



IoT-GWプロビシシステムの裏側

伊藤忠テクノソリューションズ

渡部/樽崎

- 本日の議題とテーマ
- 自己紹介
- 自動化対象のサービス概要
- 現行サービスの課題と解決
- 今回開発した自動化システムについて
- 完成までの道のりと苦労したポイント
- 議論したいこと

- IoT-GWクラウドサービスのルーターのデリバリーの仕組みを自動化した経験から、設計から製造および試験での技術的な苦難を共有させていただきます
- 現在自動化と格闘している方々、またはこれから自動化に取り組む方々と議論をさせていただければ幸いです



自動化はしてるけど
四苦八苦中 . . .



これからどんどん
自動化していこう！

- 渡部友也（わたなべ ともや）

2010年から現職にてサーバー/NW/新規商材発掘等を担当

現在は通信キャリア様向けの

NWおよびNW自動化/運用高度化開発を主務

自動化できない作業はほぼないと思っています。

今回参加させていただいた上で、次に繋がる議論ができれば嬉しいです

※写真はCisco NSO Developer Days in Swedenです



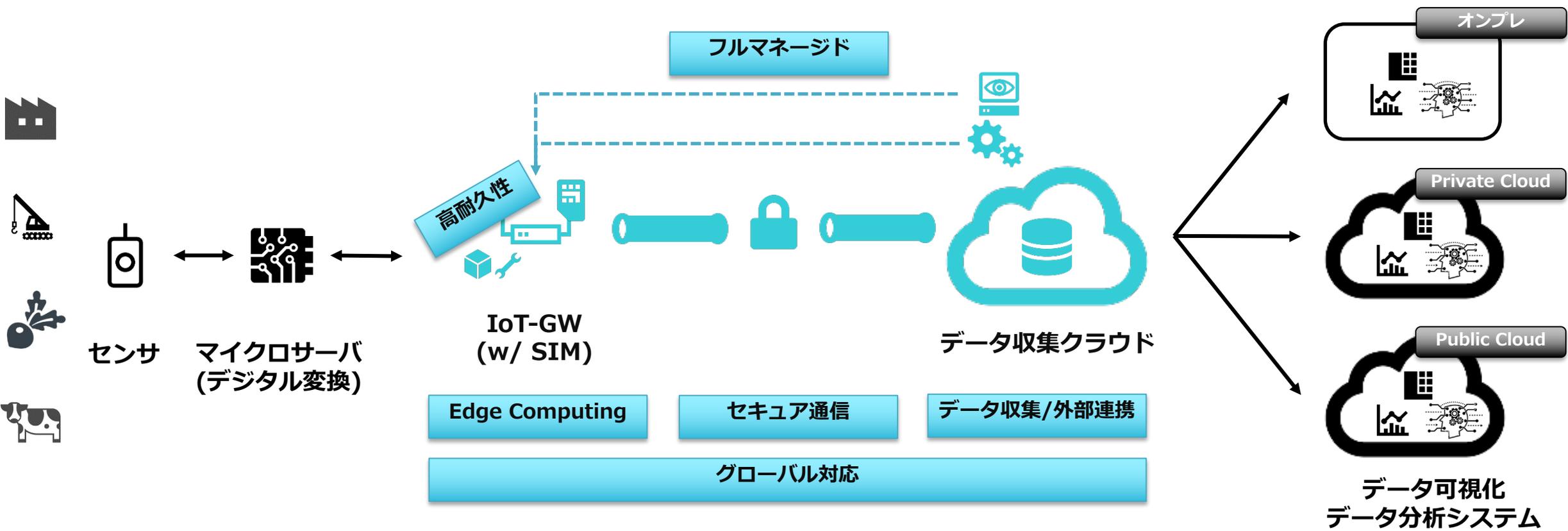
- 梶崎彩花（とがさき あやか）

2016年-2018年 通信キャリア様向け VoIP/NWを担当

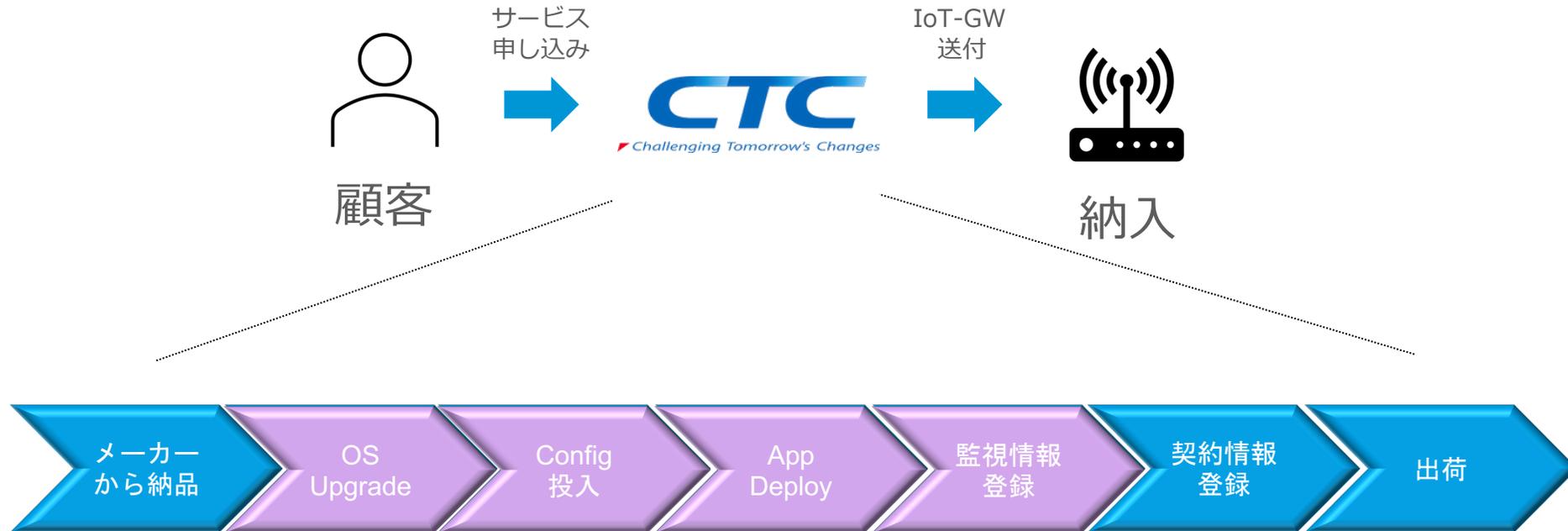
2018年-現在 同上 NWおよびNW自動化/運用高度化開発を主務

- 当たり前になってしまっている既存のめんどくさい作業はなるべく自動化していきたいなと考えてます。
- 今回の発表で苦労した点や同じ課題をもっている方達とさまざまな議論ができれば嬉しいです。

お客様は届いたルータの**電源をON**するだけで「簡単に」「安全に」IoT**データ収集可能**
 お客様は「どんなデータを」取得し、その「データをどう活用したいか」の検討/開発に専念



- サービスの申し込みからIoT-GWの納入までのながれ



- 納入までの作業はそれなりに多い
- 顧客ごとの情報等も考慮する必要がある
- 保守交換も考える必要がある

- サービスは既にローンチされていたため、自動化の仕組みは既存で存在していた。しかし、課題が大きく2つ存在していた

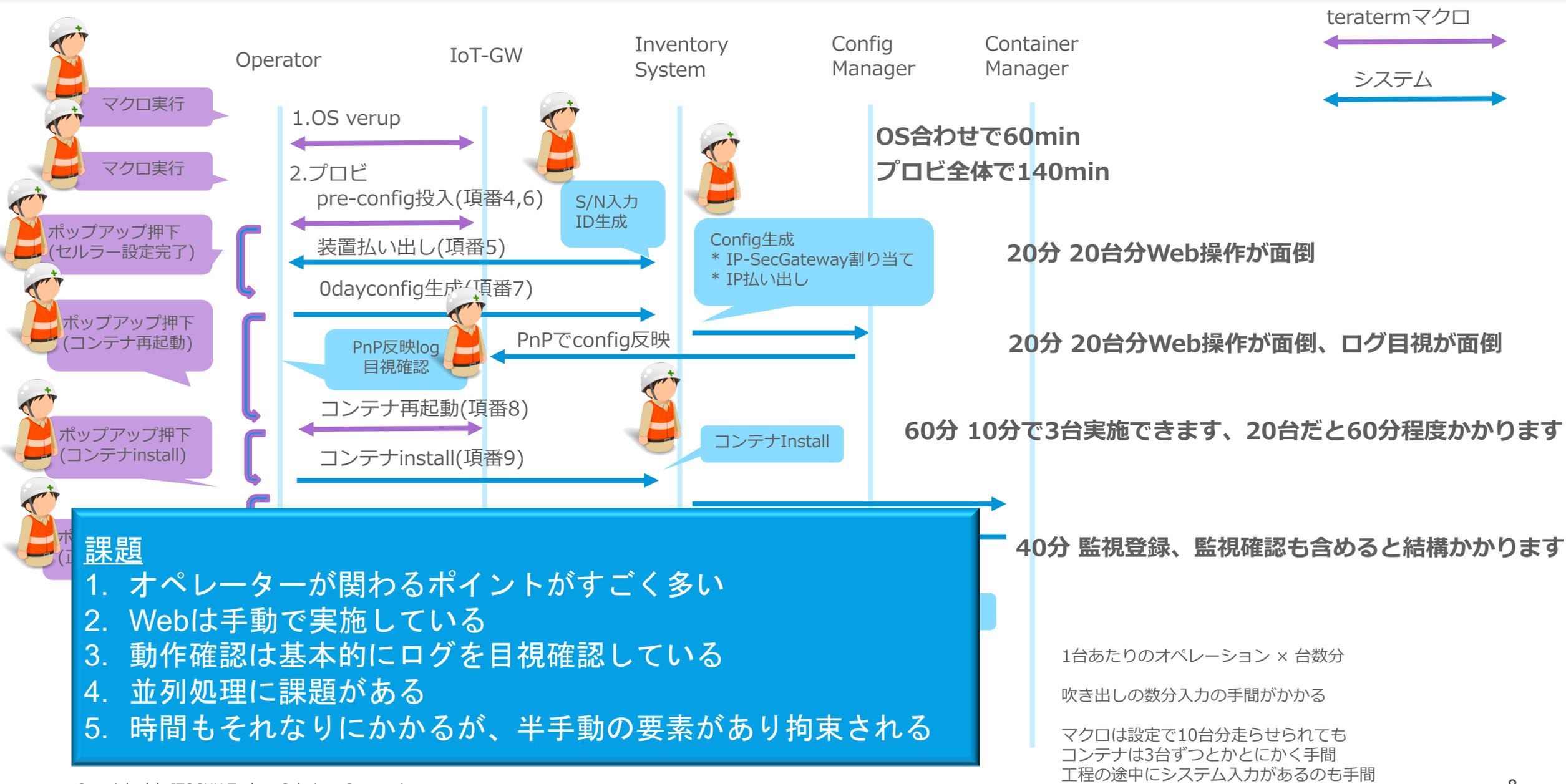
作成したマクロの機能不足

既存はTeratermマクロを利用してOS upgrade/Config投入を実施
作業が成功/失敗は基本的にマクロ終了時に人間が判断
最後の判断は人間になるため、ミスを誘発する可能性もあり

Web部分は手動作業

契約情報/監視情報はwebシステムとなるため、手動で実施していた
数十台を一度に作業するため、オペレーターへの負担が大きい

既存フローと課題（実際の資料）

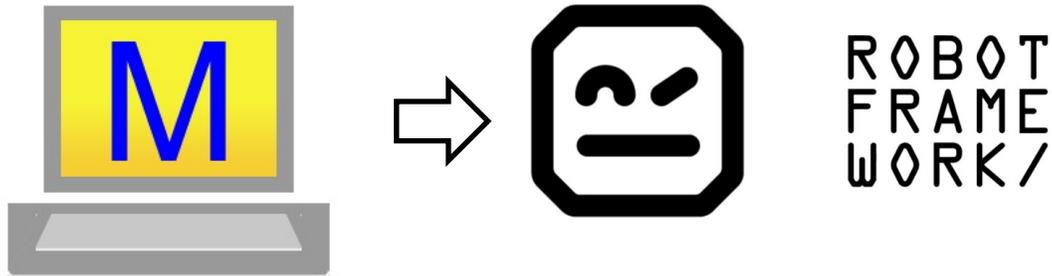


課題

1. オペレーターが関わるポイントがすごく多い
2. Webは手動で実施している
3. 動作確認は基本的にログを目視確認している
4. 並列処理に課題がある
5. 時間もそれなりにかかるが、半手動の要素があり拘束される

1台あたりのオペレーション × 台数分
 吹き出しの数分入力の手間がかかる
 マクロは設定で10台分走らせても
 コンテナは3台ずつとかとにかく手間
 工程の途中にシステム入力があるのも手間

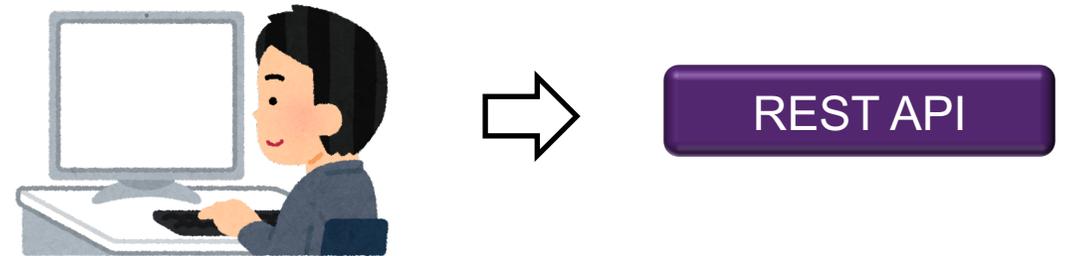
- TeratermマクロのRobot Framework化



- メリット

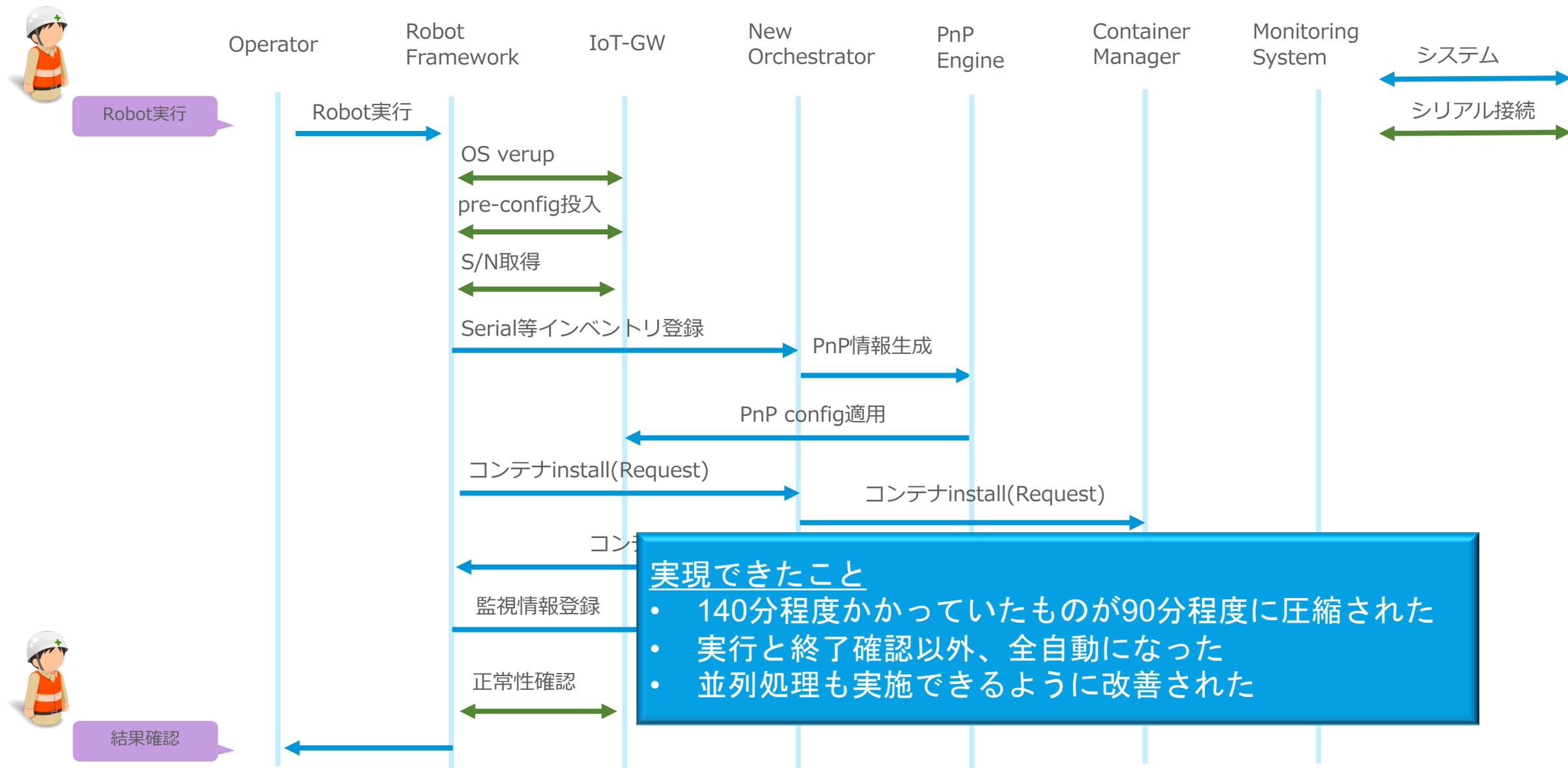
- ✓ 試験自動化FrameworkなのでAssertion(合否判定が自動で可能)
- ✓ 実行結果がhtmlのレポートとして自動生成
- ✓ PythonをDriverとして利用できるので、色々と小回りが効くので並行処理とかもうまくやれば可能

- Web系SysとのREST API結合による完全自動化

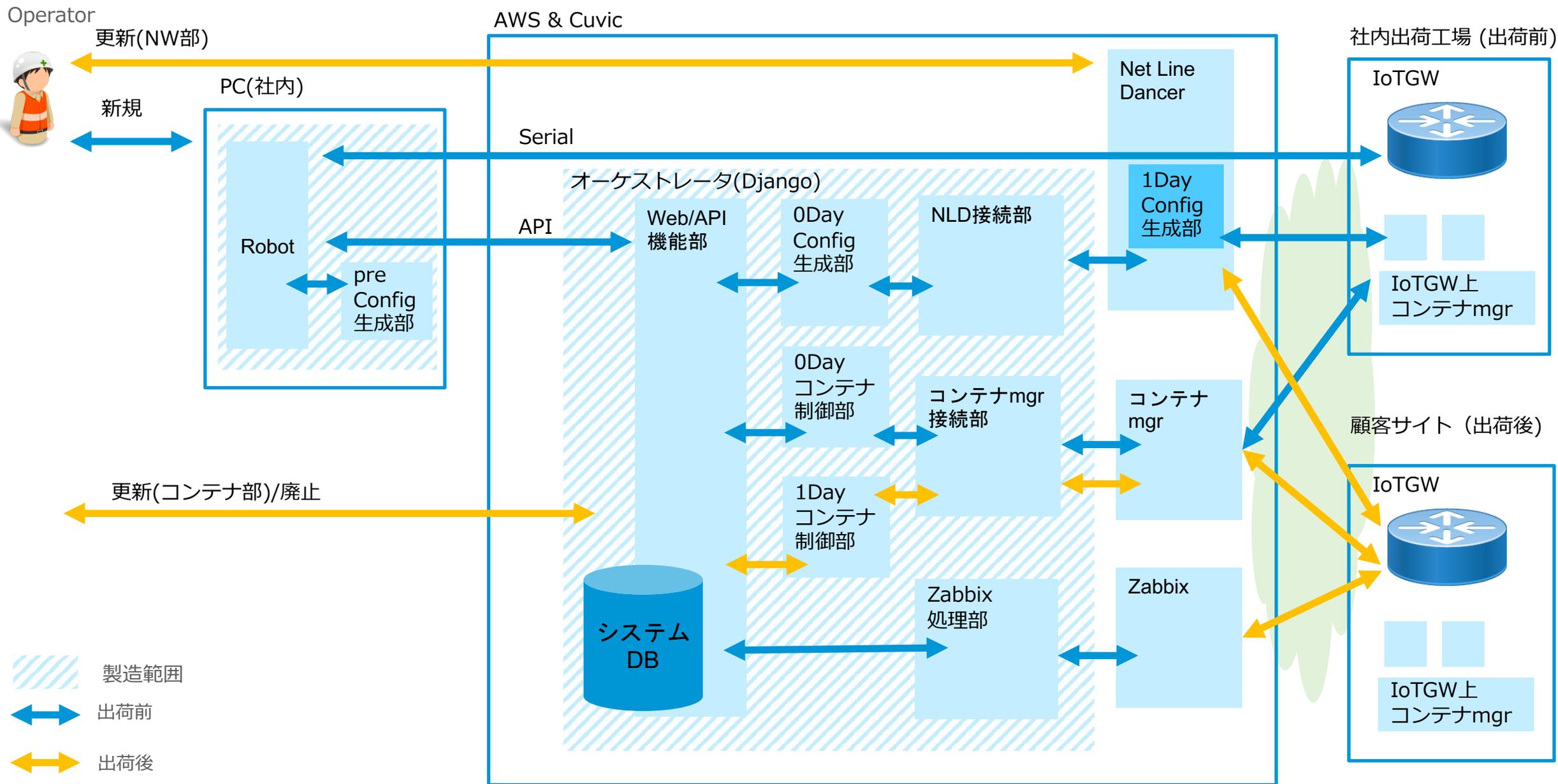


- メリット

- ✓ 手動Web操作の排除による効率化
- ✓ etc...



**ここからは実際の実装の話と苦勞の話となるので、
開発者の拇崎にスピーカーチェンジします**



シーケンス (出荷前 新規)

Operator



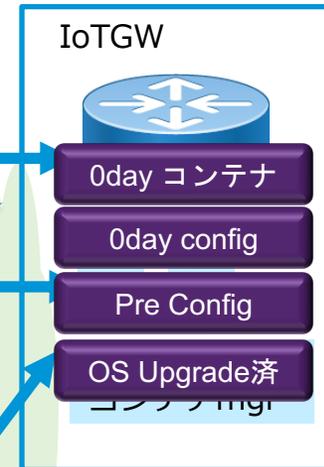
新規②作業

- オーケストレータにデバイス(シリアル)登録
- IP及び変数払い出し
- 0DayConfigの生成&NLDに登録

新規①作業

- OSのupgrade
- シリアル取得
- Pre config生成&投入

社内出荷工場 (出荷前)



新規③作業

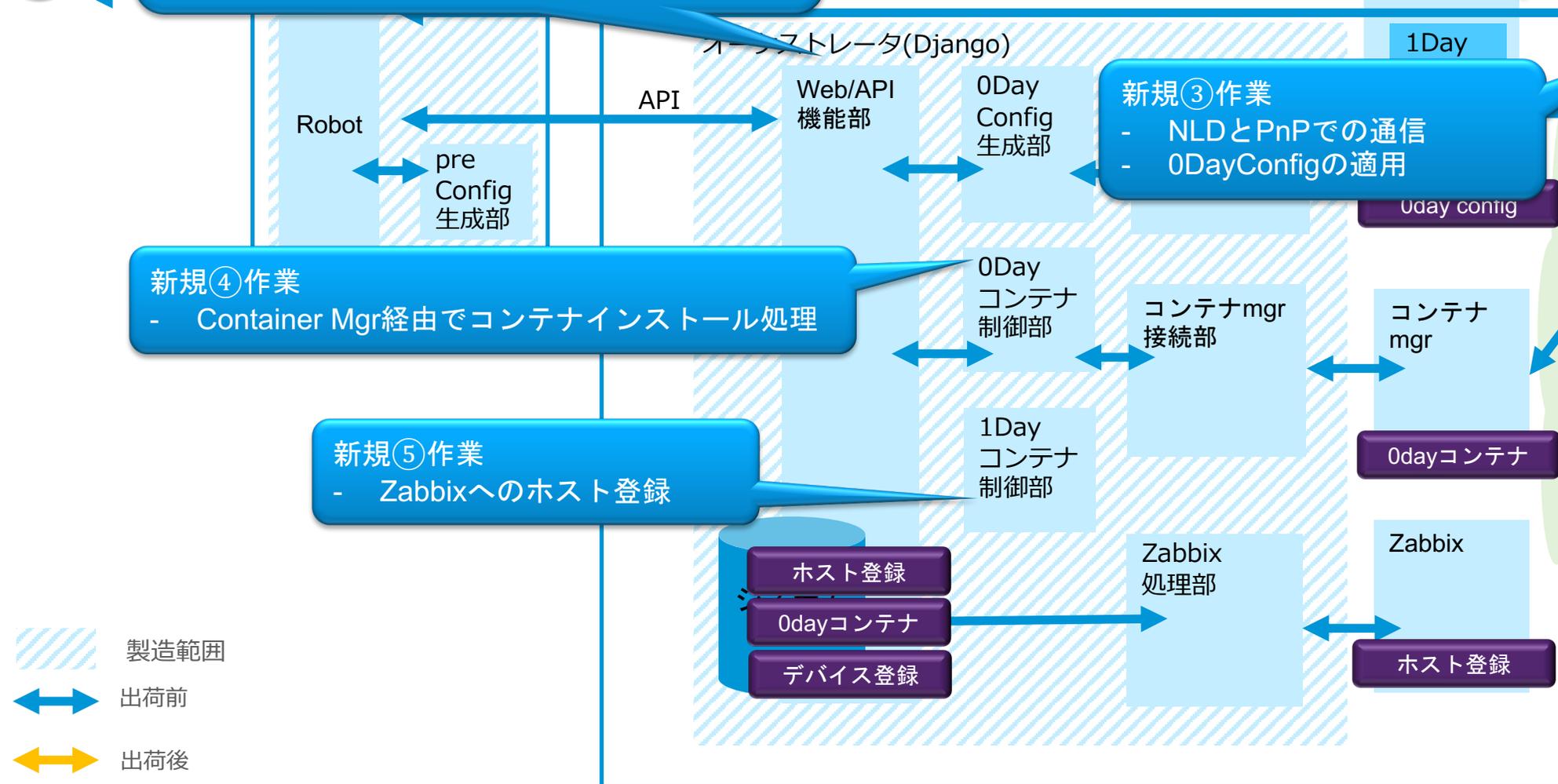
- NLDとPnPでの通信
- 0DayConfigの適用

新規④作業

- Container Mgr経由でコンテナインストール処理

新規⑤作業

- Zabbixへのホスト登録



- 製造範囲
- 出荷前
- 出荷後

シーケンス (出荷後 更新/廃止)

Operator



更新(NW部)

AWS & Cuvic

更新①作業
 - NLDからConfig push
 - NLDからOS upgrade

更新②作業
 - ContainerMgr経由でコンテナ入替
 - ContainerMgr経由でコンテナVerup

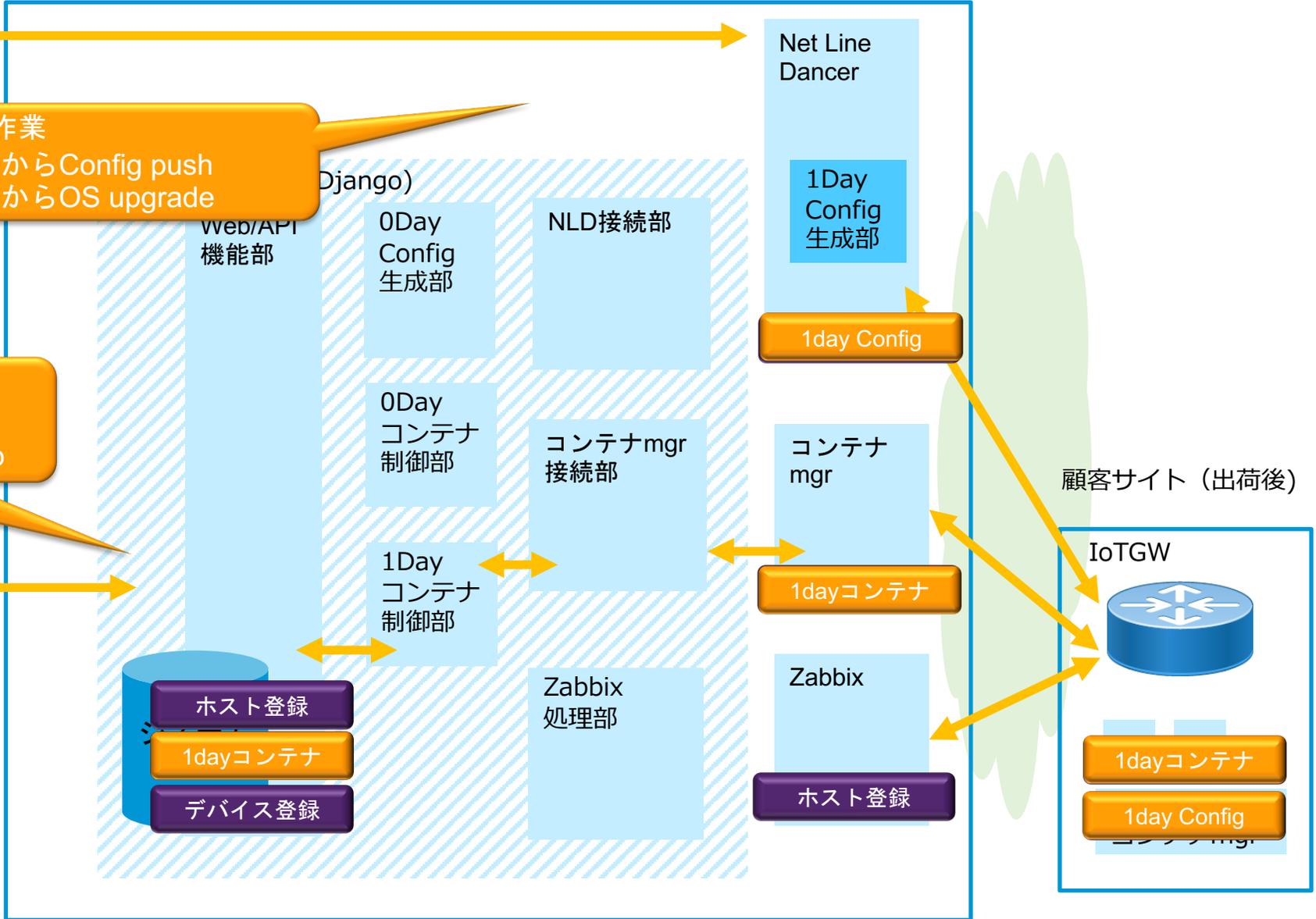
更新(コンテナ部)/廃止

廃止作業
 - Orc経由で全ての設定削除

製造範囲

出荷前

出荷後



- 完成までの道のり



設計 ~ 環境構築 ~ 製造 ~ 自動試験製造 ~ 総合試験(手動) ~ doc類整備 ~ ...

- 苦勞ポイント



- 設計がFixしない

なぜなのか



- ステークホルダーからの要望

	Operator	設計者(既存課題)	設計者(新規要望)
課題	装置ごとに払い出したInventory情報をwebに手入力するのが手間(装置シリアルとIPの紐付けとか)	現行システムの運用コストがかかりすぎる	Inventory情報を自動で払い出して欲しい(IP,シリアル情報の自動紐付け)
	キッキング中目視確認するタイミングが常にあるので離席できない(半自動)	納入後のconfig,コンテナの変更(verup等も含む)ができない	他システムとの連携(監視システムや契約管理システム)
	複数台の同時作業が手間	-	将来的に機能拡張性が欲しい

そもそもの要望ははっきりしていた
基本設計は簡単に固まった。
しかし、詳細設計Fixは難航した。

苦勞した点：設計フェーズ

- NW詳細設計のFixが遅れた
 - ✓ NWの設計が固まらなかった(pre config/0day config/1day config)
 - ✓ configのfixが試験直前までできなかつたため、試験中に自動化側で改修が必要だった
- 商用/検証環境のインフラ構成（AWS）が流動的だった
 - ✓ NWチームと自動化チームで役割分担が曖昧だった
 - ✓ 環境の設計fixが試験直前までできなかつたため、オーケストレータを動作させるnginx等のconfigが急務になった
- 自分のスキル&経験不足
 - ✓ 開発面(DB Model設計)、インフラ面の知識
 - ✓ 商用/検証/開発環境をfixさせられなかつた

苦労した点：製造フェーズ

- 製造中から試験を考慮してCIを実装していた
- APIの冪等性を考慮して実装が必要だった

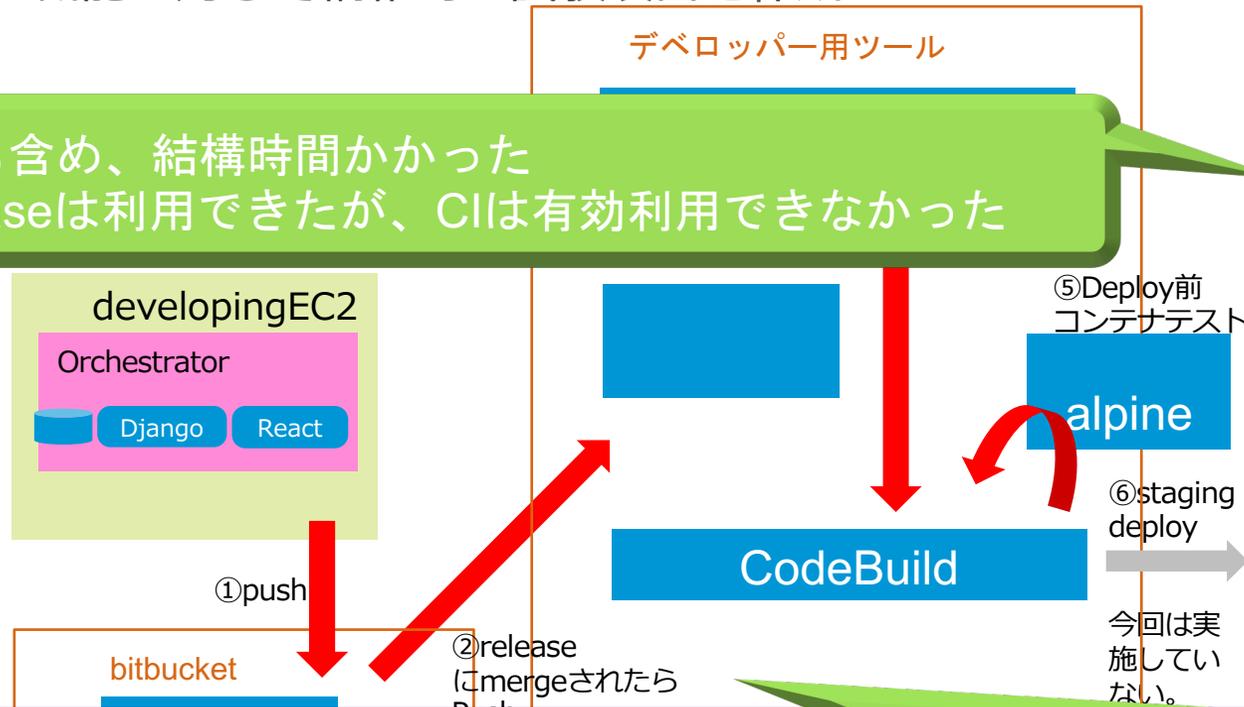
実装において幅広い技術知識が必要だった



苦勞した点：製造フェーズ

- 製造中から試験を考へてCIを実装していた
 - ✓ オークストレータのAPI製造部分をCIで自動試験
 - ✓ 自動試験の実装は、DjangoのTestcaseを使用
 - ✓ 製造した機能に対して網羅的に試験項目を作成

➤ 習熟も含め、結構時間かかった
 ➤ Testcaseは利用できたが、CIは有効利用できなかった



• 開発者や検証者と環境を共有してしまっていたため、バッティング
 • APIの中でシステム外のコンポーネントや実機との連携もある

商用deployを自動で実施するのは危険なので組み込んでいない

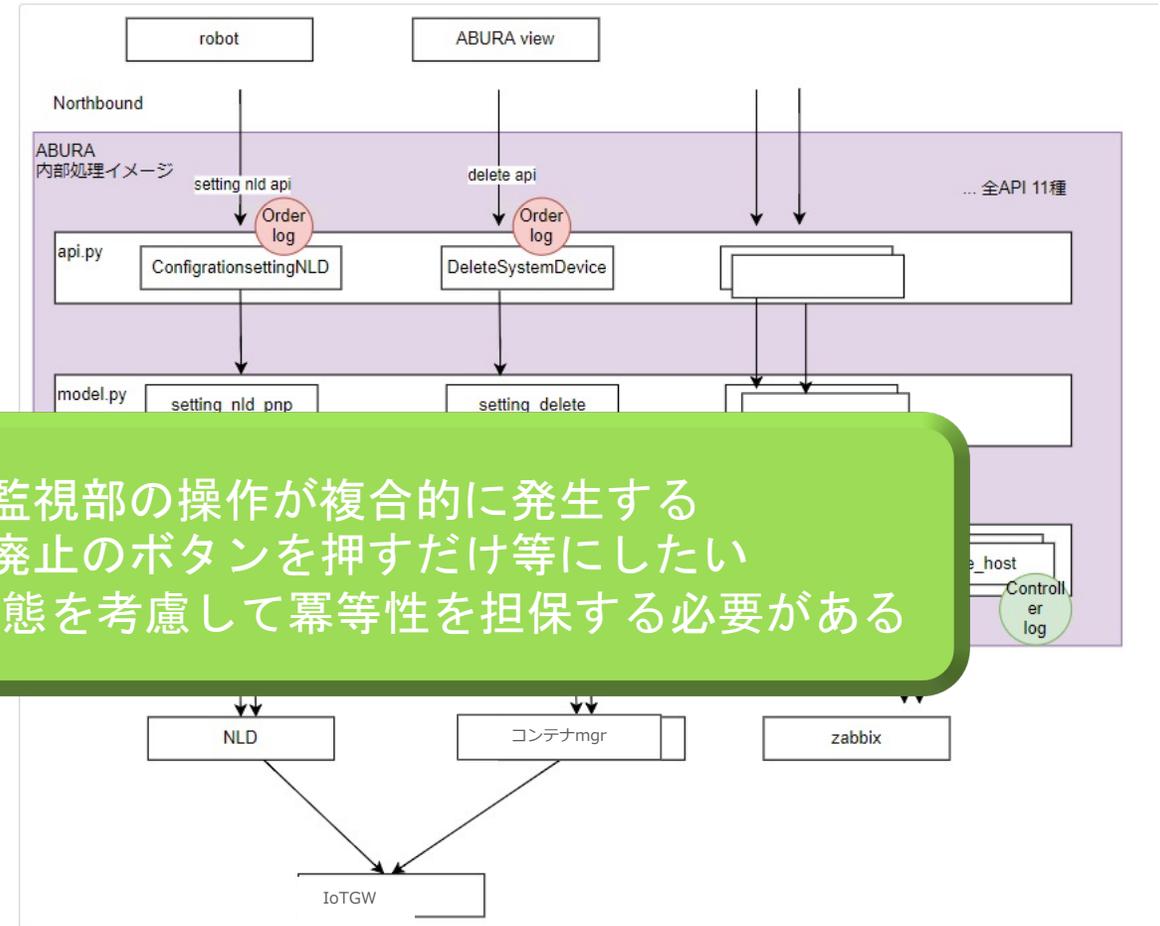
The screenshot shows the AWS CodePipeline console for a pipeline named 'ibucket-abura'. It displays three stages:

- Stage Source:** CodePipeline stage execution STARTED. Source Actions.
- Stage Build:** CodePipeline stage execution STARTED. Source Actions.
- Stage Source:** CodePipeline stage execution SUCCEEDED. Source Actions.
- Stage Build:** CodePipeline stage execution SUCCEEDED. Source Actions.

Timestamps for the notifications are 19:46, 19:46, and 19:52.

苦労した点：製造フェーズ

- APIの冪等性を考えて実装が必要だった
 - ✓ 運用者向けに、シンプルなオペレーションにしたかった
 - ✓ 外的要因での失敗時に、切り分けを不要にしたい
- Provisioning実施/更新/廃止の3つのボタン
 - ✓ 実施の途中でどこで失敗しても、廃止を押すだけで最初の状態に戻る設計



- 各APIごとにNW部、コンテナ部、監視部の操作が複合的に発生する
- 運用を考えると、どこでfailしても廃止のボタンを押すだけ等にした
- 実装上は外部システムAPI仕様や状態を考慮して冪等性を担保する必要がある

苦労した点：試験フェーズ

- Serialポートの罫で総合試験1000本ノックになった
- NWチームと自動化チームの連携がうまくいかず、試験実施が長引いた
- 環境差分に悶絶した

試験はやっぱり大変



苦勞した点：試験フェーズ

- Serialポートの罫で総合試験1000本ノックになった
Serial接続で装置仕様の制約があった

SUITE 01 Os Update

SUITE 02 Ir Setup

Full Name: Scenario.02 Ir Setup

Source: C:\Users\z5g4027\ibucket-rf\scenario\02_ir_setup.robot

Start / End / Elapsed: 20210312 14:49:03.478 / 20210312 15:51:51.489 / 01:02:

Status: 20 critical test, 6 passed, 14 failed
20 test total, 6 passed, 14 failed

TEST 2001. 事前確認

TEST 2002. ...

TEST 2003. ...

TEST 2004. ...

TEST 2005. ...

TEST 2006. ...

TEST 2007. ...

TEST 2008. ...

TEST 2009. ...

TEST 2010. ...

TEST 2011. ...

TEST 2012. ...

TEST 2013. ...

TEST 2014. ...

TEST 2015. ...

TEST 2016. ...

TEST 2017. ...

TEST 2018. ...

TEST 2019. ...

TEST 2020. ...

TEST 2021. ...

TEST 2022. ...

TEST 2023. ...

TEST 2024. ...

TEST 2025. ...

TEST 2026. ...

TEST 2027. ...

TEST 2028. ...

TEST 2029. ...

TEST 2030. ...

TEST 2031. ...

TEST 2032. ...

TEST 2033. ...

TEST 2034. ...

TEST 2035. ...

TEST 2036. ...

TEST 2037. ...

TEST 2038. ...

TEST 2039. ...

TEST 2040. ...

TEST 2041. ...

TEST 2042. ...

TEST 2043. ...

TEST 2044. ...

TEST 2045. ...

TEST 2046. ...

TEST 2047. ...

TEST 2048. ...

TEST 2049. ...

TEST 2050. ...

TEST 2051. ...

TEST 2052. ...

TEST 2053. ...

TEST 2054. ...

TEST 2055. ...

TEST 2056. ...

TEST 2057. ...

TEST 2058. ...

TEST 2059. ...

TEST 2060. ...

TEST 2061. ...

TEST 2062. ...

TEST 2063. ...

TEST 2064. ...

TEST 2065. ...

TEST 2066. ...

TEST 2067. ...

TEST 2068. ...

TEST 2069. ...

TEST 2070. ...

TEST 2071. ...

TEST 2072. ...

TEST 2073. ...

TEST 2074. ...

TEST 2075. ...

TEST 2076. ...

TEST 2077. ...

TEST 2078. ...

TEST 2079. ...

TEST 2080. ...

TEST 2081. ...

TEST 2082. ...

TEST 2083. ...

TEST 2084. ...

TEST 2085. ...

TEST 2086. ...

TEST 2087. ...

TEST 2088. ...

TEST 2089. ...

TEST 2090. ...

TEST 2091. ...

TEST 2092. ...

TEST 2093. ...

TEST 2094. ...

TEST 2095. ...

TEST 2096. ...

TEST 2097. ...

TEST 2098. ...

TEST 2099. ...

TEST 2100. ...

```

15:48:19.738 DEBUG read_channel:
15:48:19.848 FAIL ValueError: Failed to enter enable mode. Please ensure you pass the 'secret' argument to ConnectHandler.
15:48:19.848 DEBUG Traceback (most recent call last):
File "C:\Users\z5g4027\ibucket-rf\IRSetupLibrary.py", line 60, in enter_enable_mode
self.conn.enable()
File "c:\program files (x86)\python36-32\lib\site-packages\netmiko\cisco_base_connection.py", line 18, in enable
return super().enable(cmd=cmd, pattern=pattern, re_flags=re_flags)
File "c:\program files (x86)\python36-32\lib\site-packages\netmiko\base_connection.py", line 1573, in enable
raise ValueError(msg)

```

• SyslogのlogがConsole上に割り込み表示され、netmikoの想定Promptと異なりError

• 転送速度の問題で文字列を認識せずerror

- Netmikoはtelnet/ssh想定 of 処理速度。
- Serial接続は文字列の送受信の速度はもっと遅い。

```

17:27:11.560 DEBUG read_channel:
17:27:11.560 DEBUG IR800#
Pattern found: IR800
*Mar 12 08:26:39.406: %LINK-3-UPDOWN: Interface Cellular0, changed state to down
*Mar 12 08:26:39.406: %LINK-3-UPDOWN: Interface Cellular1, changed state to down
*Mar 12 08:26:55.443: %CELLWAN-2-BEARER_UP: Instance id=0, Default bearer (bearer_id=5) in Cellular0 is now UP
*Mar 12 08:26:57.554: %LINK-3-UPDOWN: Interface Cellular0, changed state to up
*Mar 12 08:26:58.554: %LINEPROTO-5-UPDOWN: Line protocol on Interface Cellular0, changed state to up

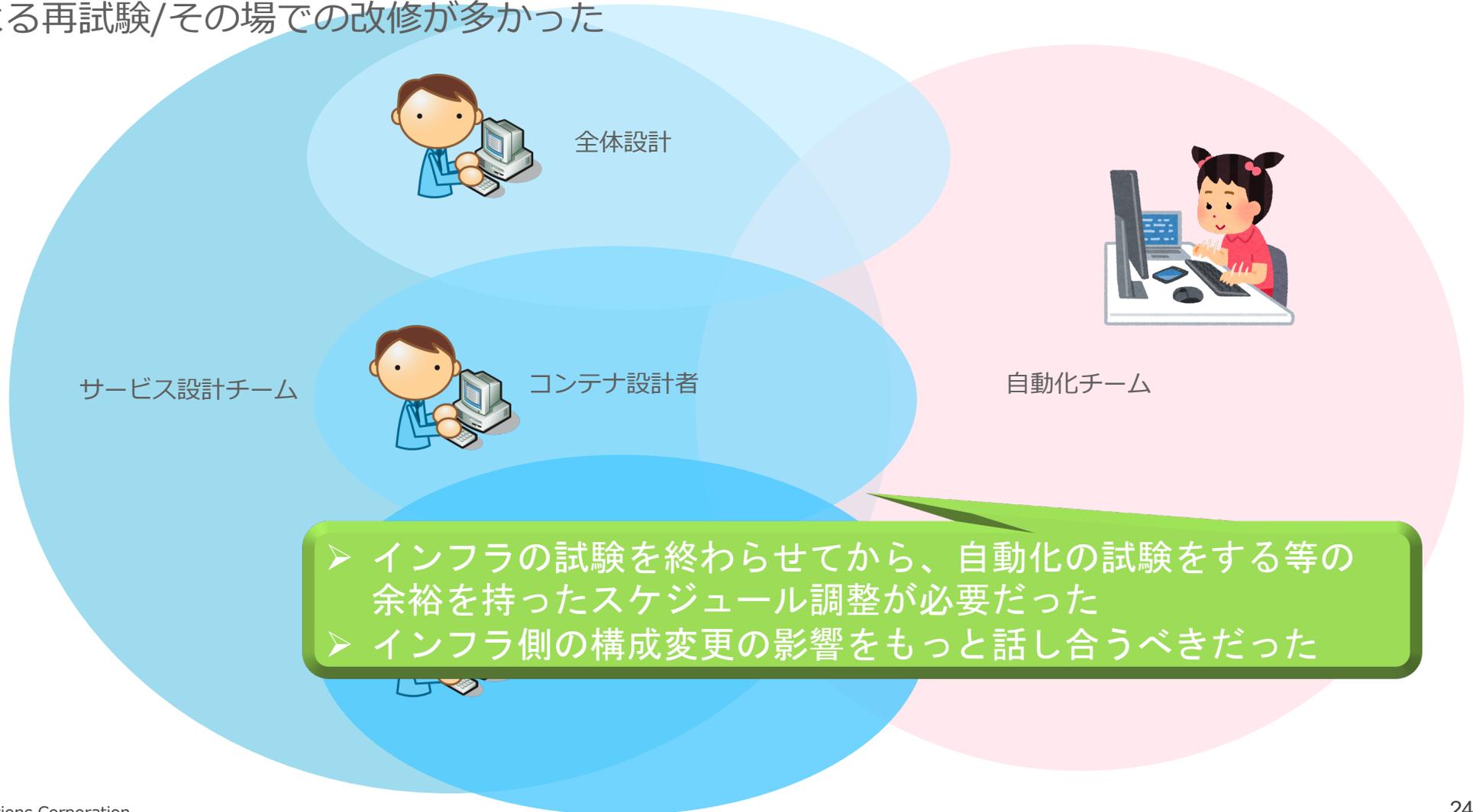
```

Copyright (c) ITOCHU Techno-Solutions Corporation

23

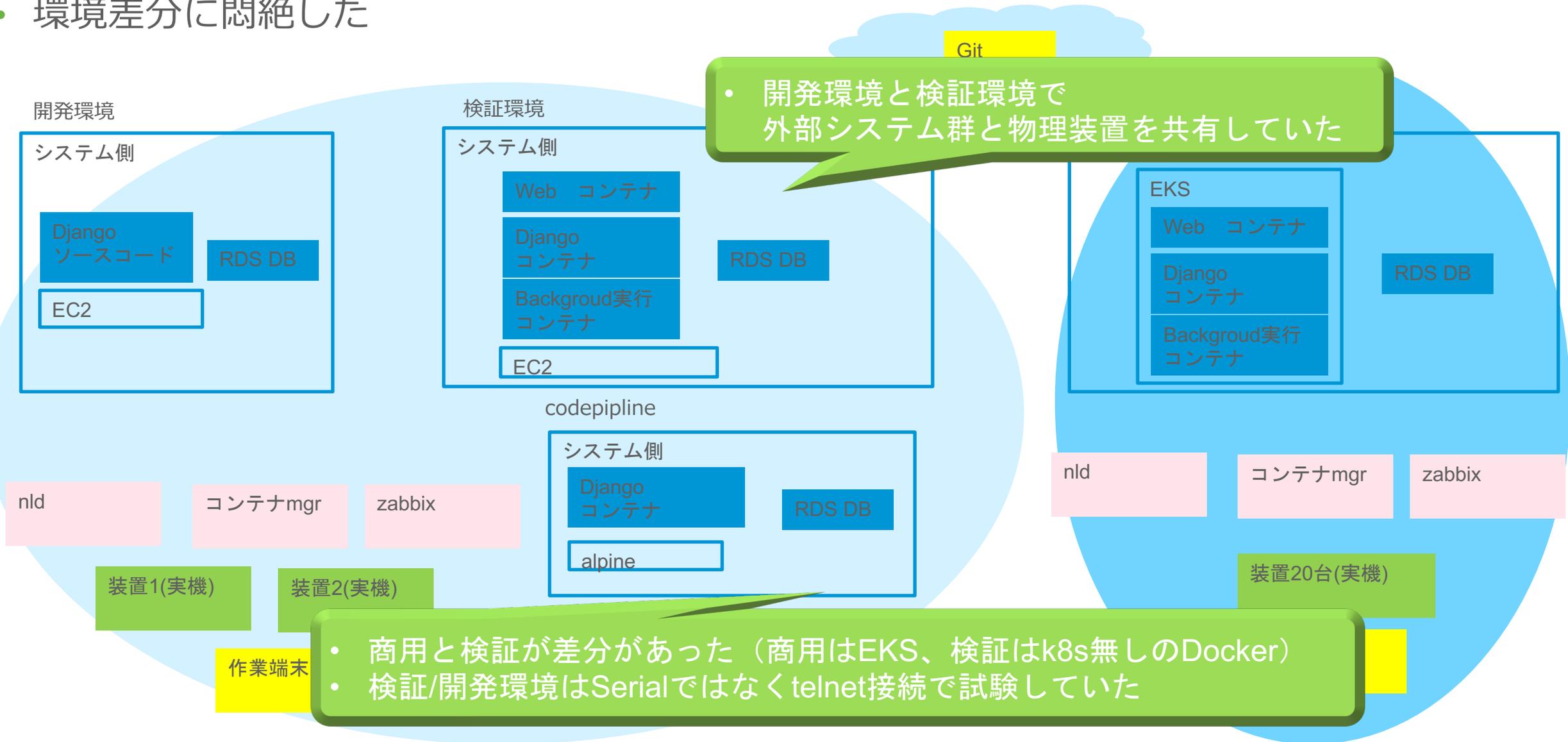
苦勞した点：試験フェーズ

- NWチームと自動化チームの連携がうまくいかず、自動化システム試験が難航した
 - 自動化チームの試験実施のタイミングで、インフラ要因で試験が失敗して進まない事象が多かった
 - 環境構成変更による再試験/その場での改修が多かった



苦労した点：試験フェーズ

- 環境差分に悶絶した



苦労した点：その後(これから…)

- 引き継ぎ
運用者、設計者共に…

- 製造者と運用していくメンバーが異なる
- Docで伝わりきるのかどうか

セットアップ(キッティング)手順書

ABURA、NLD、Fogd、Zabbixの各コンポーネントでデータが登録されているか確認する。

- ABURA

1. **Device一覧**を開く
2. 作成したデバイス(作業対象のIR829のS/N)があることを確認。成功した場合は赤枠
※この際に青枠内に記載してある、IR829のS/Nに紐づく「デバイスip」を控えてお

Device一覧

<input type="checkbox"/>	id	シリアル番号	デバイスip	order_status	pr
<input type="checkbox"/>	280	FGL2351LFLF	172.30.5.9	運用中	no
<input type="checkbox"/>	279	FGL2351LFMW	172.30.5.8	運用中	no
<input type="checkbox"/>	278	FGL2351LFLQ	172.30.5.7	運用中	no

ABURA API

View 編集

プレゼンテーション 更新履歴

ABURA API

API一覧

APIは全部で11種類。

プロビ実行時のapiだけでなく、画面からの更新、削除の際も全て内部的にはAPIを実行している。

api実行時のurlやbody、内容についてはソースコード側のabura_api.restをmasterにして管理しているため割愛。

詳細は、下記URLを参照してください。

https://bitbucket.org/ctc-repo/ibucket-abura/src/af814d7e9fa67a2ee104e0a8cb22da041f10e85f/api_rest/abura_api.rest?at=develop

APIエラー一覧

エラー一覧

全部で下記5種類

- AburaSystemError
- ZabbixAPIError

- セットアップ(キッティング)手順書
 - 前提条件
 - 物理的事前準備
 - 1.システム障害状況確認連絡

Updated by Hir
2021/04/19 19:1

- ABURA API
 - API一覧
 - APIエラー一覧
 - エラー一覧
 - 各エラー詳細
 - 参考ソースコード
 - APIテストケース
 - テスト項目作成観点
 - テストケース一覧
 - テストケース回し方

- NWチームと自動化チームの連携方法
 - 役割分担
 - サービス設計者へシステム運用改修の引き継ぎなど
 - そもそもチームが分かれてない？等

- 開発、検証、商用環境
 - 環境差分をどこまで許容しているか
 - CI/CDどこまでやるのがベストプラクティスか

CTC

▼ *Challenging Tomorrow's Changes*