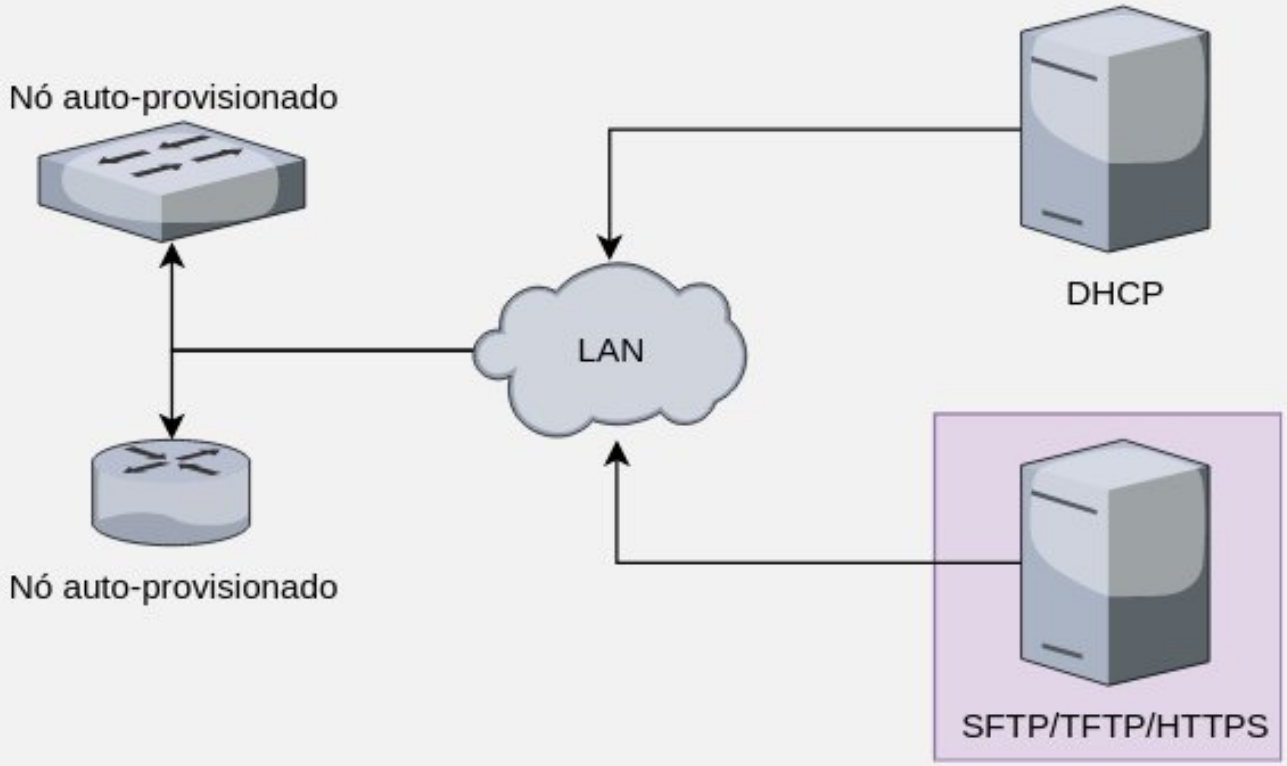
Funcionamento de cada componente

- DHCP
  - Atribui IP;
  - Informa gateway e DNS;
  - Indica onde buscar configuração.



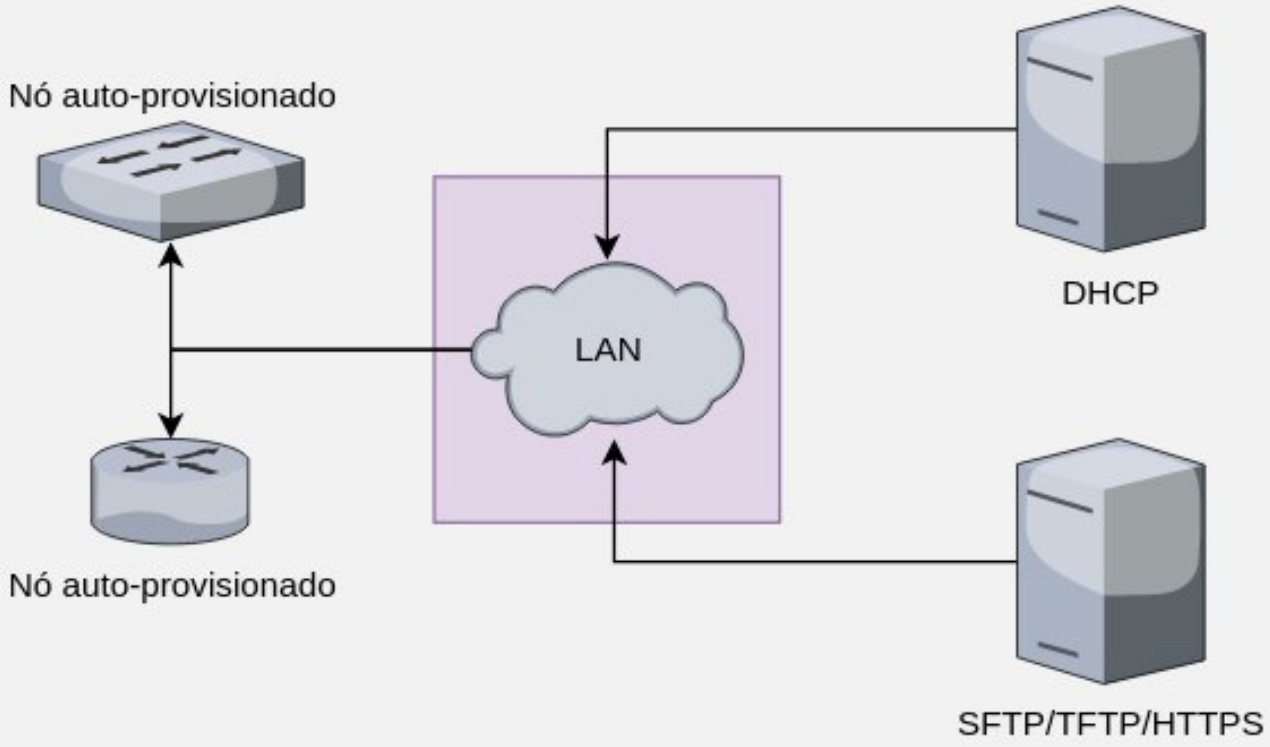
Funcionamento de cada componente

- SFTP/TFTP/HTTPS
  - HTTP ou TFTP
  - Armazena configs/scripts
  - Disponibiliza para download



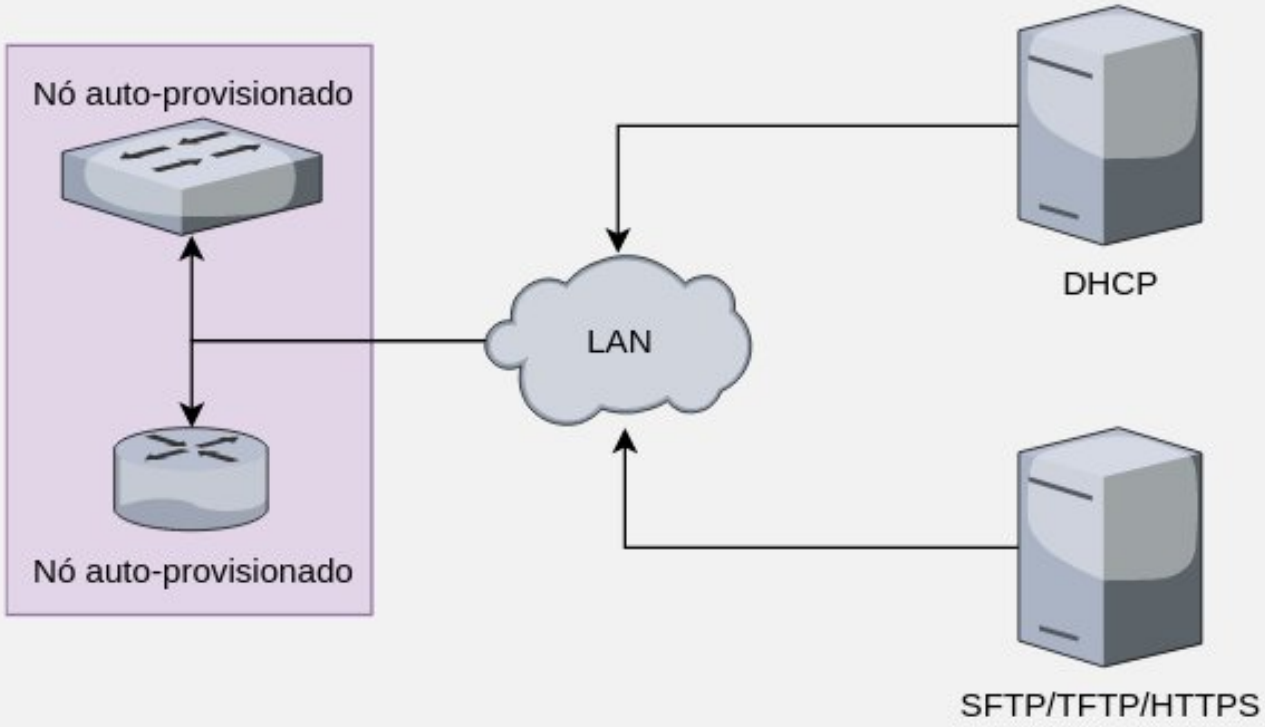
Funcionamento de cada componente

- LAN
  - Transporte em camada 2 até o nó alvo.



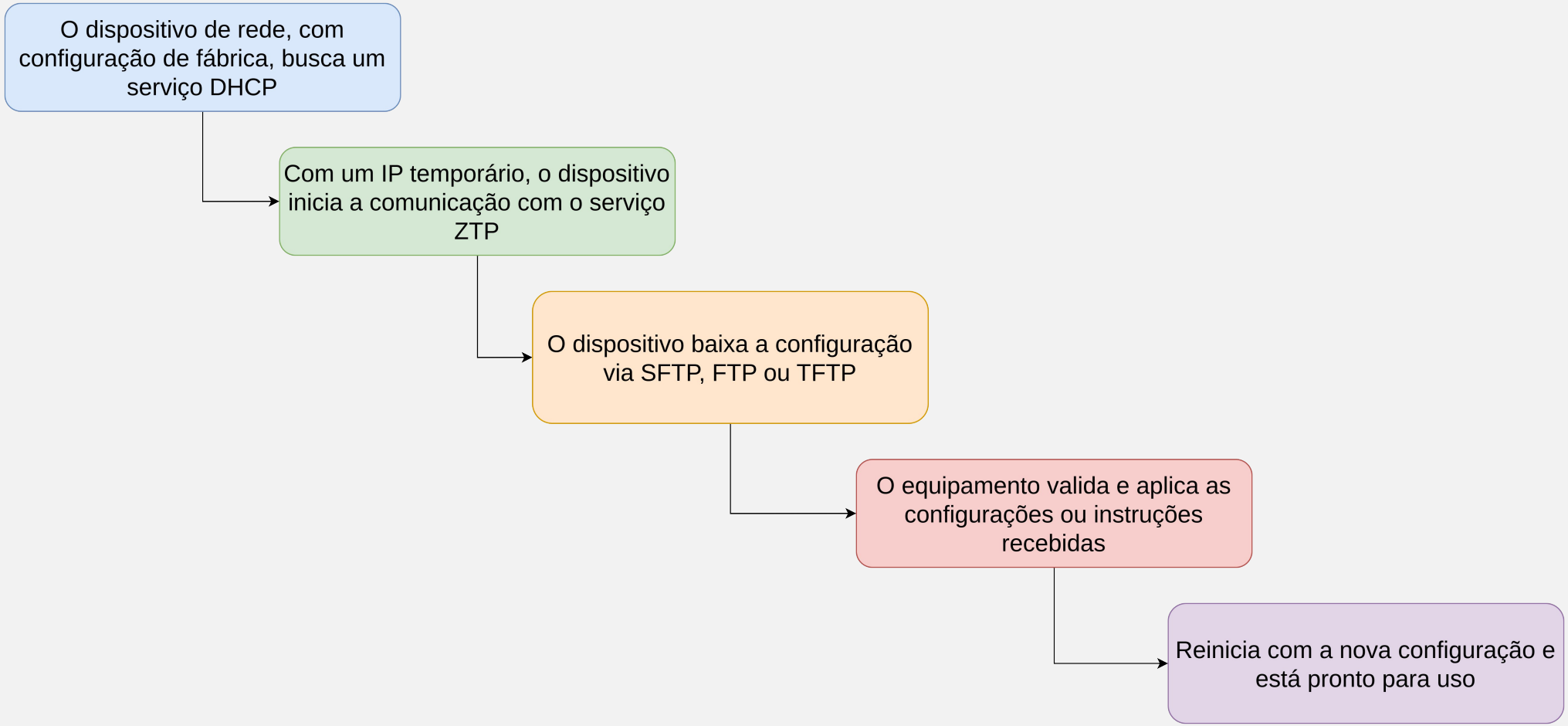
Funcionamento de cada componente

- Nó
  - Inicia sem configuração;
  - Solicita IP via DHCP;
  - Executa instruções;



Recapitulando, como esse fluxo acontece?

Resumo do funcionamento



Quem cria a configuração utilizada?

Arquivos de configuração

- Como garantir a confiabilidade da configuração?
  - Padronização de configurações
    - SNMP, Radius, TACACS, ACL e etc
  - Padronização de nomenclatura
    - Interfaces
    - Hostnames
- Como integrar esses pontos?
  - Templates de configuração

Template de configuração

- Jinja2
  - “Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.”

  - “Jinja é um mecanismo de templates rápido, expressivo e extensível. Marcadores especiais no template permitem escrever código semelhante à sintaxe do Python. Em seguida, o template recebe os dados para renderizar o documento final.”

Template de configuração

- Como funciona os templates em Jinja2?
  - Você cria um modelo (template);
  - Define valores (dados);
  - O **Jinja2 renderiza a configuração final.**
- De maneira análoga:
  - O template pode ser visto como um formulário;
  - Os dados, o preenchimento deste formulário;
  - Resultado, a configuração pronta e padronizada.

Template de configuração

- Ter em mente:
  - Deixamos de escrever a configuração;
  - Escrevemos o **modelo da configuração**.

Sintaxe do Jinja2

- Variáveis
  - Definidas entre chaves {{ exemplo }}
- Estruturas de controle
  - Definidas entre chaves e símbolo de porcentagem {% %}

```
{% if condicao %}
  conteudo
{% endif %}
```

- Estruturas de repetição
  - Definidas entre chaves e símbolo de porcentagem {% %}

```
{% for intf in interfaces %}
interface {{ intf.name }}
 ip address {{ intf.ip }} {{ intf.mask }}
{% endfor %}
```

Dados de entrada

- Fontes possíveis:
  - JSON;
  - YAML;
  - API's;

Quais os benefício de usar os templates?

Considerando que esses itens podem ser padronizado:

- Hostname;
- Endereçamento IP;
- Nomes de interface;
- Serviços habilitados;
- Segurança (ACL, SSH, etc.).

Mantendo estes tópicos uniformes, podemos observar os seguintes benefícios:

- Redução de erros humanos;
- Facilidade de troubleshooting;
- Escalabilidade;
- Governança.

Exemplo de template

```
sysname {{ hostname }}
stelnet server enable
interface {{ loopback.name }}
   ip address {{ loopback_ip }} {{ loopback_mask }}
```

Todos os equipamentos terão a interface loopback configurada e o SSH habilitado.

Observação: Sintaxe utilizada da huawei_vrp, no entanto, o template é agnóstico a sintaxe, é apenas por exemplificação.

Como centralizar os dados e renderizar as configurações?

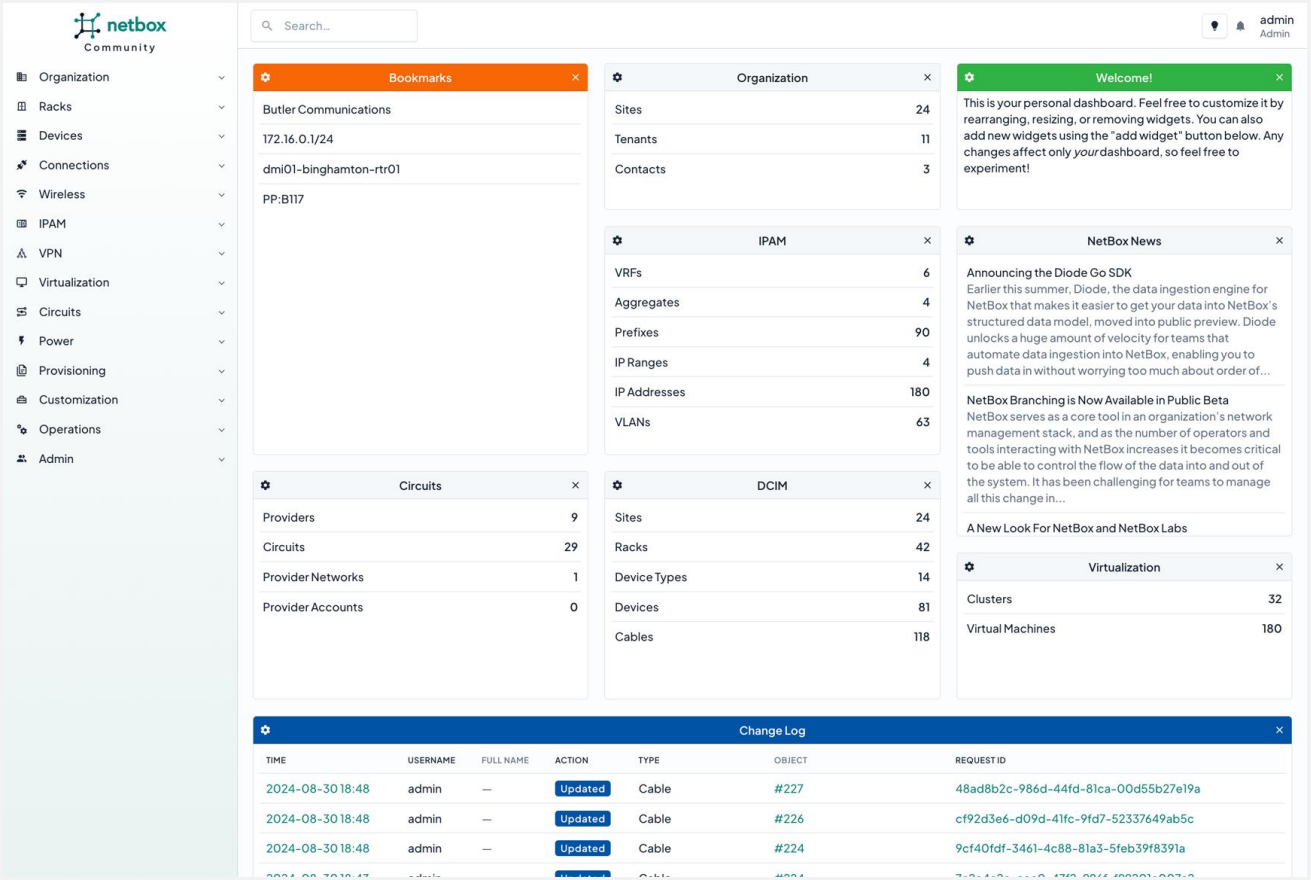
- Nada adianta ter os templates, sem ter uma fonte única da verdade (Single Source of Truth) e dados estruturados.

- Existem algumas alternativas de aplicações que servem essa fonte, dentre as mais famosas, temos o Netbox e o Nautobot (fork do Netbox).

- Neste treinamento, vamos subir uma instância do NetBox em containers Docker para servir os dados e renderizar configurações do dispositivo a ser configurado através do ZTP.

Netbox

- Principais funcionalidades
  - Inventário de rede;
  - Gerenciamento de IP (IPAM);
  - Modelagem de dispositivos;
  - **API para integração.**



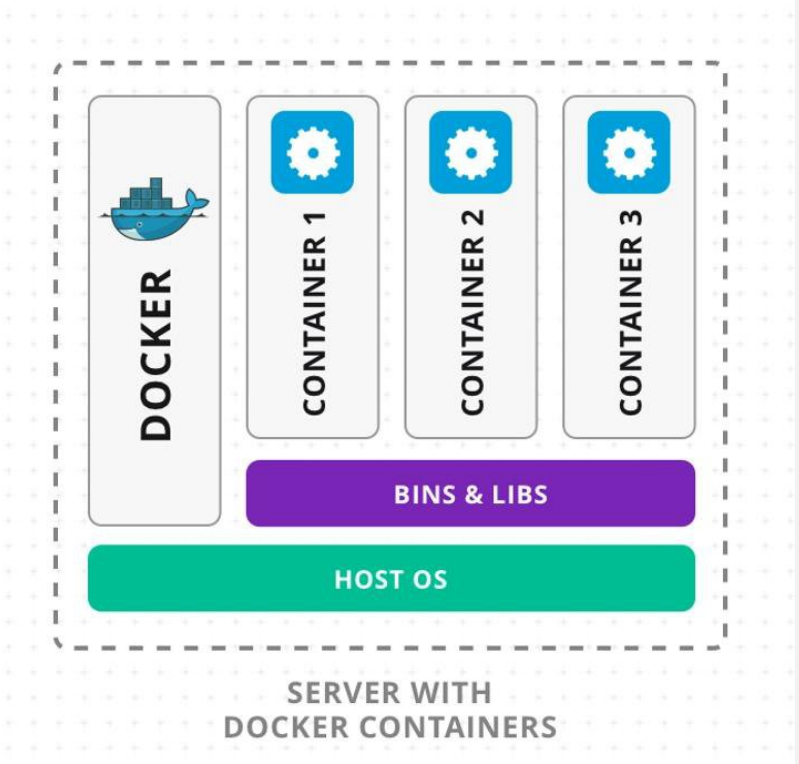
Como manter todos esses serviços de maneira concisa?

Ambiente de automação

- Para a automação, é preciso um ambiente facilmente reprodutível e facilmente recriável.
- Existem diversas maneiras de servir esse ambientes, alternativas possíveis são:
  - Máquinas virtuais (VM's),
    - Libvirt;
    - Vagrant;
    - VirtualBox.
  - Containers
    - Podman
    - Docker
- Neste treinamento, vamos usufruir do **Docker** para prover nosso ambiente, tanto do SSOT (**Netbox**) quanto dos serviços para o funcionamento pleno do ZTP.

O que é um container?

- Um container (ou contêiner) é uma unidade padrão de software que empacota o código de uma aplicação junto com todas as suas dependências (bibliotecas, configurações, binários). Isso garante que a aplicação rode de forma idêntica e isolada em qualquer ambiente de computação.



Fonte: https://hanwenzhang123.medium.com/docker-vs-virtual-machine-vs-kubernetes-overview-389db7de7618

Quais as vantagens de uso dos containers?

- Ambiente reprodutível e consistente
- Rápido de subir e recriar
- Baixo consumo de recursos
- Isolamento entre serviços
- Infraestrutura como código
- Fácil replicação em diferentes ambientes

Hora da prática!

- O que vamos fazer?
  1. Subir uma instância do Netbox como nossa fonte única da verdade.
  2. Utilizar os “configs templates” com Jinja2 para gerar configurações.
  3. Utilizar a configuração gerada em um ambiente de ZTP containerizado para provisionar um dispositivo de rede de exemplo.

Agora que vimos funcionando...

Cabe mencionar que esse é uma estrutura simples de funcionamento do ZTP, esse é o básico necessário para que um dispositivo seja provisionado sem interação humana, mas toda a arquitetura pode ser integrada com mais camadas de automação.

Por exemplo, isso pode ser feito por scripts, pipelines CI/CD ou até mesmo uma ferramenta que esteja na frente e gerencie todos esses processos;

Para tornar mais visual, vou mostrar a ferramenta, Oyá, que está sendo desenvolvida no PoP-BA para servir de interface web e integração desses processos.

Materiais complementares

Docker: https://leanpub.com/dockerparadesenvolvedores
Jinja2: https://jinja.palletsprojects.com/en/stable/
Netbox: https://netboxlabs.com/docs/netbox/

Obrigado!

pedro.rodrigues@pop-ba.rnp.br