Bugゼロ

Chaos in a dev world from zero

から始めるカオスエンジニアリング

Photo by Morinobu

セッション5 15:05 - 15:35

品質活動における各領域の現在地

知っている・理解している

テスト

Jest pytest

知らない・理解している

Citation: ポプテピピック

知っている・理解していない

モニタリング

Alarms Dashboards

知らない・理解していない

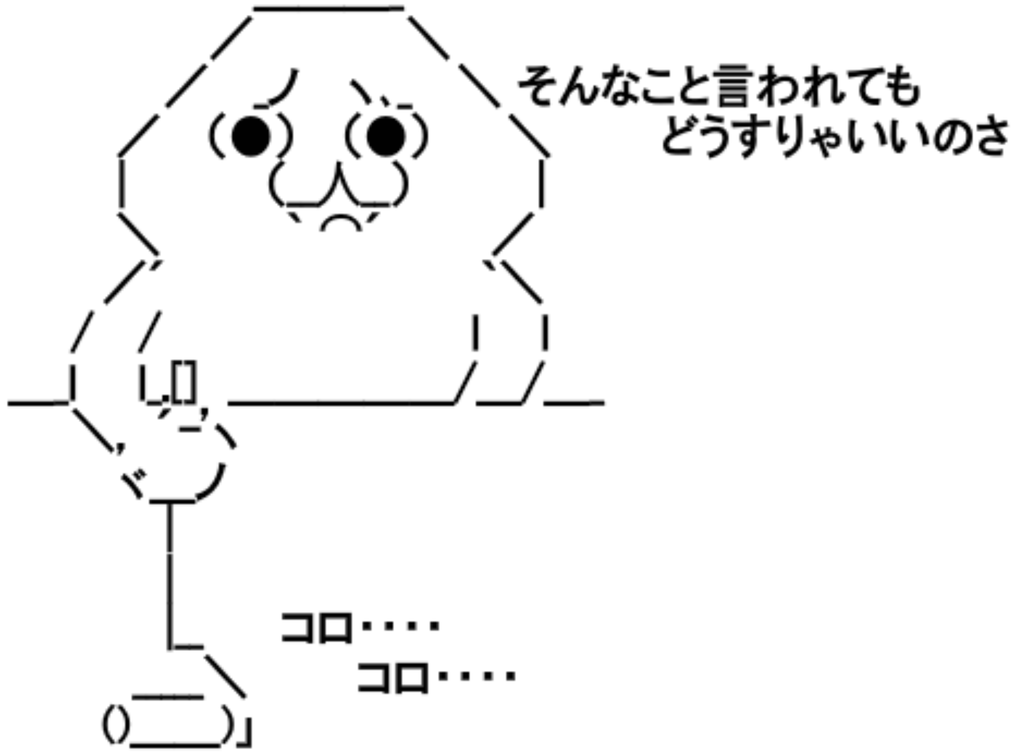
カオス
エンジニアリング

Observability

RUM

基礎となる原則

1. 定常状態に関する仮説を立てる
2. 実世界の事象を多様化させる
3. **本番環境**で実験する
4. **継続的に実行**できるよう実験を自動化する
5. 影響範囲を局所化する



Citation：旧2ch

運用中のシステムに意図的に障害は起こしたくない

-> 導入へのハードルは高い

自己紹介１

ダイキン工業株式会社

Speaker：谷尾 虎之介

Quality **C**ontrol **E**ngineer

@アジャイル内製化チーム

ダイキン情報技術大学４期生として入社

入社3年目

現チームでは開発者として1年半従事

好きなこと：CI/CDの高度・最適化

ダイキン情報技術大学

不足するデジタル人材を
育成するための社内大学

広い分野でDXを推進

✓データ分析

✓AI技術開発/活用

✓システム開発

ダイキン新卒100人「仕事しなくてOK」のなぜ

AI人材争奪線とは一線、自前育成の大胆戦略

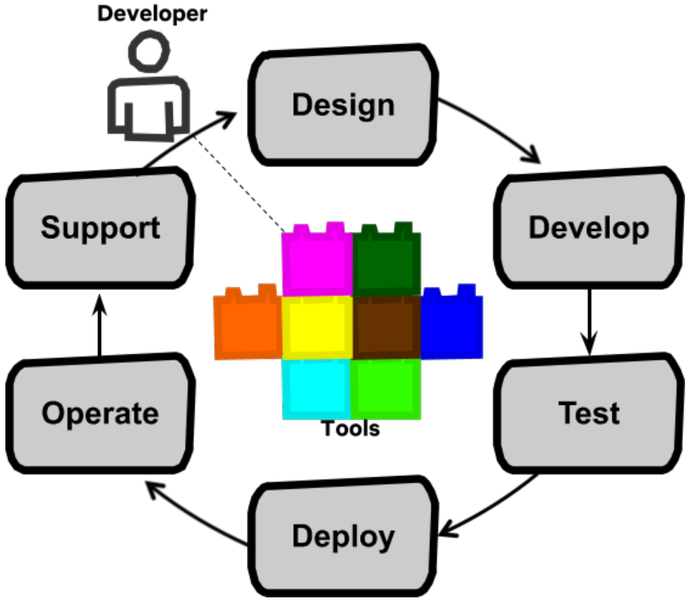
ダイキン工業
制作：東洋経済ブランドスタジオ

AD
2019/04/26

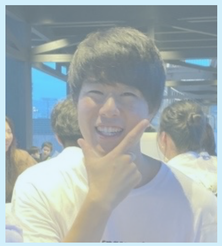
Citation: https://toyokeizai.net/articles/-/271400

アジャイル内製化チーム

設計/テスト/運用を含めてフルサイクル開発



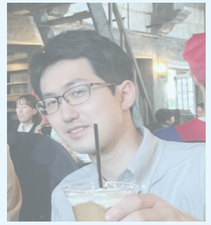
PO



SM

開発

PO



SM

開発

アジャイルコーチ

チーム体制とプロダクト

ダイキン エネルギー マネジメント サービス

EneFocus α

エネフォーカス アルファ

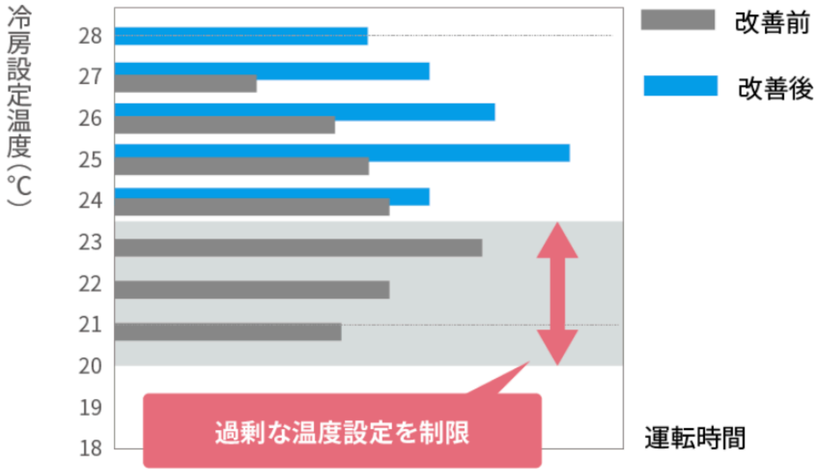
空調運用改善のサブスクリプションサービス

設定温度の上下限管理

「冷やしすぎていた部屋」 の23℃以下を

設定不可 にすることで省エネ

1F エントランス



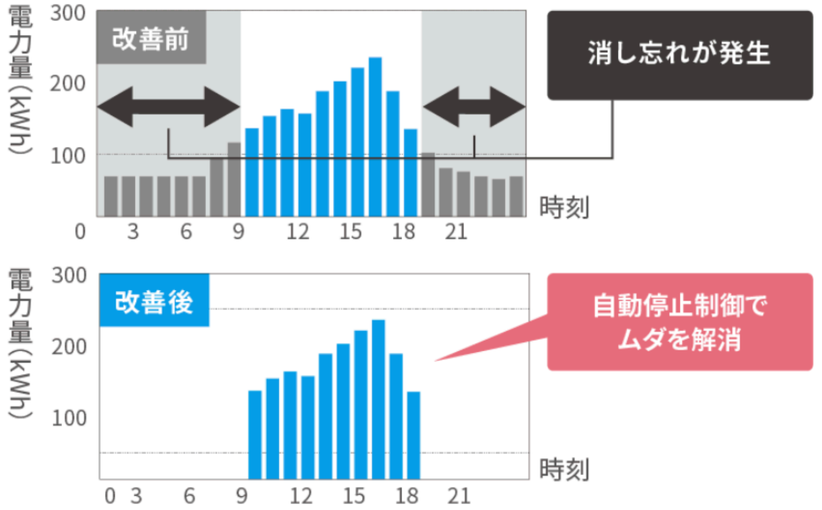
※グラフはイメージです。

消し忘れ防止設定

「夜間の消し忘れがあった部屋」 を

消し忘れ防止設定で省エネ

1F エントランス

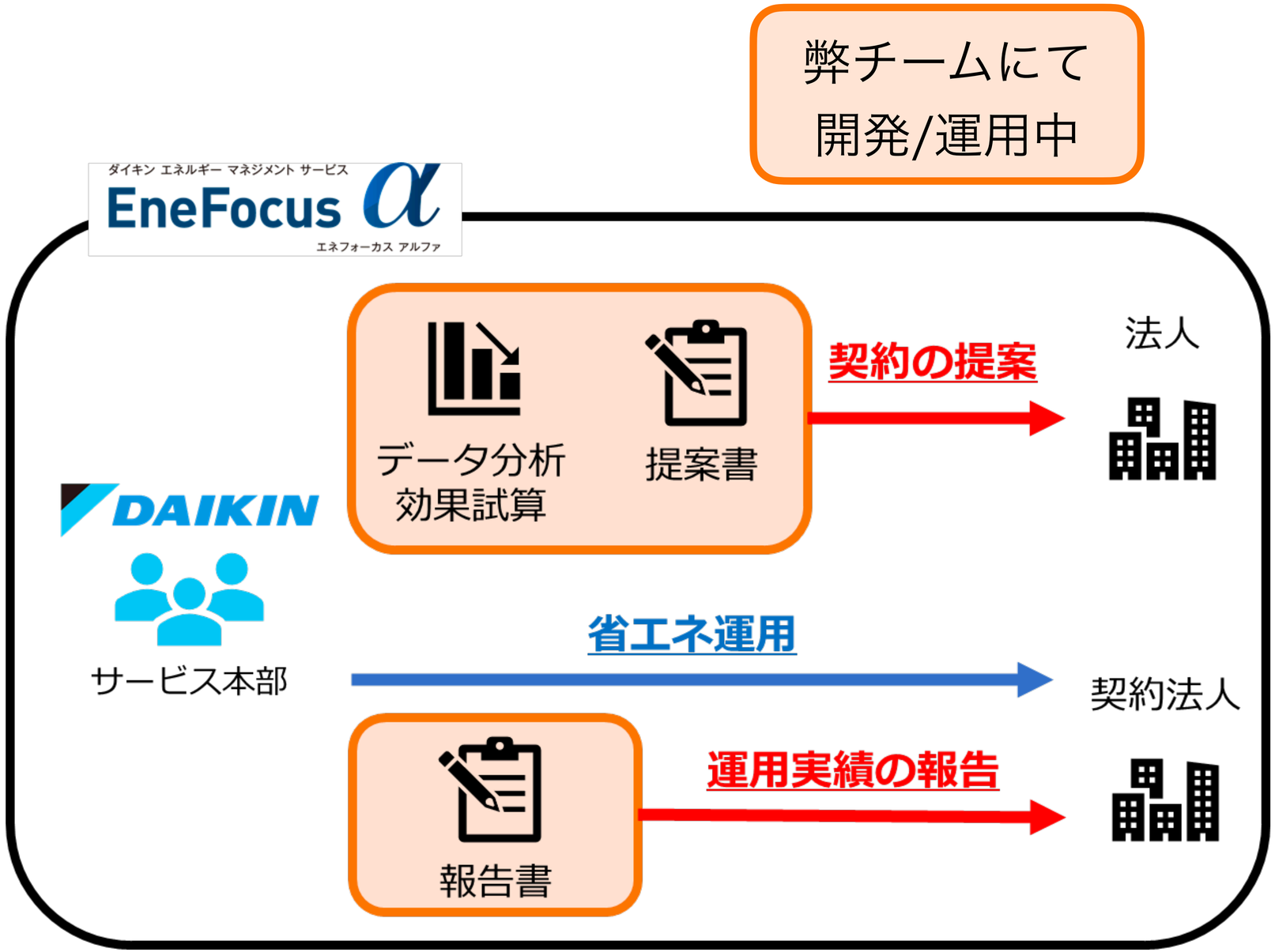


※グラフはイメージです。

システム概要

EneFocus α 支援ツール

- ユーザー数：100人以上
- 物件数：2000件以上
- 設計：マイクロサービス



カオスエンジニアリングを始めた動機

モダナイゼーション起因による障害発生

- インシデント対応のため現状復帰を要求
- 想定手段によるロールバックに失敗
- 翌営業日まで必死に復旧作業

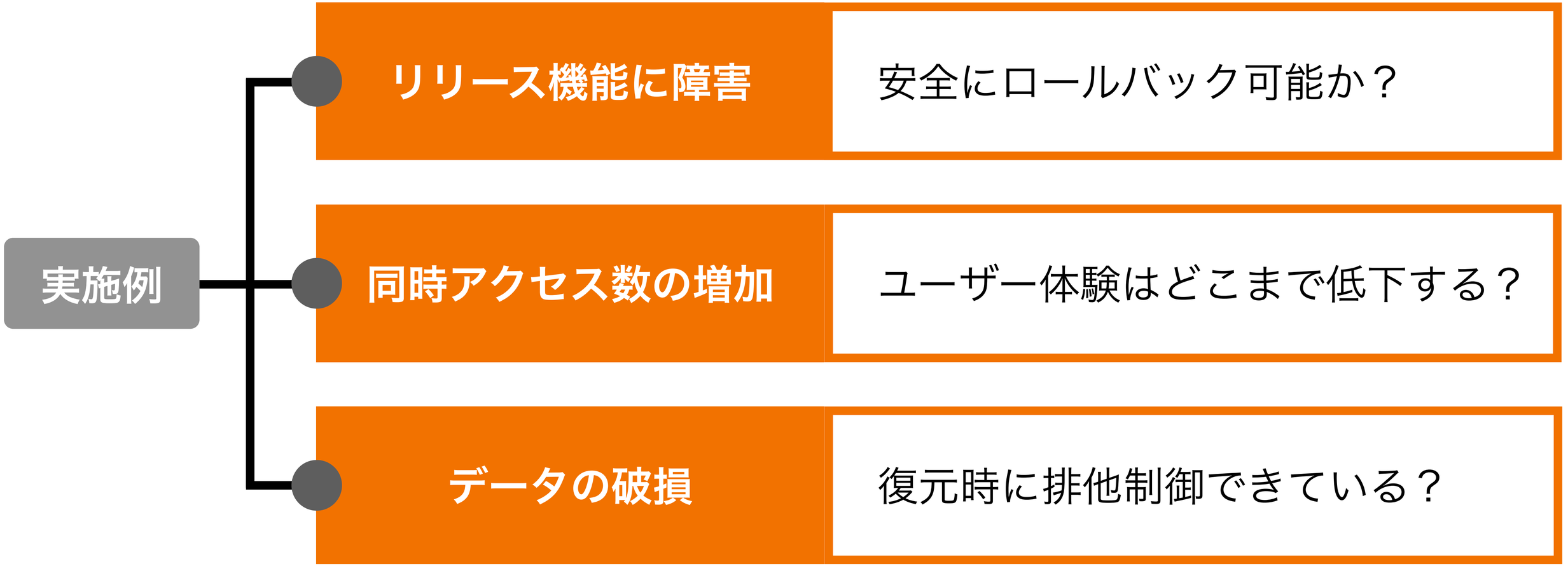
ポストモーテムの実施

- 本番環境特有の仕様
- 復旧テストの重要性
- 実行しなければ分からない事象

Citation：Twitter

復旧テスト DiRT (Disaster Testing)

緊急事態を事前に想定・分析する



自己紹介２

ダイキン工業株式会社

Speaker：森鳩 武史

Team Leader Engineer

@アジャイル内製化チーム

・データサイエンス（3年間）、Ph.D（情報科学）

・アジャイル内製化チーム設立

・システム開発者（5年目）

好きなこと：スクラム/ テスト/ 数理最適化

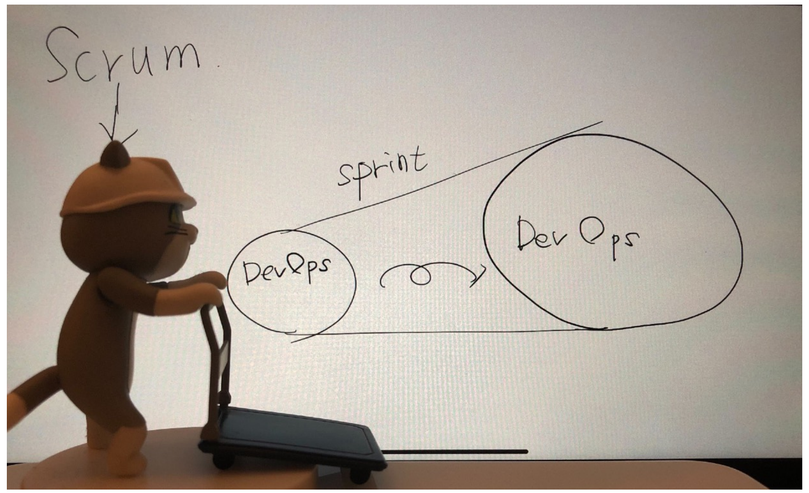


Photo by Morinobu

アジャイル内製化チーム

システム開発の
ゆりかごから墓場まで
コミットしたい

カオスエンジニアリングの導入は困難なのか

	本番に障害を注入して修正	障害を想定して復旧
リスク	High	Low
影響	未知数なものが多数	比較的コントロールが容易
承諾	ステークホルダーの同意が必要	開発者だけで可能

安全なカオスエンジニアリング（復旧テスト）

STEP1 想定手順の確認
（1.定常状態の仮説
5.影響範囲を局所化）

STEP2 リスクの洗い出し
（2.事象を多様化）

STEP3 実験
（3.本番環境で実験）

STEP4 振り返り

Citation: 悪循環ジェネレーター

実例：データ消失時に復元する

STEP1 想定手順の確認

Q. バックアップのデータの存在箇所は？

Q. データの差し替え方法は？

STEP2 リスクの洗い出し

Q. 消失時点とバックアップとの差分は吸収できるか？

Q. 復旧に失敗した時はどうすればいい？

Citation: ふたば☆ちゃんねる

STEP3 実験

STEP4 振り返り

実例：データ消失時に復元する

STEP1 想定手順の確認

STEP2 リスクの洗い出し

STEP3 実験

Q. MTTRが想定より長くならないか？

Q. 利用中のユーザーに影響はないか？

STEP4 振り返り

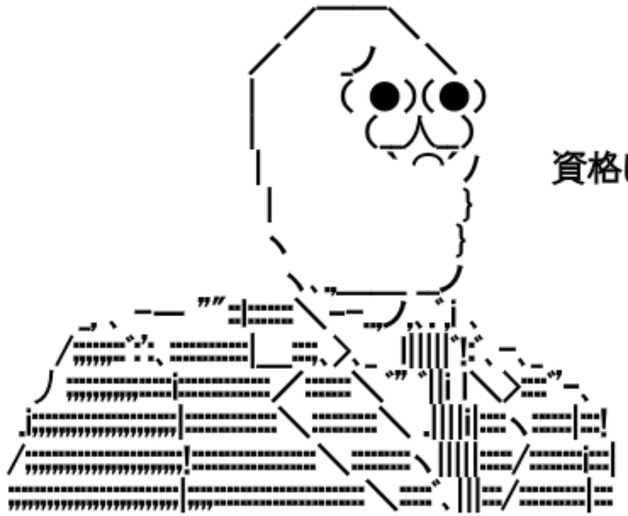
A. 対処すべきリスクをIssue化

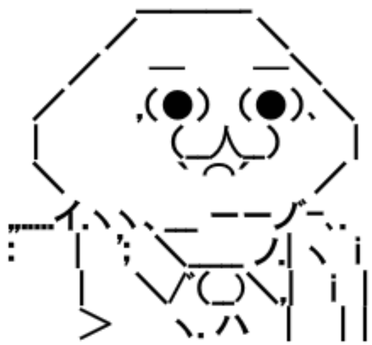
A. 手順の更新

Citation: ふたば☆ちゃんねる

1. 何があっても復旧できる自信

- 潜在リスク項目のリストを作成
  - 各項目の復旧テストを実施
  - 振り返りによる改善
  - リストを更新





Citation: 旧2ch

2. オブザーバビリティの必要性

- 注入された障害を検知できるか？
  - 定常状態との差分を観測する
- 得られる情報のみでデバッグ可能か？
  - オブザーバビリティを改善する

Citation: https://learning.oreilly.com/

カオスエンジニアリングの効果は可測性が高める

3. 自動的に障害を発生させて復旧

1	障害を注入	安全にロールバック可能か？
2	障害を検知	障害は可測か？検知まで何分かかる？
3	復旧作業	デバッグは民主化されているか？
4	潜在リスクの抽出	実験を通して新たなリスクは見つけた？

4. 潜在リスクに対する対抗

知っている・理解している

テスト

Jest

pytest

知らない・理解している

Citation: ポプテピピック

知っている・理解していない

モニタリング

Alarms

Dashboards

知らない・理解していない

カオス
エンジニアリング

Observability

RUM

安心した週末を迎えるために

1. 障害を想定した復旧テスト

   障害が起きても復旧できる自信を得る

2. オブザーバビリティの向上

   平常時からの逸脱を可測に

3. 実験的な障害の注入

   拡大するカオス領域に対応

4. 継続的な学習と対応

   実験毎に強くなる

Citation: Reddit

Photo by Morinobu

ご清聴
ありがとうございました