



# NMSの第一歩 ~OSSによるLab環境の改善~

JANOG51 2023年1月25日

シスコシステムズ合同会社:嶋 勝也, 東京大学:伊藤 広記, 熊本大学:山田 裕靖

# Agenda

- 0 はじめに
- 1 自己紹介
- 2 ラボ環境の課題・ソリューション調査
- 3 システムの設計・コンセプトデザイン
- 4 システムの紹介
- 5 構築時のトラブル・知見
- 6 Q&A・議論

# 自己紹介



嶋 勝也

シスコシステムズ合同会社

Katsuya Shima (Cisco Systems G.K)



伊藤 広記

東京大学大学院

Koki Ito (The University of Tokyo)



山田 裕靖

熊本大学大学院

Yusei Yamada (Kumamoto University)

はじめに

はじめに

## 本プログラムの背景とお願い

- 本プログラムは2022/8/1~9/30までに開催されたCisco CX Delivery CEインターンシッププログラムの内容です。
- Cisco CX初めての技術エンジニアインターンシップで、フルリモートでの勤務形態である事もあり、足りてない部分が多いですが許してください。
- フルリモートでの技術インターンシップ皆さんどうやってるのかあとでこっそり教えてください

はじめに

# Cisco CX Delivery CE インターンシッププログラム概要

## プログラムコンセプト

1. 技術要素でビジネス課題を解決する方法を考え、実行することを”楽しめる”
2. Cisco (CX Delivery) の文化、環境を”知れる”

### 仮想のお客様（実際にはCX LAB）の課題に対しての解決プロジェクト対応

- お客様課題の提示 -> 課題解決策の検討 -> 解決策の立案と定義 -> 解決策の検証・導入までの対応を通じて、CE業務の一例を体験

### Cisco CX Delivery チームの文化、環境を知るための内容

- Cisco CX Delivery チームを知る機会への参加
- Cisco CX Delivery チームが関係する各部署との連携機会への参加
- 第一線で活躍するCX Delivery エンジニア、他職種メンバーとの交流機会への参加

はじめに

# 業務の流れ

## 準備

- 新たなお客様とのコネクションを作る。
- お客様とコンタクトをとり、ご要望をヒアリングするための接点を創る

## 提案・計画

- お客様の課題を明らかにし、その課題を解決するために最適な提案を行う。

## 設計

- お客様が必要としているニーズを、技術でどのように実現するか検討し、設計におとしこむ。

## 導入

- 設計されたシステムが仕様通りに動作することを確認する。
- お客様からもご確認いただき、システムの妥当性を確認する。

## 運用

- システムを運用し、適宜、ソフトウェアおよびハードウェアのメンテナンスを実施し、システムを最適な状態に保つ。

## 最適化

- パフォーマンスの向上、運用コストの削減、およびその他の継続的な改善含め、システム更新時等にシステムの最適化を行う。

PLAN

Build

Manage

はじめに

# CE職インターン スケジュール 8月

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
	▲ 業務環境キャッチアップ、課題および関連情報ソースの通達、要件確認					
	初日（キックオフ /CX Japan説明）	要件定義手法 レクチャー				
7	8	9	10	11	12	13
	▲ コンセプトデザイン、手法調査（ソリューションの簡単なレクチャー含む）			▲ お盆休み期間		
			チェックポイント			
14	15	16	17	18	19	20
	▲ コンセプトデザイン、手法調査（ソリューションの簡単なレクチャー含む）、設計/導入案整理開始					▲
					チェックポイント・レビュー	
21	22	23	24	25	26	27
	▲ 山田さん、伊藤さんお休み				▲ 設計、導入案の整理	
					チェックポイント・レビュー	
28	29	30	31	1	2	
	▲ 設計、導入案の整理				▲	
				チェックポイント・レビュー		



はじめに

# CE職インターン スケジュール 9月

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1 設計、導入案の整理 チェックポイント・レビュー	2	3
4	5 ラボ環境への導入、動作確認	6	7	8 山田さん、伊藤さんお休み	9 ラボ環境への導入、動作確認 チェックポイント	10
11	12	13	14	15	16 ラボ環境への導入、動作確認 チェックポイント	17
18	19 祝日	20	21	22 実施内容のまとめ チェックポイント	23 祝日	24
25	26 実施内容のまとめ	27	28	29 成果報告	30 クロージング	

TMTオンサイトでの業務  
オフィス/LABツアー

# ラボ環境の課題・ソリューション調査

# ラボ環境の課題

- 機器の動作が正常でないときがある → **機器の状態監視**
  - ラボ内の機器にアクセスできなかったり, 利用時動作が鈍い時がある
  - 100台ほどあるラボネットワーク機器の疎通性に問題があるかないかをすぐに判断できない
  - 想定外のリスタートが発生しても気が付かない
- IPアドレスの管理が面倒 → **機器の状態監視**
  - 管理から漏れているものもあり, 割り当て前にpingを打ち使用可能か否かを確認する必要がある
- 前回の続きの作業をすぐに行いたい → **Configの管理**
  - 設定内容を変更する機会が多いが, 過去にさかのぼって設定ファイルを戻したいときがある

# CISCO製品

- Cisco DNA Center

- ダッシュボードでの一元管理が可能、機能要件の大部分を満足
- ポリシー管理や自動化、高度なセキュリティは不要
- 各機器の設定は中央からではなく、個々で行いたい
- 専用HWが必要

- Cisco FindIT Network Manager / Network Discovery Utility

- 対応機種が限定的 ex. Cisco100-500, Cisco Small Business 200/300

- Cisco Evolved Programmable Network Manager(EPNM)

- 機器の状態監視、config管理を行うことができる
- GitLabへのpushができない
- VM要件：“16vCPU および 64G RAM、 2.8TB” (version 6.0)



OSSでシステムを構築, コンテナサポートを重視

# 機器の状態監視

- 主なネットワーク機器監視ツール(OSS)
  - 疎通性の確認やSNMPでの監視などの基本的な機能であればどのOSSでも実現可能

	ping/traceroute の手動実行	収集した情報 の可視化	通知	ホストの 自動検出	コンテナ サポート
Zabbix	○	○	○	○	○
Nagios	×(有償版○)	○	○	△(有償版○)	×
Cacti	△	○	○	○	×
Prometheus + Grafana	×	○	○	×	○



機器の状態監視にはZabbixを使用

# Config管理

- 主なConfig管理ツール(OSS)
  - rConfig, NeDiは旧バージョンのみがOSSとして公開されている

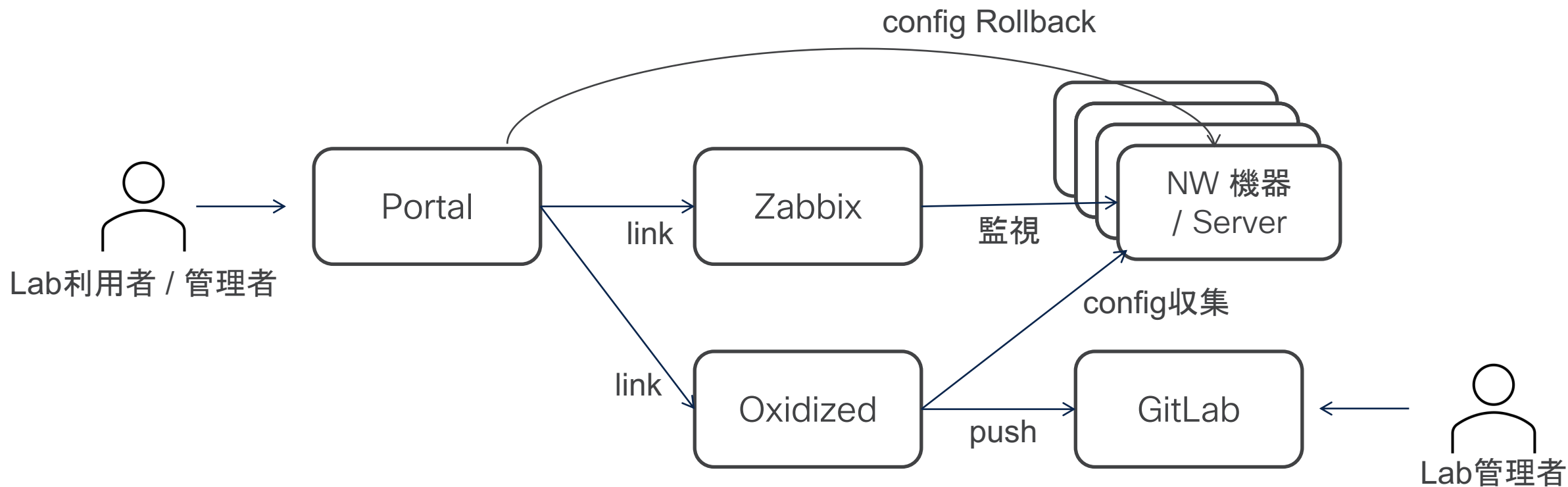
	GitLab連携	GUIでの表示	ロールバック	コンテナサポート
Oxidized	○	○	×	○
RANCID	○	○(Viewvcと連携)	×	×
rConfig	×	○	×(有償版○)	×
NeDi	×	△(差分表示不可)	○	○

 Configの管理にはOxidizedを使用, ロールバックはAnsibleを用いて実装

# システムの設計・コンセプトデザイン

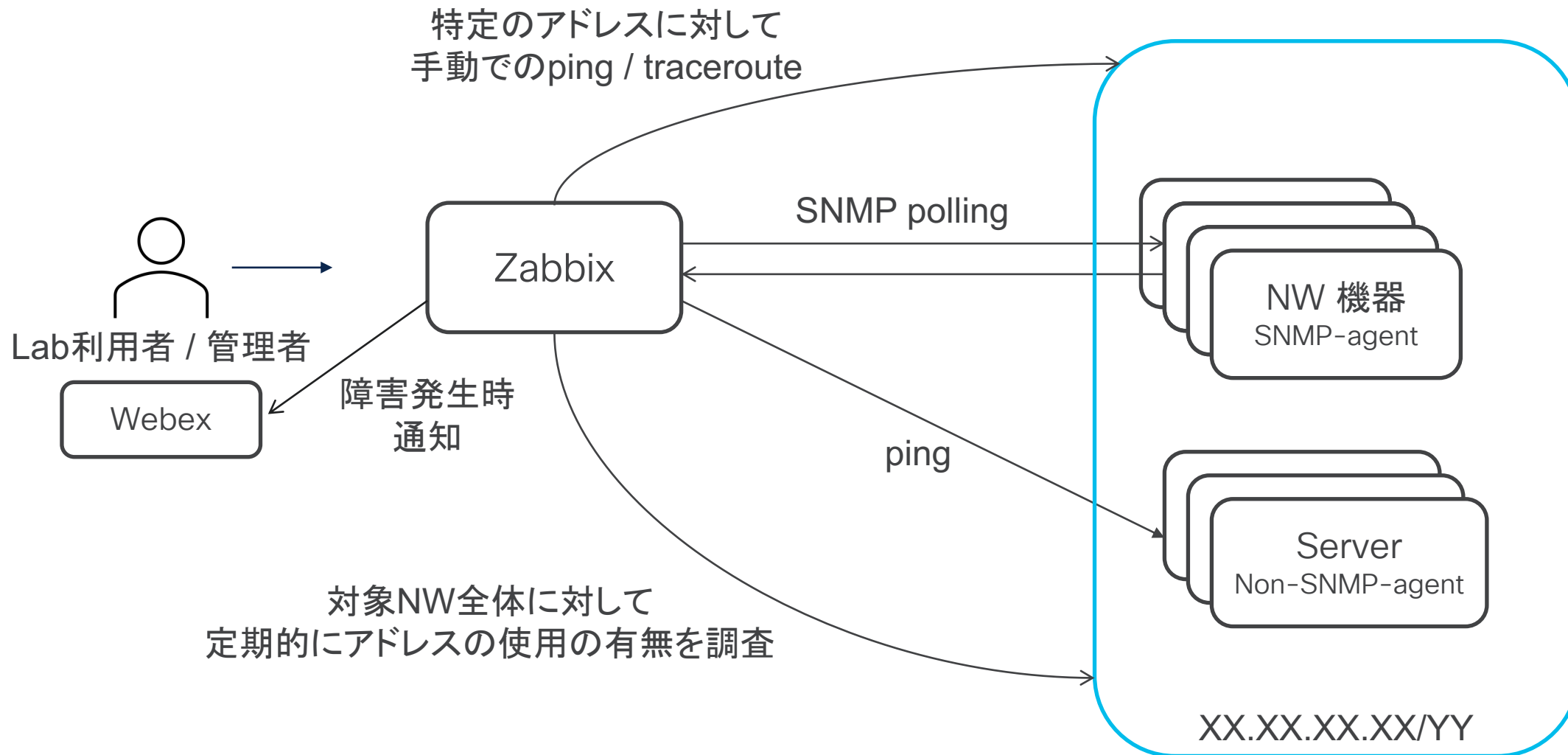
# 設計の概要

- Portal, Zabbix, Oxidized, GitLabの4つで構成





# 機器の状態監視

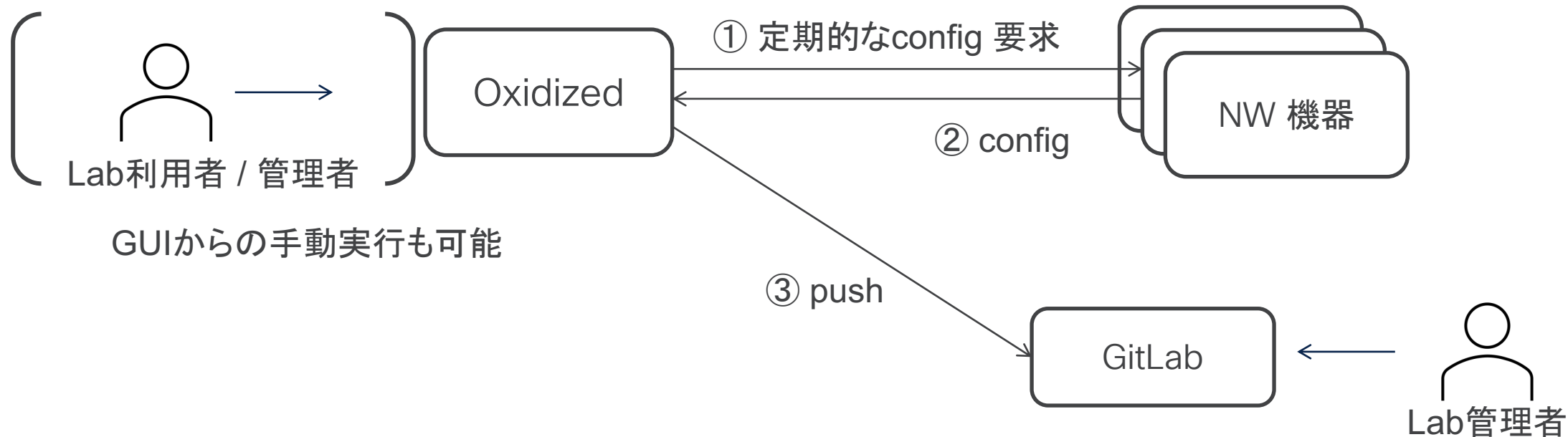


# Config管理

- collect

収集時に必要な情報

- 対象機器のIPアドレス
- login ユーザ名 / パスワード
- enable パスワード

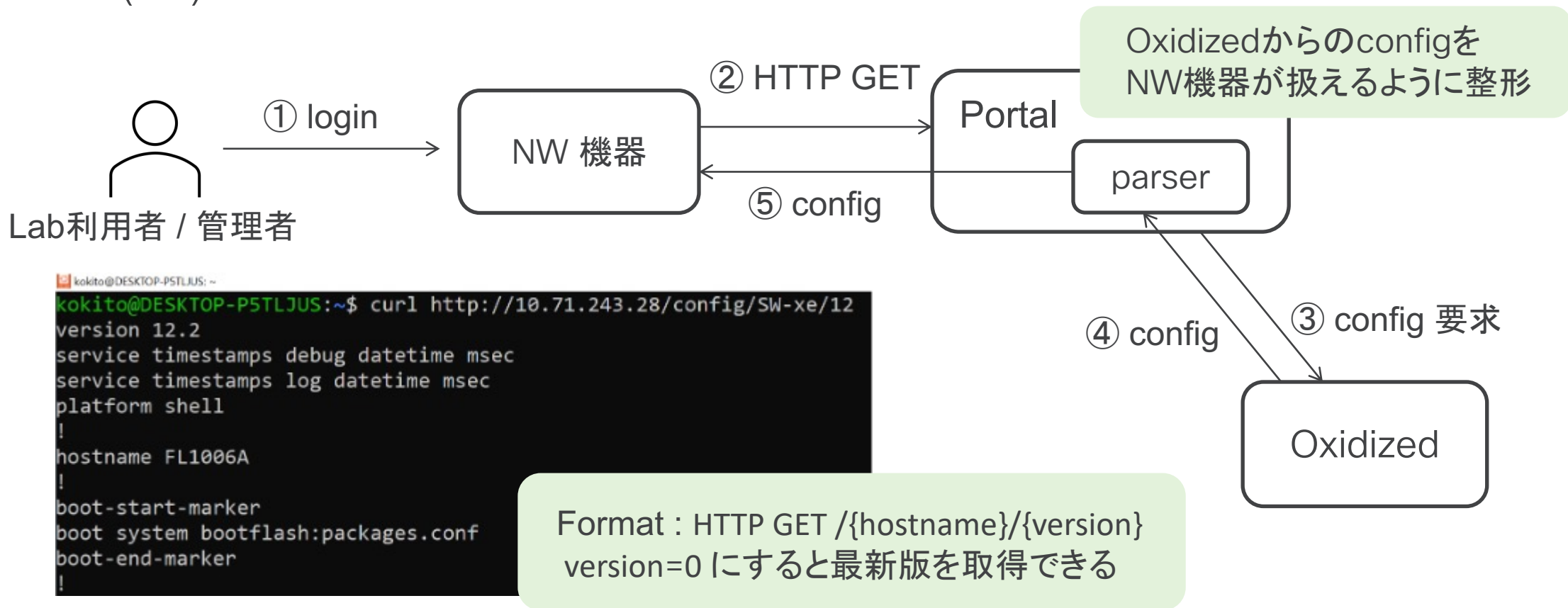


GUIからの手動実行も可能

Config自体はoxidizedで保存・管理されているが  
バックアップのためにGitLabにも保存

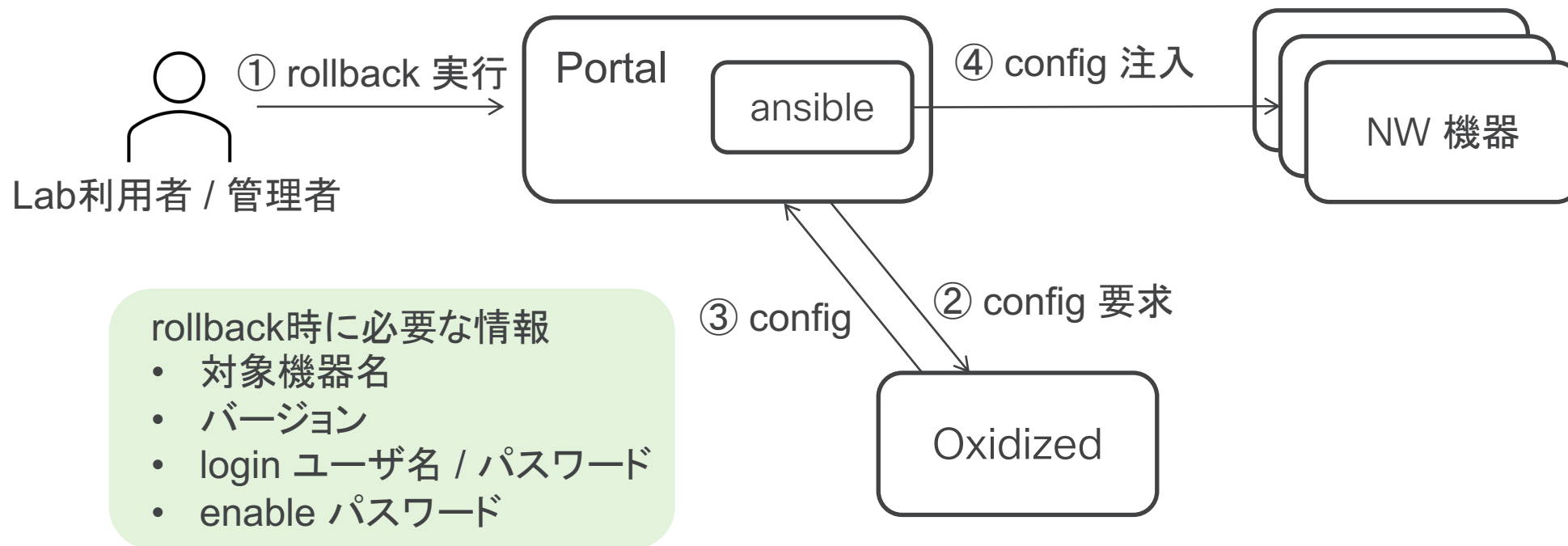
# Config管理

- Rollback (CLI)



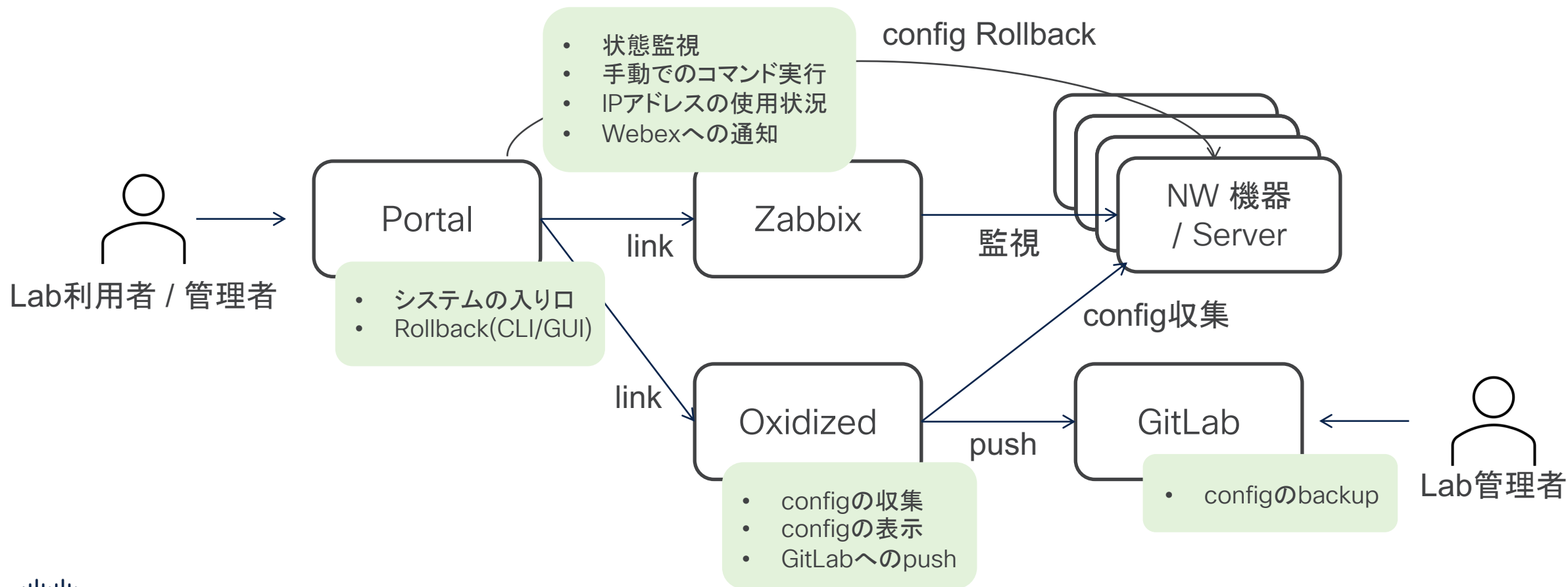
# Config管理

- Rollback (GUI)



# 設計の概要

- Portal, Zabbix, Oxidized, GitLabの4つで構成



# システムの紹介

# サーバスペック / 監視対象

- サーバ
  - 4 vCPU
  - メモリ 16GB
  - ストレージ 64GB
- 監視対象
  - /24のネットワークを2つ
  - 実際稼働しているホストは100 ~ 200程度



## localhost.localdomain


Version: 6.7.0 (Build 9484548)  
State: Normal (not connected to any vCenter Server)  
Uptime: 134.41 days

▼ Hardware	
Manufacturer	Cisco Systems Inc
Model	UCSC-C220-M5SX
▶ CPU	36 CPUs x Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz
Memory	382.66 GB
Persistent Memory	0 B
▶ Virtual flash	0 B used, 0 B capacity

▼ Hardware Configuration	
▶ CPU	4 vCPUs
Memory	16 GB
▶ Hard disk 1	64 GB
USB controller	USB 2.0
▶ Network adapter 1	VM Network (Connected)
▶ Video card	4 MB
▶ CD/DVD drive 1	ISO [datastore1] intern/ubuntu-22.04.1-live-server-amd64.iso <a href="#">Select disc image</a>
▶ Others	Additional Hardware

# Portal

Welcome to Laboratory!!!



**Zabbix**  
ネットワークの状態監視

Oxidized

**Oxidized**  
コンフィグ管理

Hostname

Version

**check**

**Rollback**  
コンフィグロールバック

Copyright © 2022 Cisco Systems, Inc.



# Zabbix

The screenshot displays the Zabbix web interface. On the left is a navigation sidebar with categories like Monitoring, Services, Inventory, Reports, Configuration, and Administration. The main area shows a table of hosts. One host, 10.71.243.41, is highlighted in blue. An overlay window shows a detailed message for this host, indicating a problem: 'Unavailable by ICMP ping'. The message includes the start time (2022.09.21 13:21:02), problem name, host IP (10.71.243.41), severity (High), and operational data (Down (0)).

Name	Interface	Availability	Tags	Status	Latest data	Problems	Graphs
10.71.243.1	10.71.243.1:10050	ZBX SNMP	class: network target: cisco target: cisco-ios	Enabled	Latest data 21	Problems	Graphs 4
10.71.243.2	10.71.243.2:10050	ZBX		Enabled	Latest data 3	1	Graphs
10.71.243.3	10.71.243.3:10050	ZBX		Enabled	Latest data 3	Problems	Graphs
10.71.243.4	10.71.243.4:10050	ZBX		Enabled	Latest data 3	Problems	Graphs
10.71.243.5	10.71.243.5:10050	ZBX		Enabled	Latest data 3	Problems	Graphs
10.71.243.7	10.71.243.7:10050	ZBX					
10.71.243.8	10.71.243.8:10050	ZBX					
10.71.243.9	10.71.243.9:10050	ZBX					
10.71.243.10	10.71.243.10:10050	ZBX					
10.71.243.14	10.71.243.14:10050	ZBX					
10.71.243.15	10.71.243.15:10050	ZBX					
10.71.243.16	10.71.243.16:10050	ZBX					
10.71.243.17	10.71.243.17:10050	ZBX					
10.71.243.18	10.71.243.18:10050	ZBX					
10.71.243.20	10.71.243.20:10050	ZBX					
10.71.243.21	10.71.243.21:10050	ZBX SNMP	cla				

**Messages** People (3) Content Meetings +Apps

zabbix 1:26 PM

To: **Koki Ito**

Subject: Problem: Unavailable by ICMP ping

Problem started at 13:21:02 on 2022.09.21

Problem name: Unavailable by ICMP ping

Host: [10.71.243.41](#)

Severity: High

Operational data: Down (0)

Original problem ID: 138989

# Oxidized

Oxidized Stats Migration

nodes /

Show  entries

Show / hide columns Refresh Reload

Name	Model	Group	Last Status	Last Update	Last Changed	Actions
SW-ios	IOS	default	<span style="color: green;">■</span>	2022-09-21 09:18:21 JST	unknown	
SW-xe	IOS	default	<span style="color: green;">■</span>	2022-09-21 09:20:44 JST	unknown	

nodes / Versions for Node

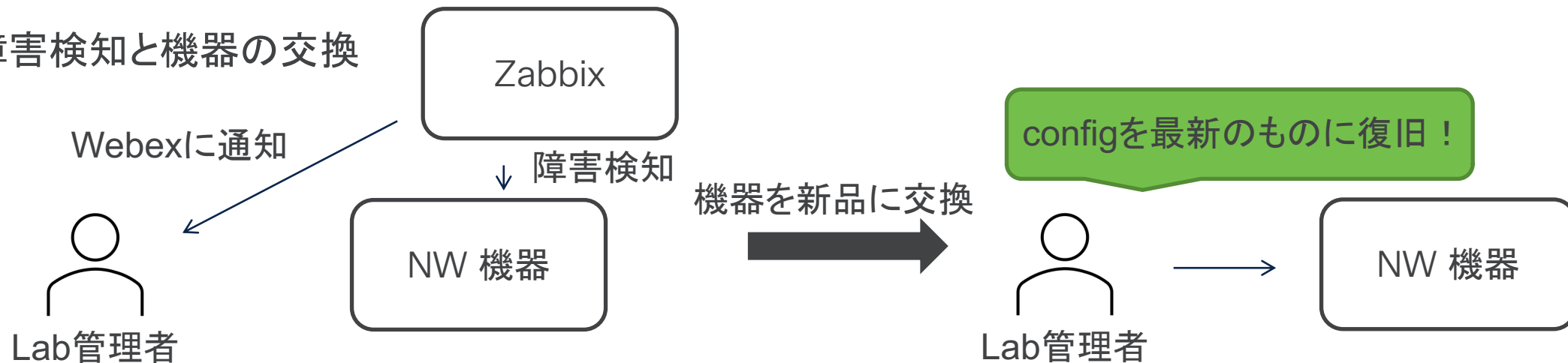
Show  entries

Show / hide columns Refresh

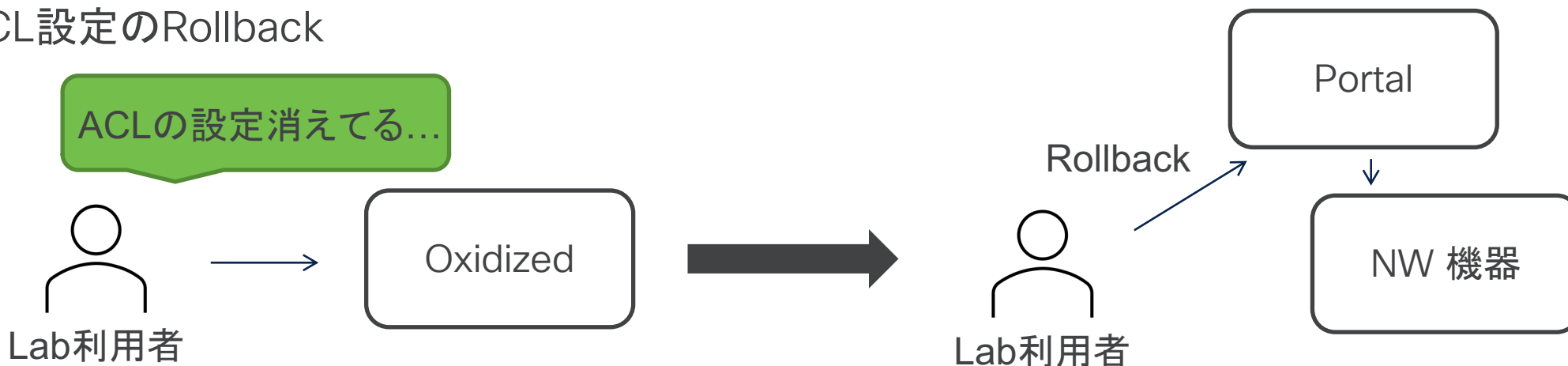
Version	Dates	Actions
25	4 days 19 hours ago	
24	4 days 19 hours ago	
23	4 days 19 hours ago	
22	4 days 19 hours ago	
21	4 days 20 hours ago	

# デモ

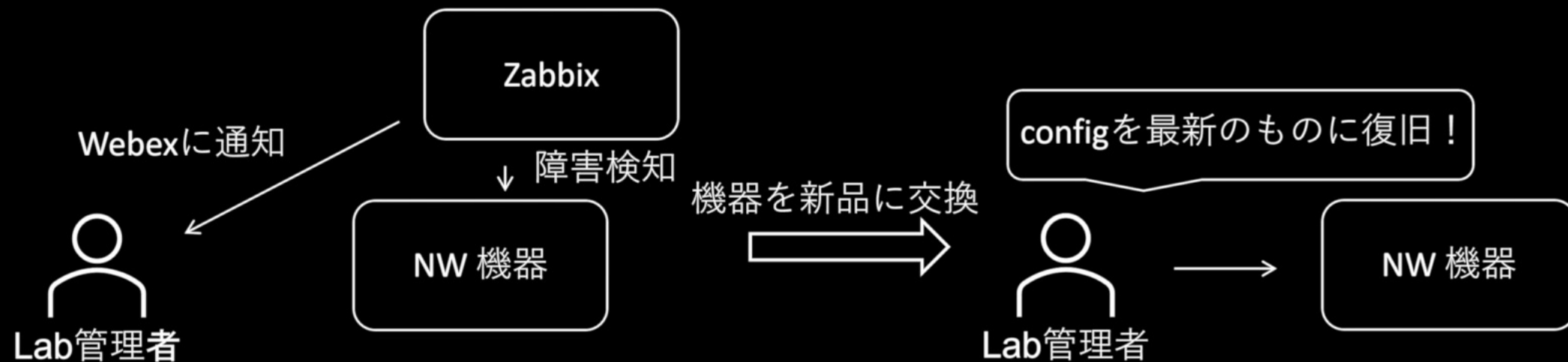
## 1. 障害検知と機器の交換



## 2. ACL設定のRollback



# 1. 障害検知と機器の交換



# NMSの導入により実現できたこと

- 機器の状態監視

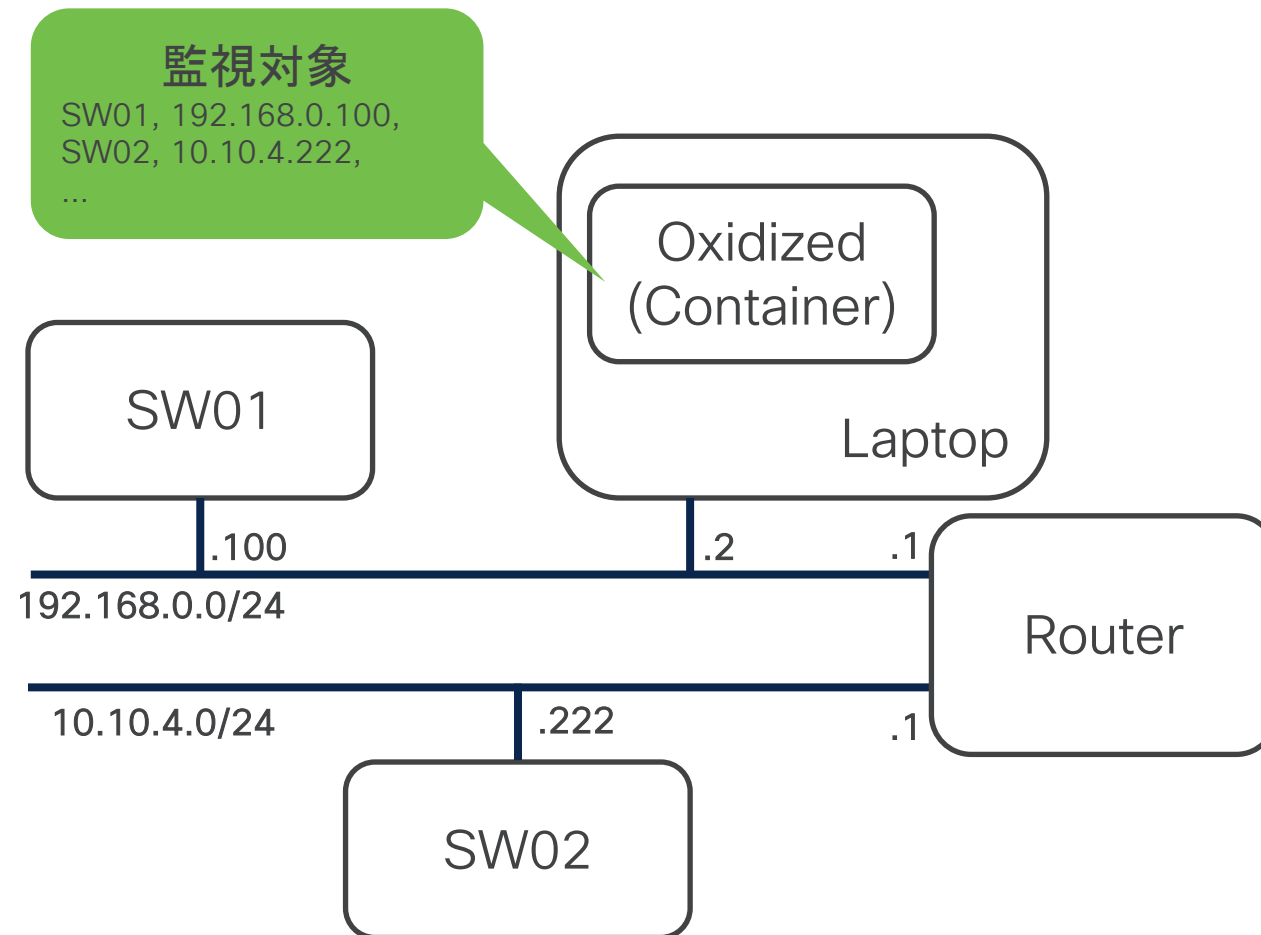
- 機器への疎通を確認でき、**使用可能な機器を把握**できるようになった
- 障害発生時の検知/通知で管理者が**すぐに故障を認識**できるようになった
- 検知によって復旧作業までのダウンタイムが短くなり、**可用性が向上**した

- Configの管理

- Configの収集を行うことで、**過去のConfigを確認**できるようになった
- 他のメンバーがConfigを変更していた場合でも、Rollbackを行うことで**以前の作業を再開**できるようになった

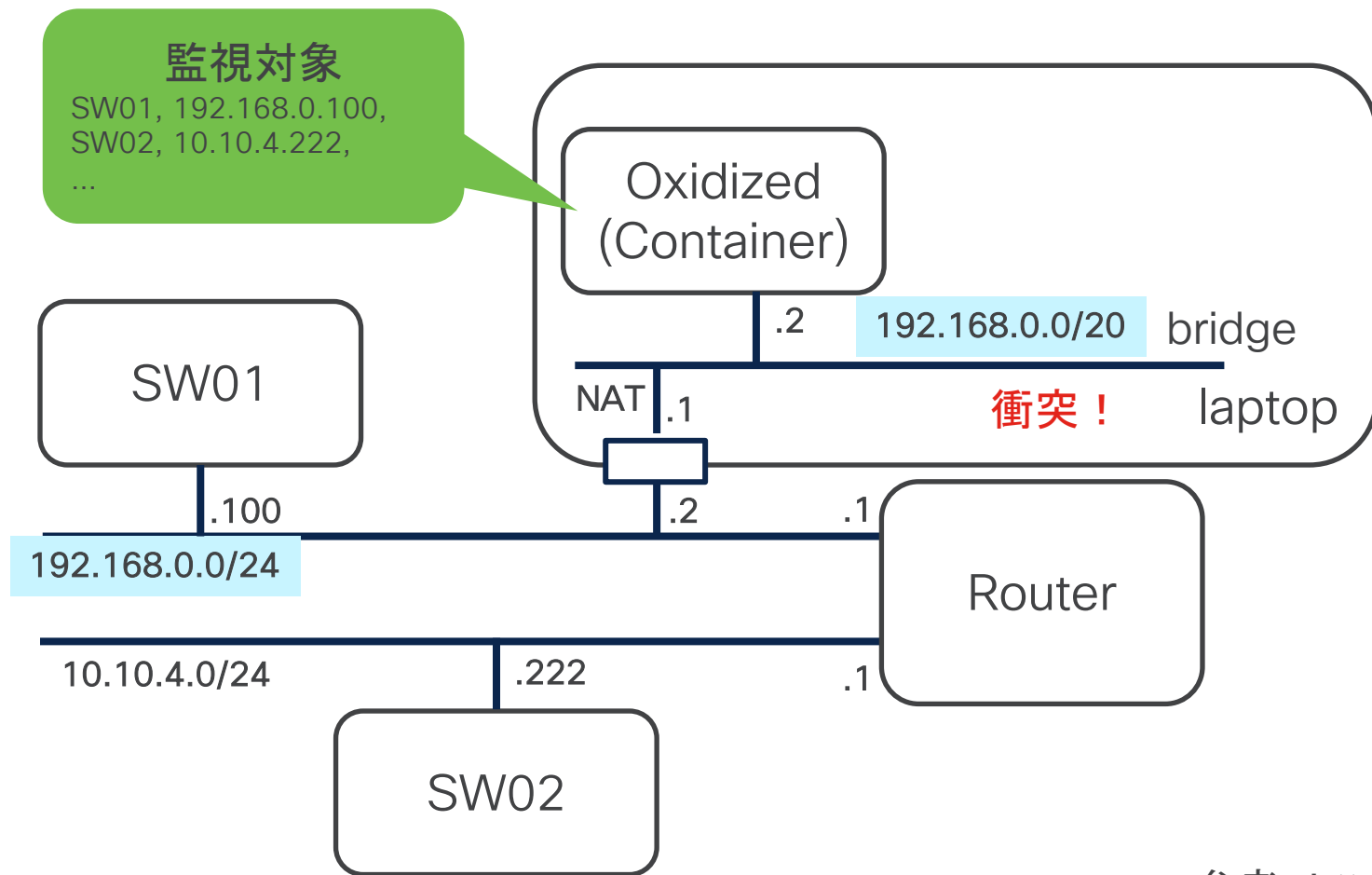
# 構築時のトラブル・知見

# docker bridge networkとLANの衝突



- Laptop上でdocker containerとしてOxidizedを構築中
- SW01のconfigの収集に失敗
- SW02では成功

# docker bridge networkとLANの衝突



docker bridge のアドレス (default)

- 172.17.0.0/16
- 172.18.0.0/16
- ...
- 172.31.0.0/16
- 192.168.0.0/20 ←16個目で衝突
- 192.168.16.0/20
- ...
- 192.168.240.0/20

参考: <https://github.com/docker/docs/issues/8663>



# OxidizedのREST API

- 最新のconfigは下記などで取得可能  
curl http://{oxidized IP address}/node/fetch/default/SW01
- 過去のversionについてはraw textやjsonで取得できない
  - 今回はHTMLのパーサを書いて解決した
  - 各configは内部のidで管理されており、欲しいversionのconfigのidを取得するために、version一覧のhtmlからidを探してくる必要もあった

Ruby学習して拡張する選択肢も...

```
</div><div>! NAME: &quot;1&quot;, DESCR: &quot;WS-C3560CX-8XPD-S&quot;  
</div><div>! PID: WS-C3560CX-8XPD-S , VID: V03 , SN: F0C2412L4H5  
</div><div>  
</div><div>  
</div><div>  
</div><div>! NVRAM config last updated at 06:35:28 UTC Tue Sep 20 2022  
</div><div>  
</div><div>version 15.2  
</div><div>no service pad  
</div><div>service timestamps debug datetime msec  
</div><div>service timestamps log datetime msec  
</div><div>no service password-encryption  
</div><div>  
</div><div>hostname 3560cx
```

# 対応機器

- 150以上のOSに対応
  - 詳細 -> <https://github.com/ytti/oxidized/tree/master/lib/oxidized/model>
  - 動作確認済み
    - Cisco - IOS (Catalyst 2960, 3560), IOS-XR (NCS 5500), NX-OS (Nexus 7000, 9000)
    - Brocade - NOS (VDX 6940)
    - Juniper - Junos (MX480, MX204)
    - (model自作) Alaxala - OS-L3A (AX3600S)
- 個人での拡張も容易
  - 細かいcommandを書き換えるだけ
    - ex) terminal length 0
    - set cli screen-length 0
    - set terminal pager disable

```
13 cmd 'show running-config' do |cfg|
14   cfg = cfg.each_line.to_a[3..-1].join
15   cfg
16 end
17
18 cfg :telnet do
19   username /^Username:/
20   password /^Password:/
21 end
22
23 cfg :telnet, :ssh do
24   post_login do
25     if vars(:enable) == true
26       cmd "enable"
27     elsif vars(:enable)
28       cmd "enable", /^([pP])assword:/
29       cmd vars(:enable)
30     end
31   end
32   post_login 'set terminal pager disable'
33   pre_logout 'exit'
34 end
35 end
```

↑ 作成したAlaxalaのmodel

<https://github.com/kkti4216/oxidized/blob/dev/lib/oxidized/model/alaxala.rb>

# Q&A · 議論

## 皆様にお聞きしたいこと

- アプローチについてのご意見
  - もっといいOSSあるよ！
  - こんな形でも実現できるのでは？
- 何を使ってLab環境を監視していますか？
  - Lab環境で必要な監視項目は？
- 何を使ってconfig管理/復元していますか？
  - 自動収集？手動で実行？
  - configを確認したいときは？sshしてshow run？