aws

DevAx

DevAx::Connect

雰囲気でモダン開発手法の実践をしている人のためのCI/CD再入門

Yuji Nomura
Solutions Architect
Amazon Web Services Japan G.K.

2022/4/7

自己紹介

名前：野村　侑志　　@ugnomura

所属：技術統括本部 /
　　インダストリソリューション部

略歴：AWSソリューションアーキテクト
　← 生保会社のSA・インターフェース開発リード
　← 米国で日系製造業の現地販社のインターフェース開発リード
　← 製造業で複合機のプログラム開発
　← 音楽大学（打楽器）

興味：API, コンテナ, Haskell, 圏論, 競プロ

好きなAWSサービス：Amazon SQS

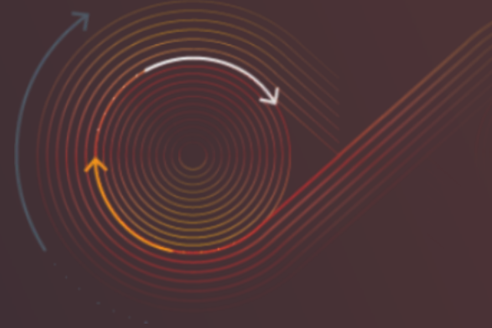
aws DevAx

本日の目的

- 今後自分の組織のソフトウェア開発手法をどのように発展させれば良いかの指針を見つけていただく

- 次回以降の各論に備えてCI/CDの勘所を思い出していただく

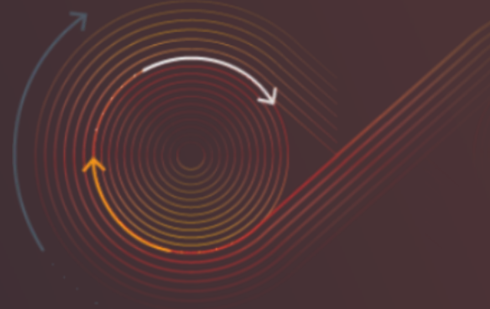
Agenda



- CI/CDとは
- CI/CDの目的
- CI/CDの導入
- まとめ

aws DevAx

CI/CDの導入

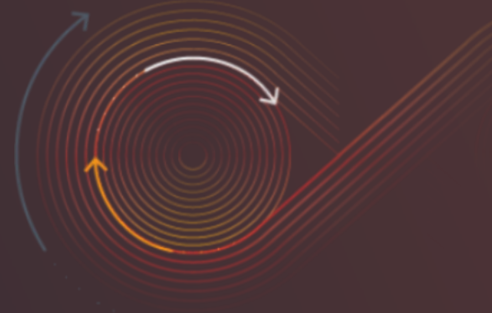


- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

aws DevAx

CI/CDとは

CI/CDとは



継続的インテグレーション
Continuous Integration（CI）

＋

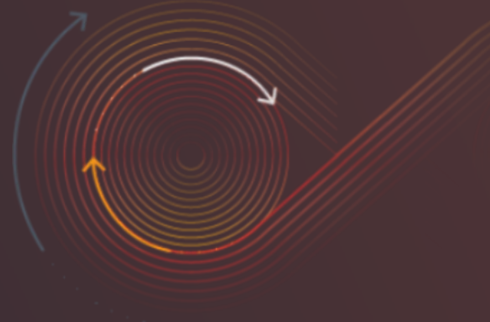
継続的デリバリー/デプロイ
Continuous Delivery/Deployment（CD）

aws DevAx

継続的？
インテグレーション？
デリバリー？
デプロイ？

aws DevAx

リリースプロセスのステージ



Source	Build	Test	Production
• ソースコードのチェックイン • コードのピアレビュー	• コードのコンパイル • ユニットテスト • スタイルチェッカー • コンテナイメージ、関数デプロイパッケージの作成	• 周辺システムとの統合テスト • 負荷テスト • UIテスト • セキュリティテスト	• 本番環境へのデプロイ • エラーを素早く検知するための本番環境のモニタリング

aws DevAx

リリースプロセスのステージ

CI：継続的インテグレーション

CD：継続的デリバリー

別のCD：継続的デプロイ

aws DevAx

CI/CDの目的

誰かが良いアイディアを思いついたとき、できるだけ早くユーザーに届けるためのソフトウェア開発手法

- ソフトウェアのビルド・デプロイ・テスト・リリースというプロセスのあらゆる部分が関係者全員から見えるようにし、共同作業をやりやすくすること
- フィードバックを改善し、プロセスにおいてできる限り早い時期に問題が特定されて解決されるようにすること
- ソフトウェアの任意のバージョンを任意の環境に対して、完全に自動化されたプロセスを通じて好きなようにデプロイできるようにすること

aws DevAx

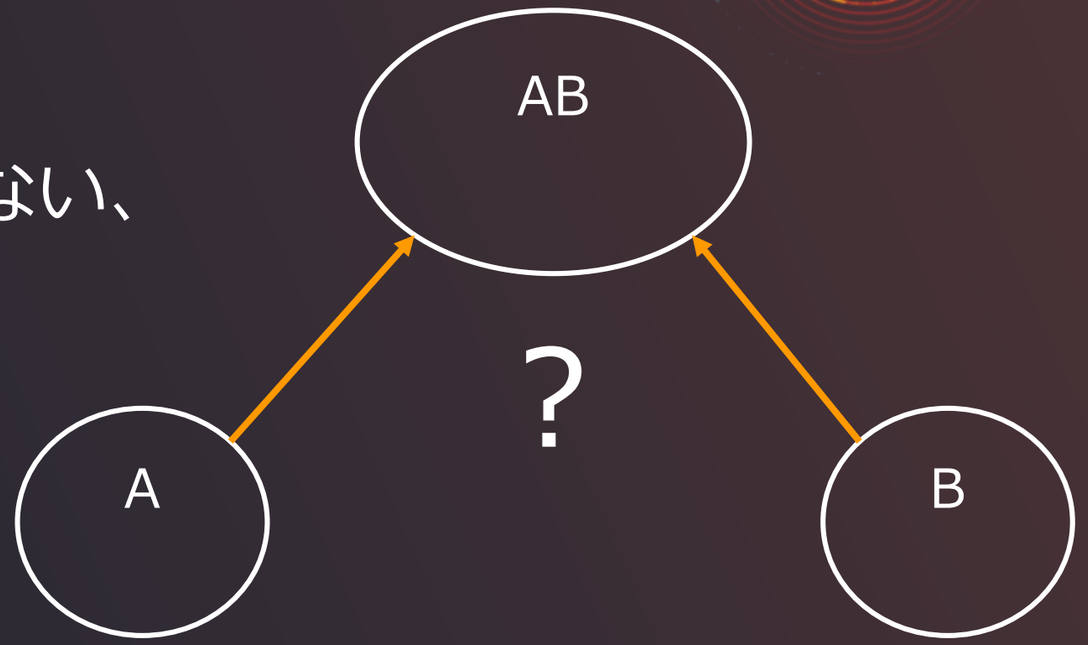
継続的インテグレーション（CI）

Continuous：[kəntínjuəs] 形容詞

連続的な、切れ目のない、途切れない、
継続的な

Integration：[ìnṭəgréɪʃən] 名詞

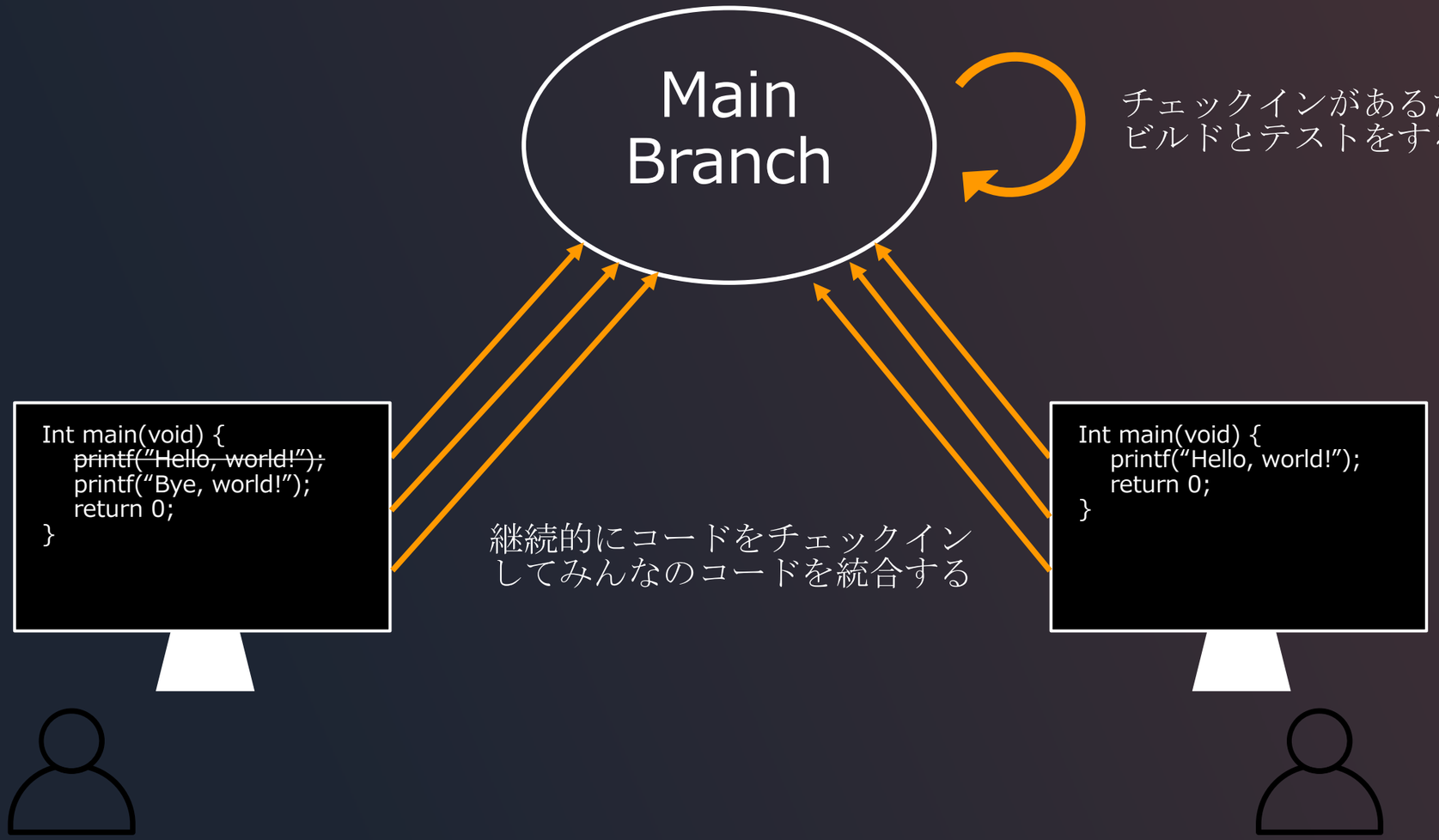
統合、完成、調整



何を継続的に統合したいのか？

aws DevAx

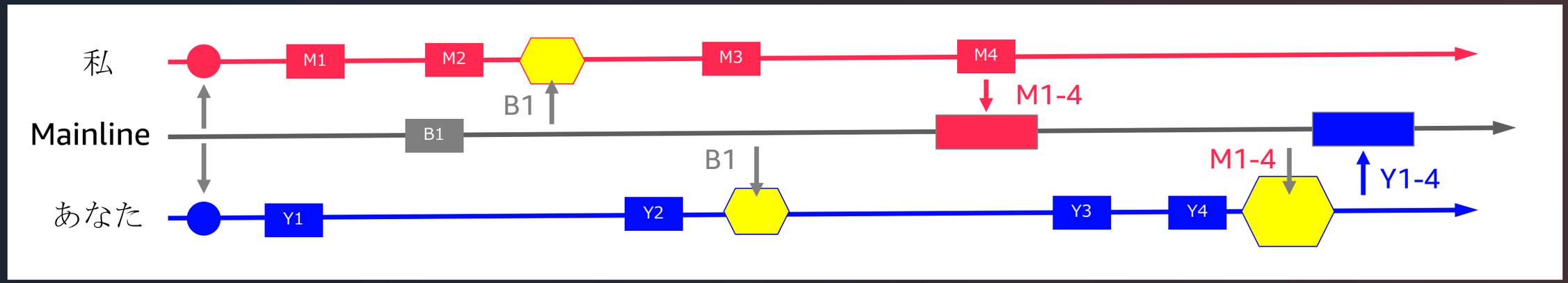
継続的インテグレーション



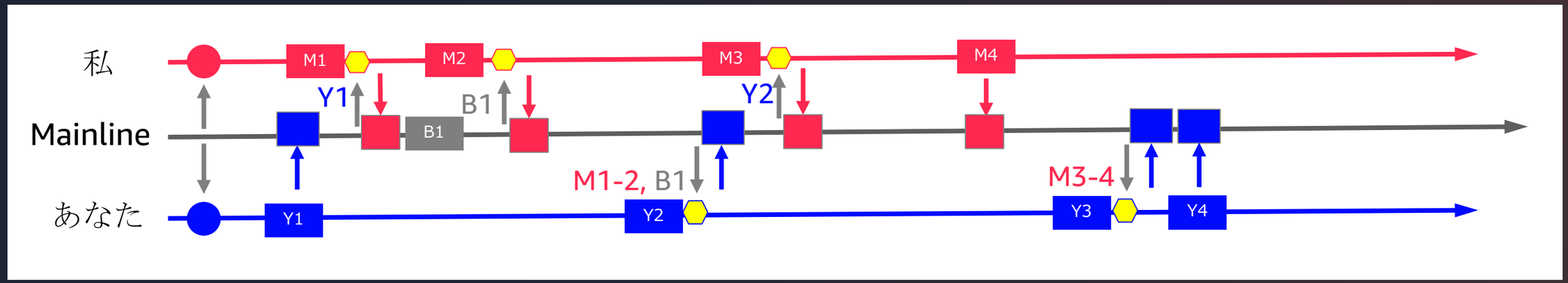
aws DevAx

高頻度な統合の必要性

低頻度



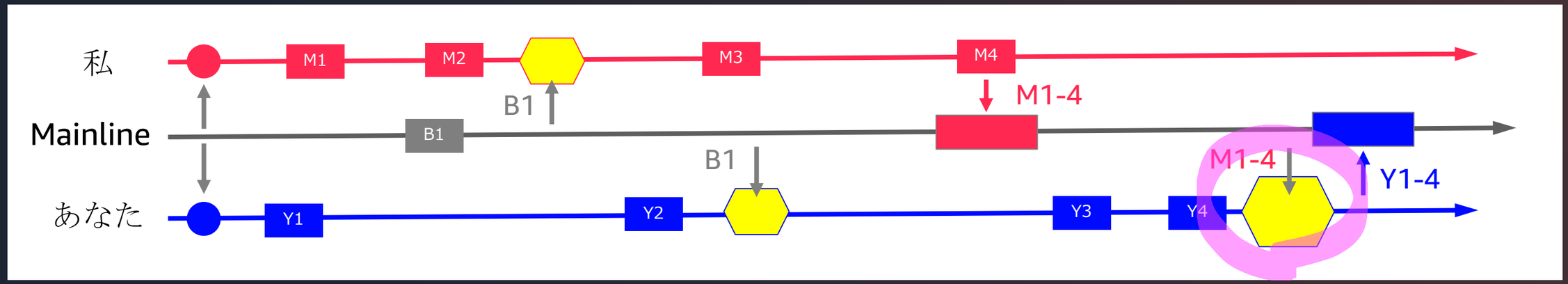
高頻度



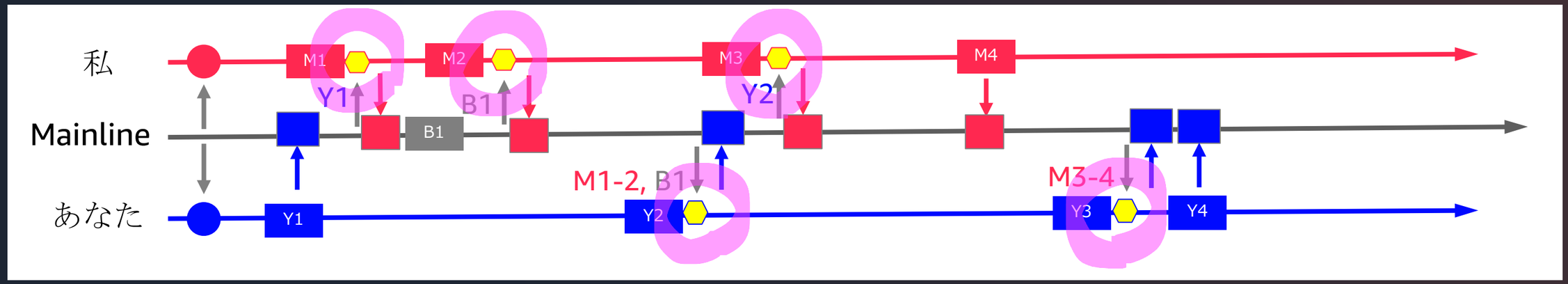
aws DevAx

高頻度な統合の必要性

低頻度



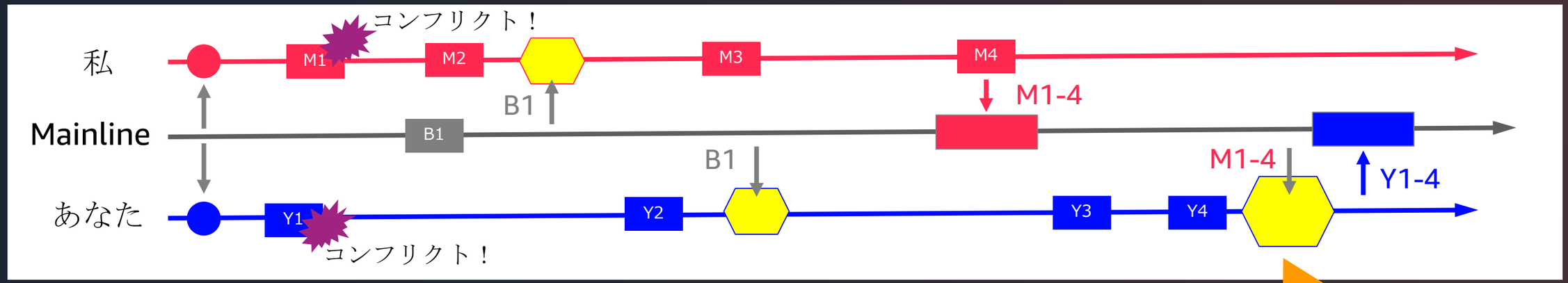
高頻度



aws DevAx

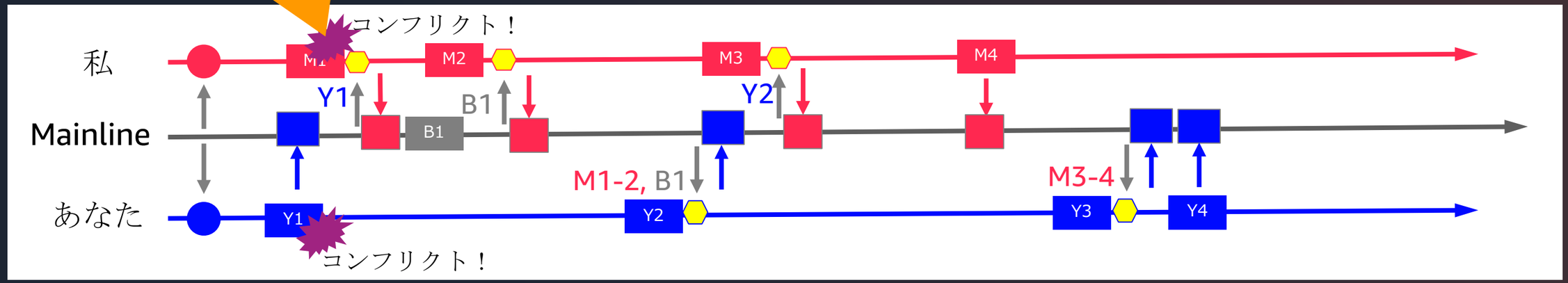
高頻度な統合の必要性

低頻度



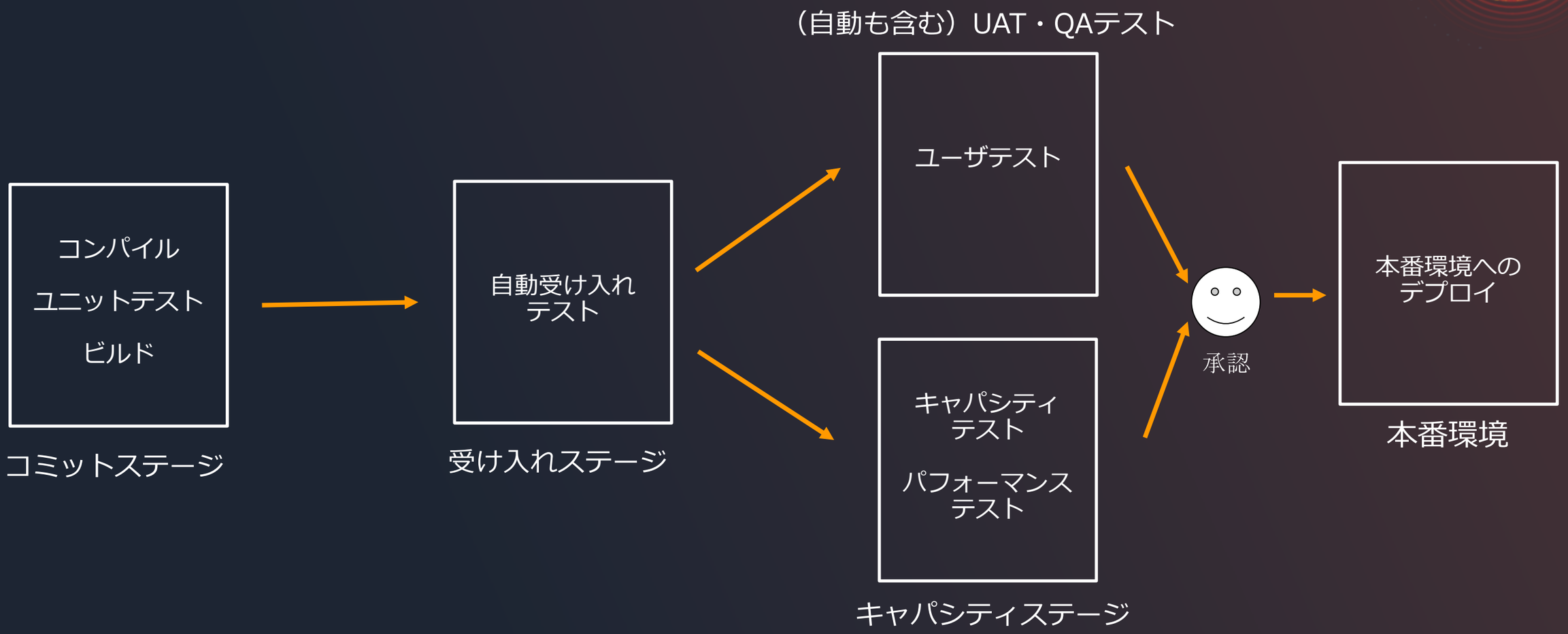
高頻度

ここで気づく



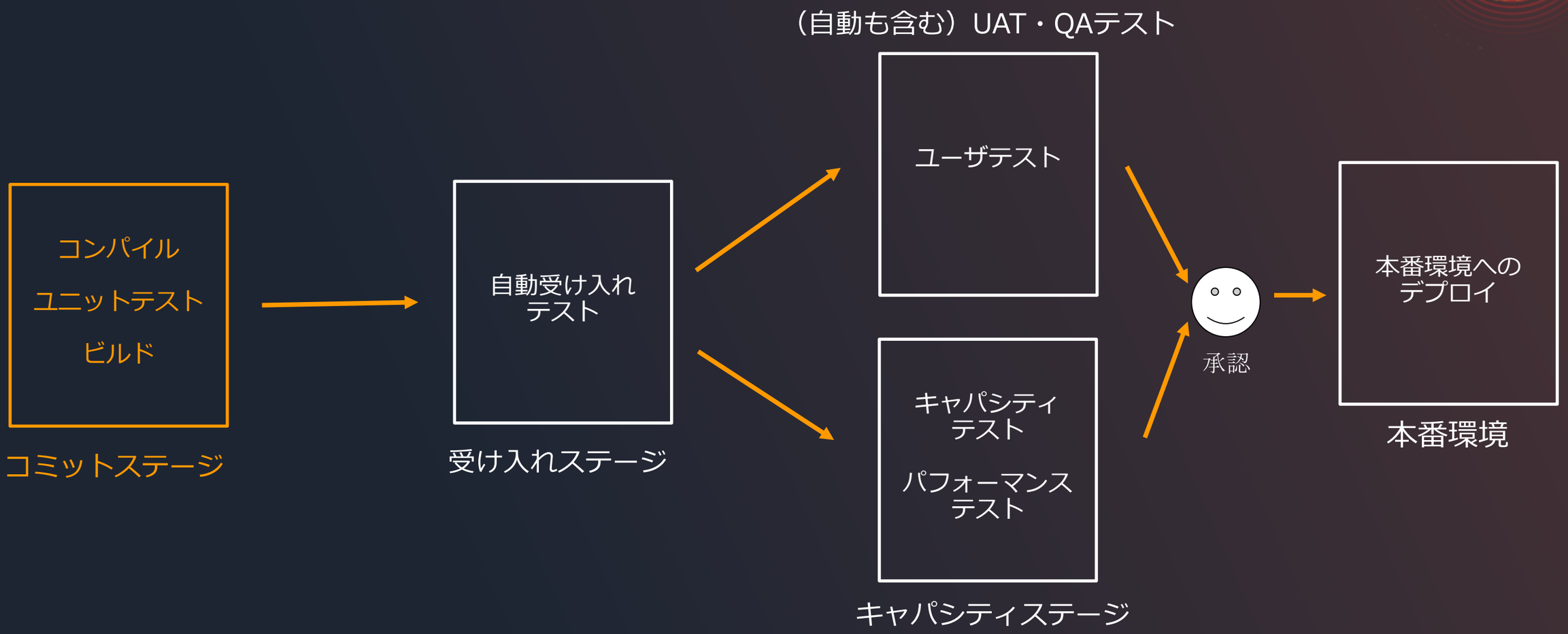
aws DevAx

継続的デリバリー：デプロイメントパイプライン



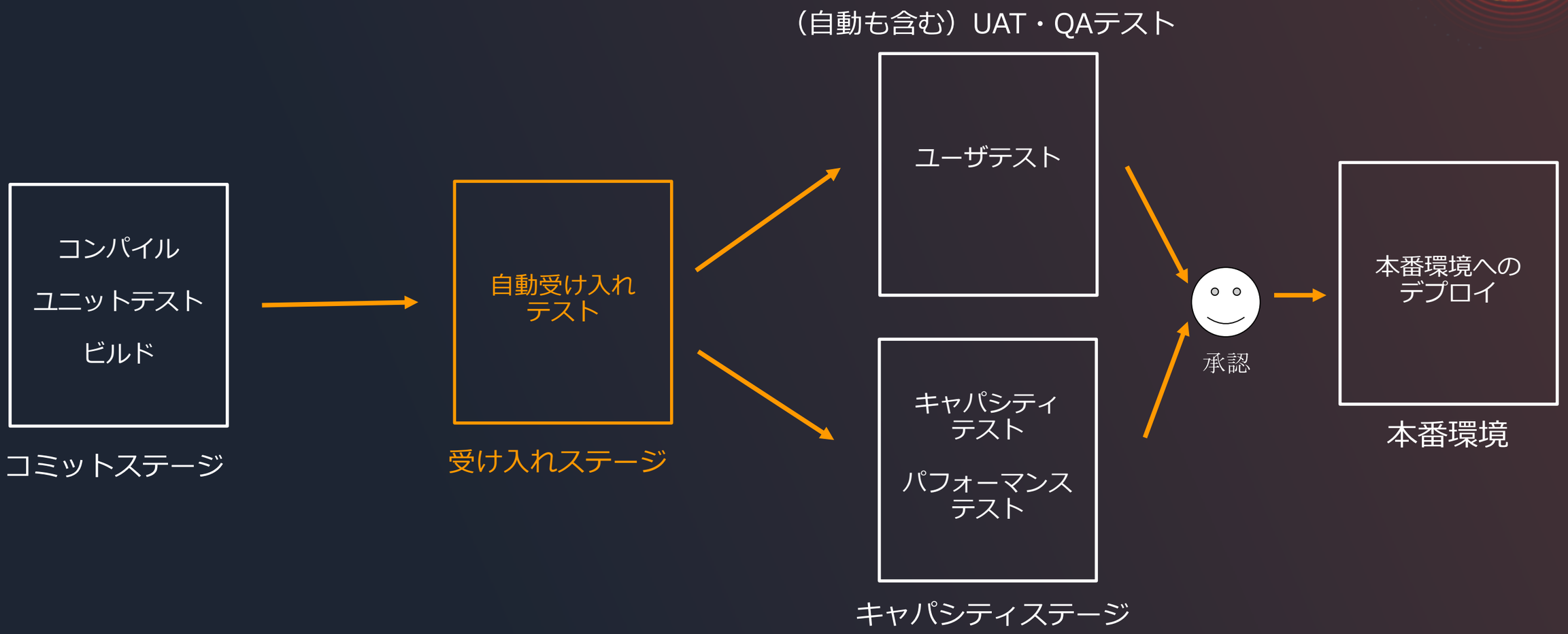
aws DevAx

継続的デリバリー：デプロイメントパイプライン



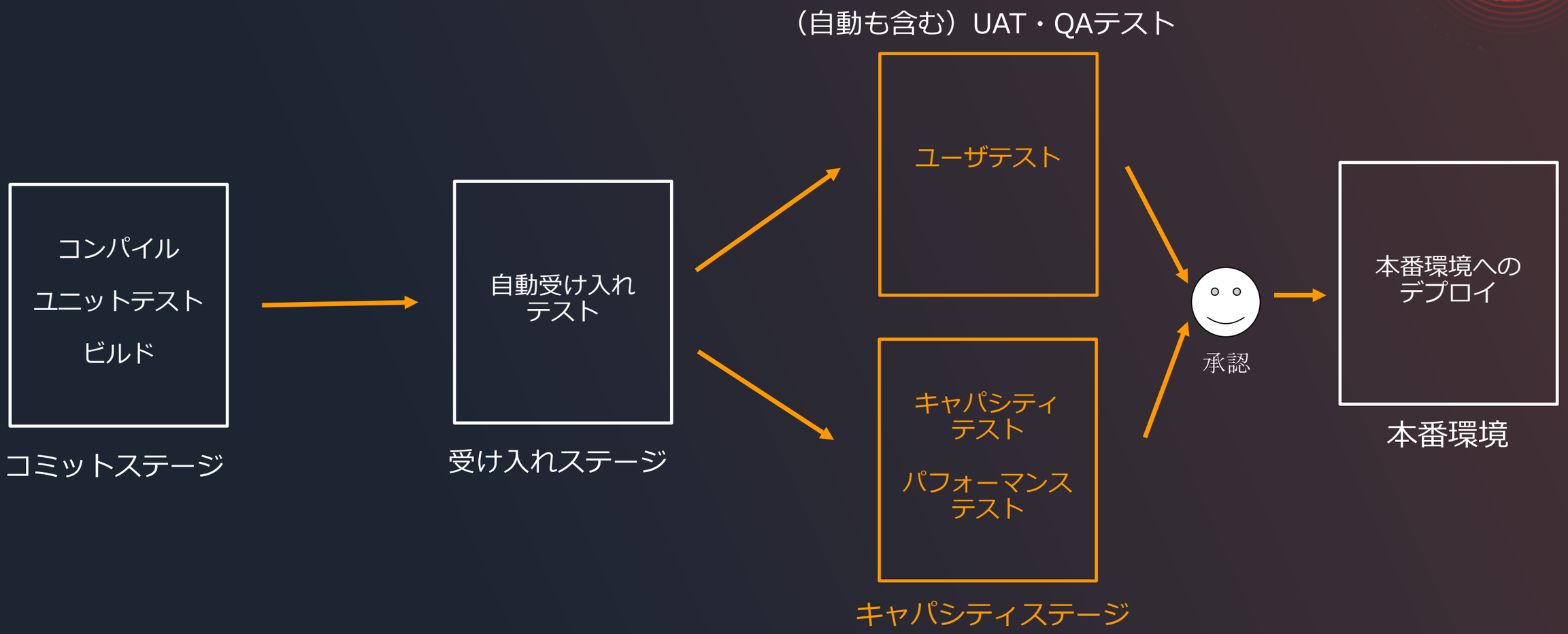
aws DevAx

継続的デリバリー：デプロイメントパイプライン



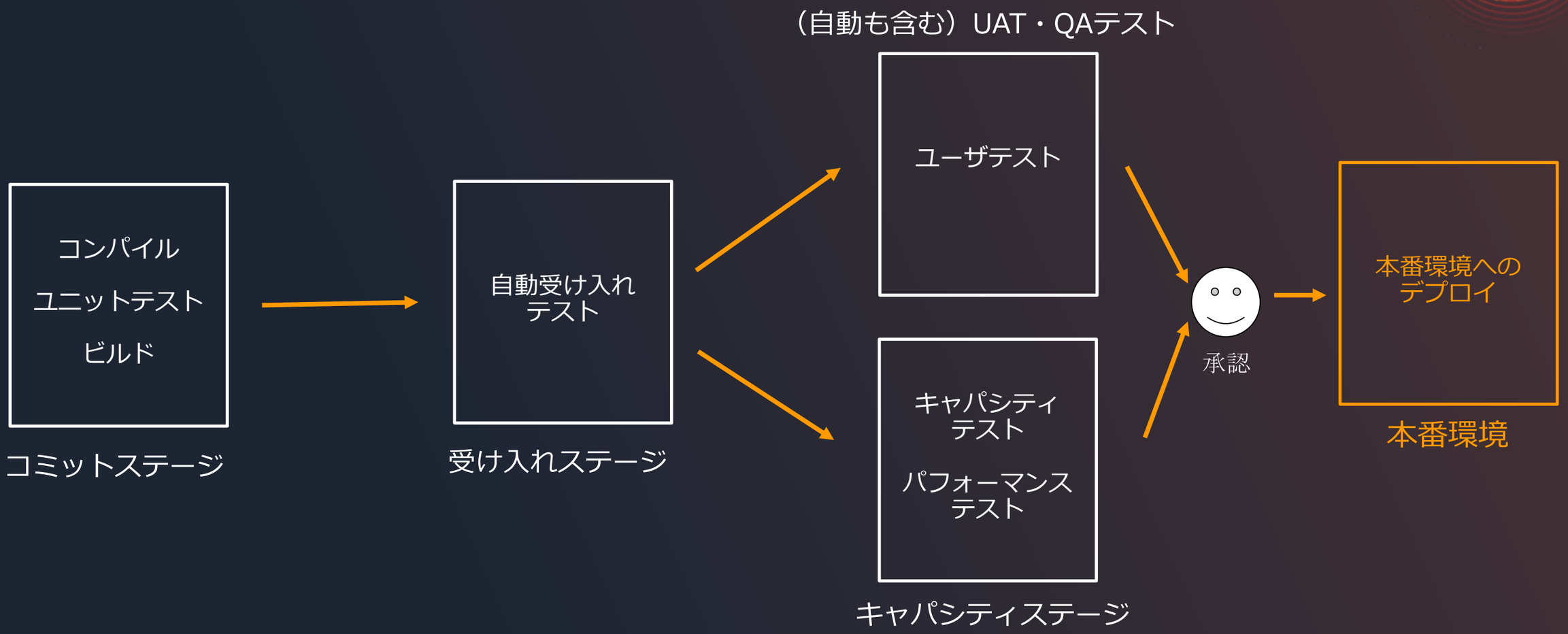
aws DevAx

継続的デリバリー：デプロイメントパイプライン



aws DevAx

継続的デリバリー：デプロイメントパイプライン



aws DevAx

継続的デプロイ

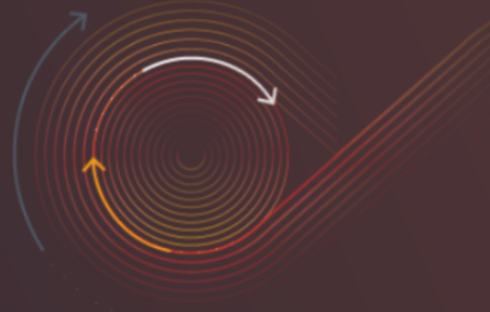
CI：継続的インテグレーション

CD：継続的デリバリー

別のCD：継続的デプロイ

aws DevAx

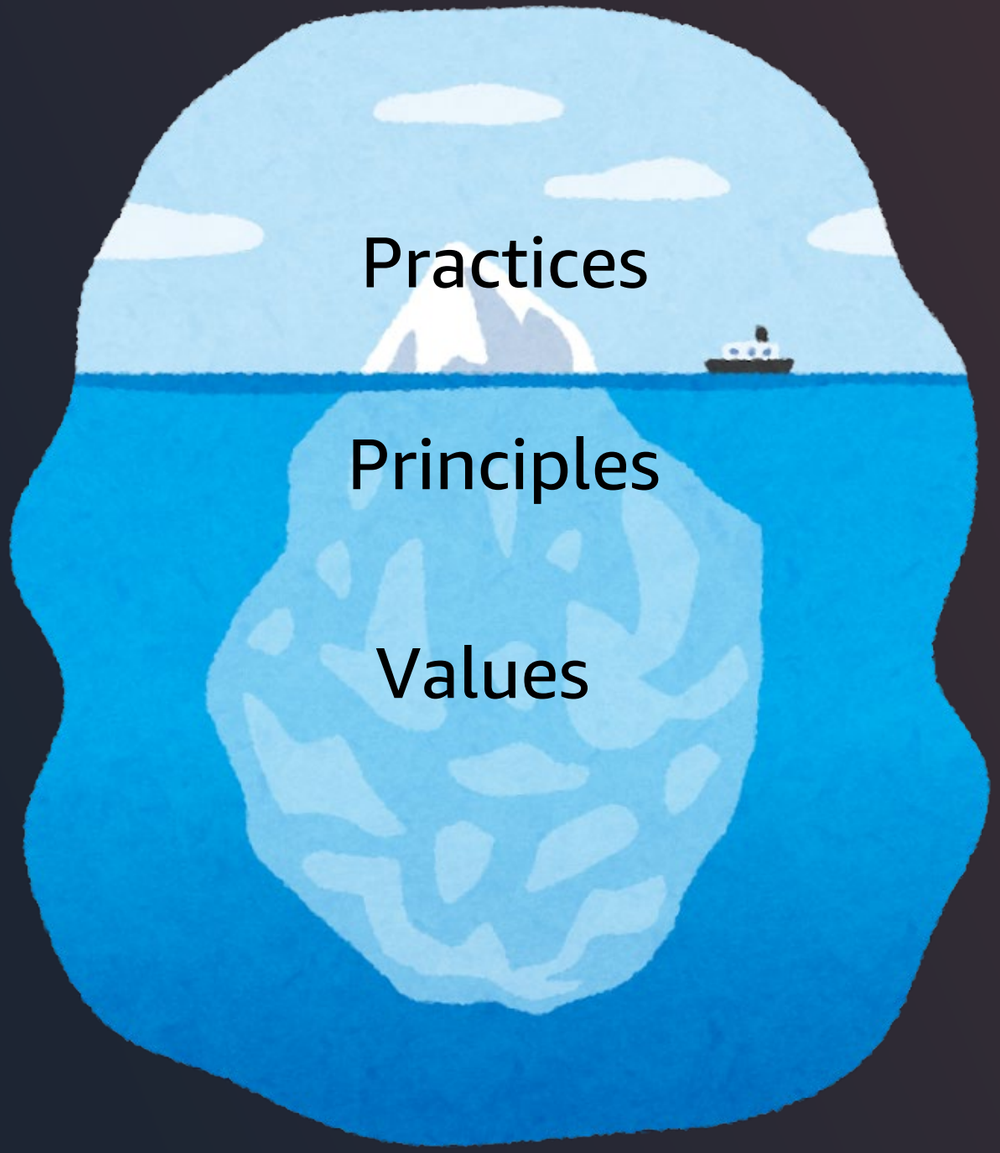
CI/CDとは



- 継続的インテグレーション（CI）
  - 継続的にコードをチェックインして最新のコードがマージされた状態にしておくこと
  - 安全にマージを行うために自動ビルドや自動テストを構築すること
- 継続的デリバリー/デプロイ（CD）
  - デプロイメントパイプラインを構築すること
  - パイプラインの可視化をすること

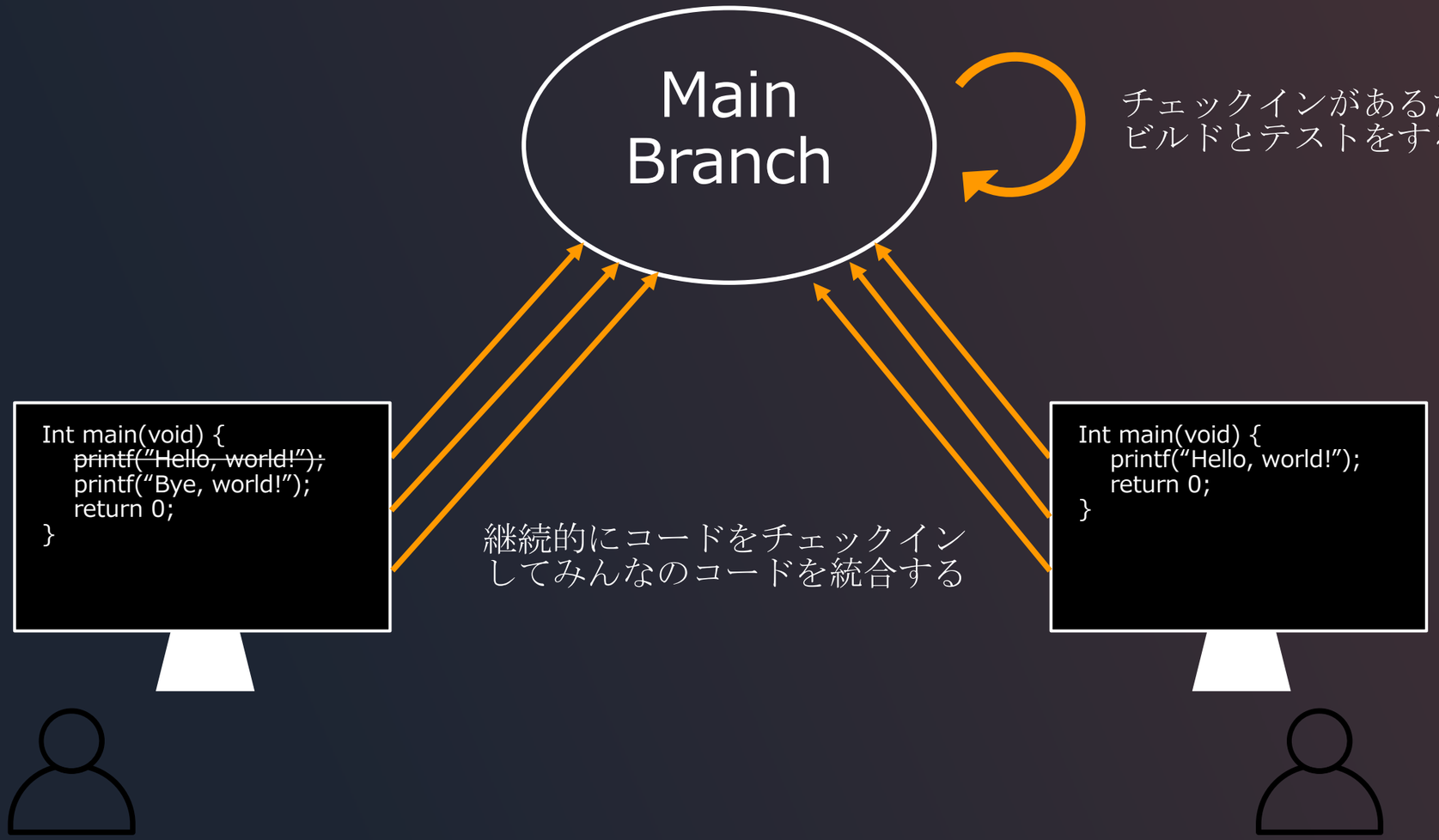
CI/CDの目的

価値・原則・プラクティス
Values/Principles/Practices



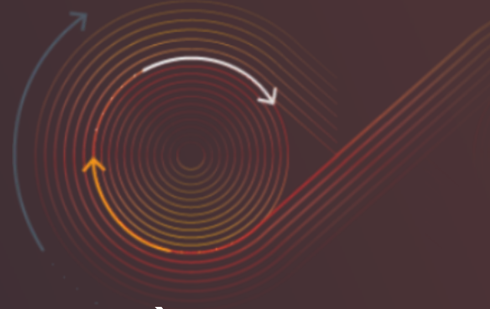
aws DevAx

継続的インテグレーション



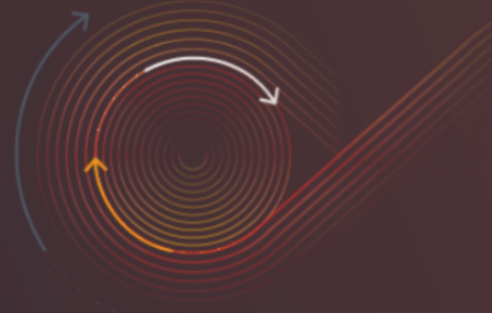
aws DevAx

参考：継続的インテグレーションの歴史



- 初出:公式には「Booch法：オブジェクト指向分析と設計」（1991）という本のなかでオブジェクト指向でのソフトウェア開発手法の一つとして提案された
- その後Extreme Programmingでコンセプトが採用されてほぼ現在の形になった（一般にはこちらが初出とも言われる）
- 2007年に”Continuous Integration: Improving Software Quality and Reducing Risk”という本が出版されている
- CIツールであるHudson(2008~) / Jenkins(2011~)が出てきたり、CDの概念が定式化されたり、（私の観測範囲の）世の中の認知が高まったのが2010年前後

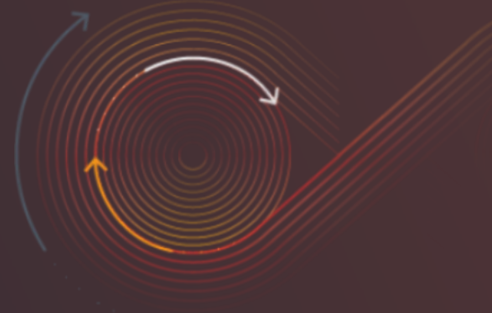
Extreme Programming（XP）



- XPは、軽量である
- XPは、ソフトウェア開発の制約に対応するための方法論である
- XPは、あらゆる規模のチームに使える
- XPは、曖昧で急速に変化する要件に対応する

変化する顧客の要求への対応力を高めることが目的

XPの価値・原則・プラクティス



価値

コミュニケーション・シンプリシティ・フィードバック
勇気・リスペクト

原則

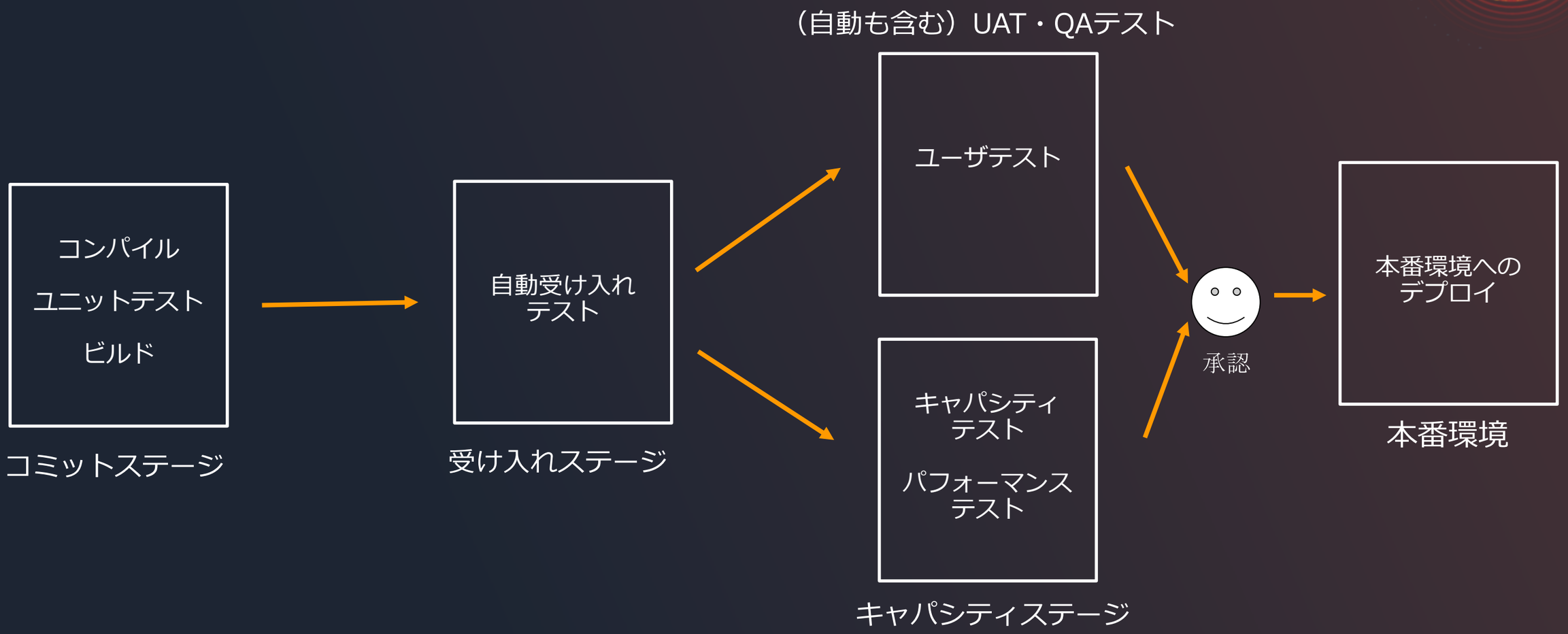
人間性・経済性・相互利益・自己相似性
改善・多様性・振り返り・流れ・機会
冗長性・失敗・品質・ベイビーステップ
責任の引き受け

プラクティス

全員同席・チーム全体・情報満載のワークスペース・生き生きとした仕事
ペアプログラミング・ストーリー・週次サイクル・四半期サイクル・ゆとり
10分ビルド・継続的インテグレーション・テストファーストプログラミング
インクリメンタルな設計

aws DevAx

継続的デリバリー：デプロイメントパイプライン



aws DevAx

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- From “Principles behind the Agile Manifesto”

私たちは、ソフトウェア開発の実践
あるいは実践を手助けをする活動を通じて、
よりよい開発方法を見つけだそうとしている。
この活動を通して、私たちは以下の価値に至った。
プロセスやツールよりも**個人と対話**を、
包括的なドキュメントよりも**動くソフトウェア**を、
契約交渉よりも**顧客との協調**を、
計画に従うことよりも**変化への対応**を、

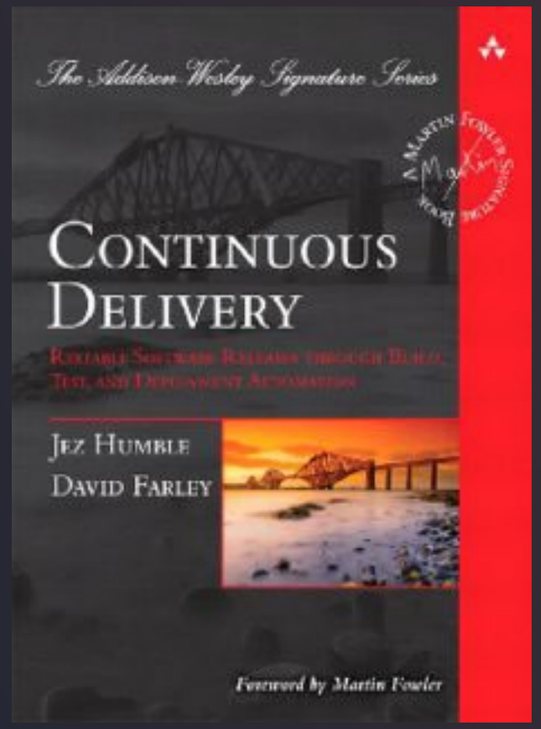
価値とする。すなわち、左記のことがらに価値があることを
認めながらも、私たちは右記のことがらにより価値をおく。

←※有名な方はこちら

aws DevAx

継続的デリバリー

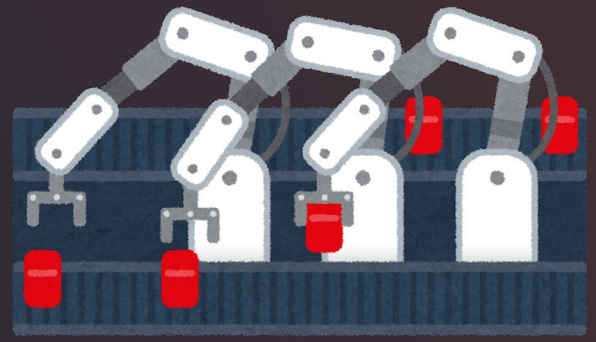
- 継続的デリバリーという言葉が（おそらく）正式に現在の概念として使われはじめた本
- 2010年出版



aws DevAx

継続的デリバリーの価値

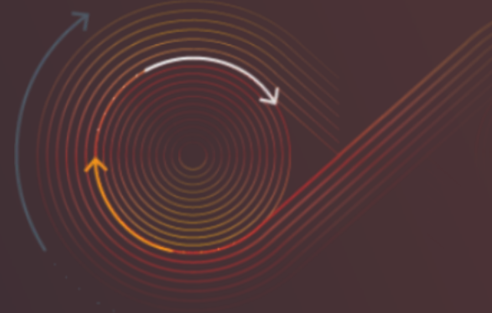
Automation



Collaboration

aws DevAx

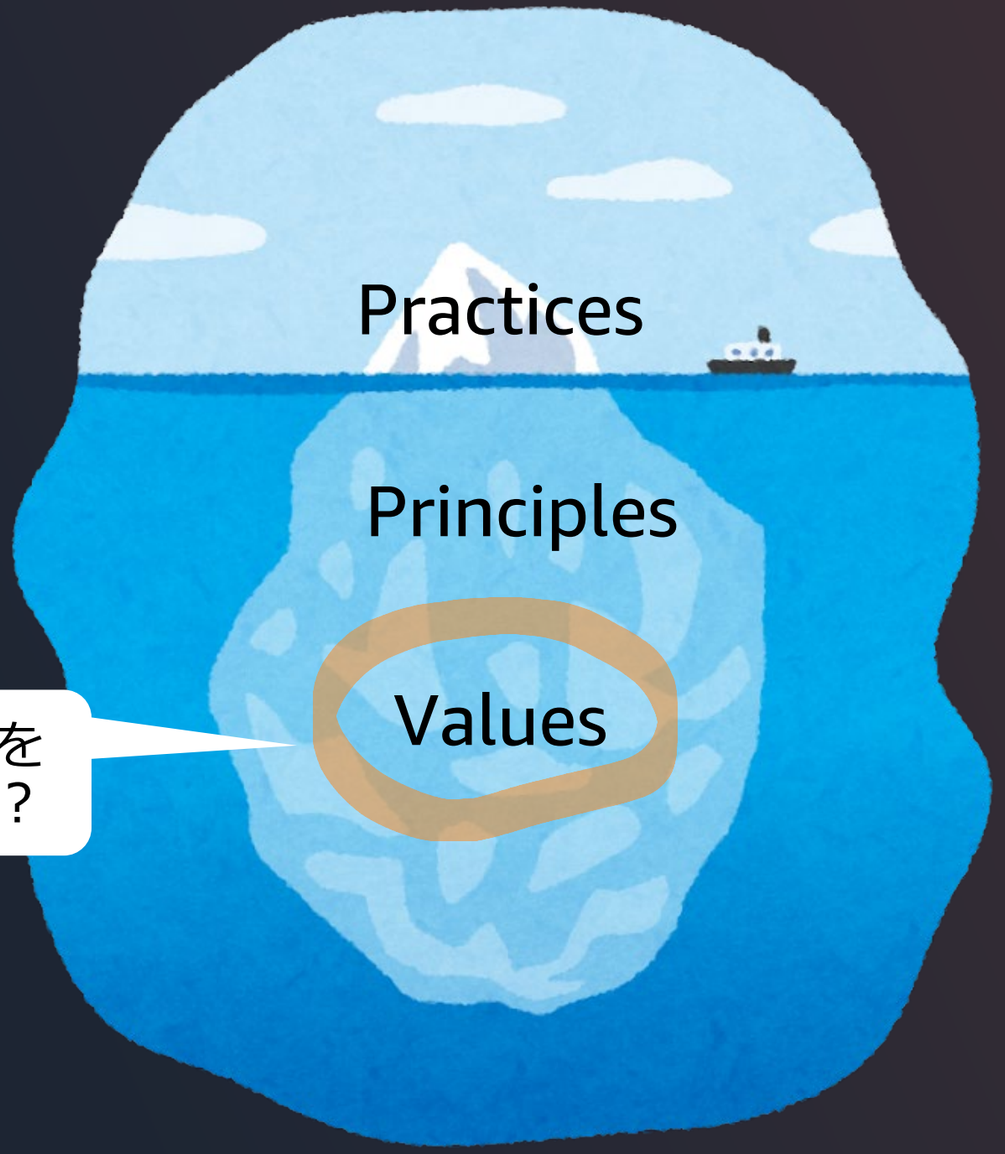
継続的デリバリーの原則



- 品質を作り込む
- 小さなバッチ単位で作業する
- コンピュータは繰り返し作業を行い、人間は問題を解決する
- 絶え間なく継続的な改善を繰り返す
- 全員が責任者である

参考：Principles of Continuous Delivery (https://continuousdelivery.com/principles/)

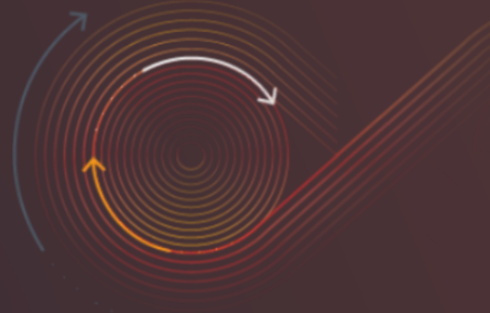
価値・原則・プラクティス



あなたの組織はなにを
価値としていますか？

aws DevAx

アジャイル開発宣言



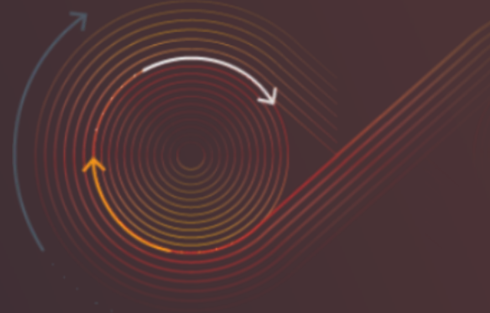
私たちは、ソフトウェア開発の実践
あるいは実践を手助けをする活動を通じて、
よりよい開発方法を見つけだそうとしている。
この活動を通して、私たちは以下の価値に至った。

プロセスやツールよりも**個人と対話**を、
包括的なドキュメントよりも**動くソフトウェア**を、
契約交渉よりも**顧客との協調**を、
計画に従うことよりも**変化への対応**を、

価値とする。すなわち、左記のことがらに価値があることを
認めながらも、私たちは右記のことがらにより価値をおく。

aws DevAx

ここまで：CI/CDとその目的

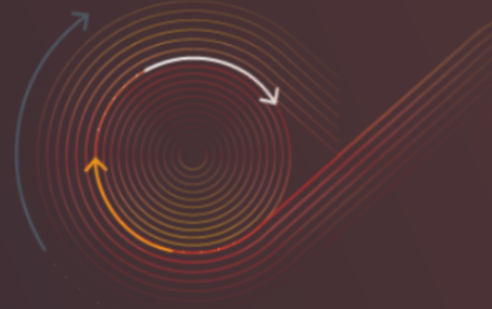


- 継続的インテグレーション（CI）
  - 継続的にコードをチェックインして最新のコードがマージされた状態にしておくこと
  - 安全にマージを行うために自動ビルドや自動テストを構築すること

- 継続的デリバリー/デプロイ（CD）
  - デプロイメントパイプラインを構築すること
  - パイプラインの可視化をすること

- CI/CDの目的
  - 価値・原則・プラクティス
  - 組織によって重要視する価値は異なる

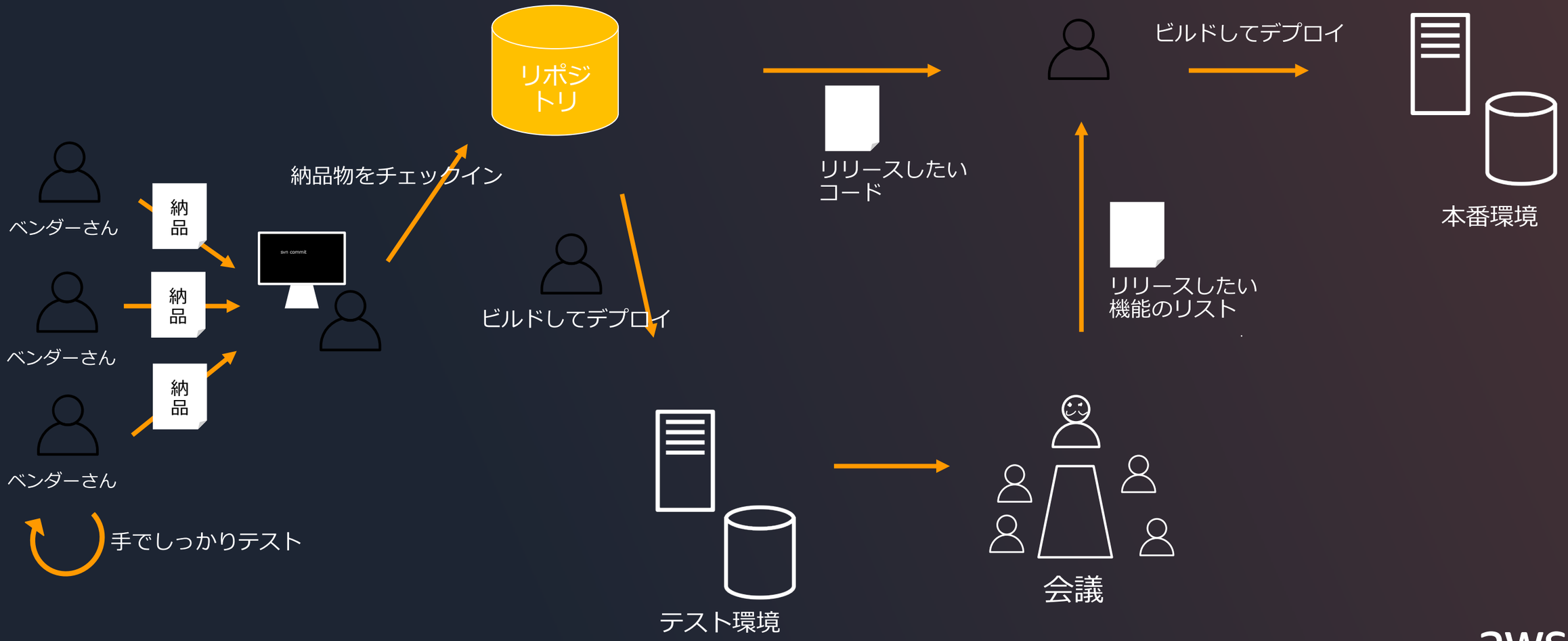
ここからはもう少し具体的な話を



- うちの環境にちゃんとフィットするのかまだよくわからない
- 前一回やってみたんだけど挫折した
- 一応CI/CDやっているけれど正しいのかわからない

CI/CDの導入

例：とある企業の開発アプローチ

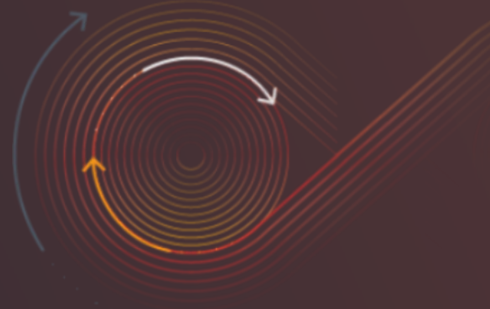


aws DevAx

最近の担当者の悩み

- 本番環境でバグがよく見つかる
- ビルドが通らない
- テストではうまく行ったのに本番で動かない
- リリースにやたら時間がかかってダウンタイムが長い
- 開発をお願いしてから本番環境に行くまでに時間がかかる

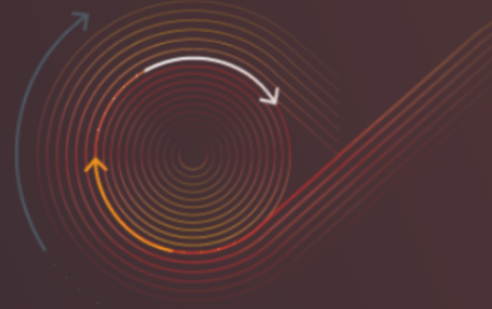
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

aws DevAx

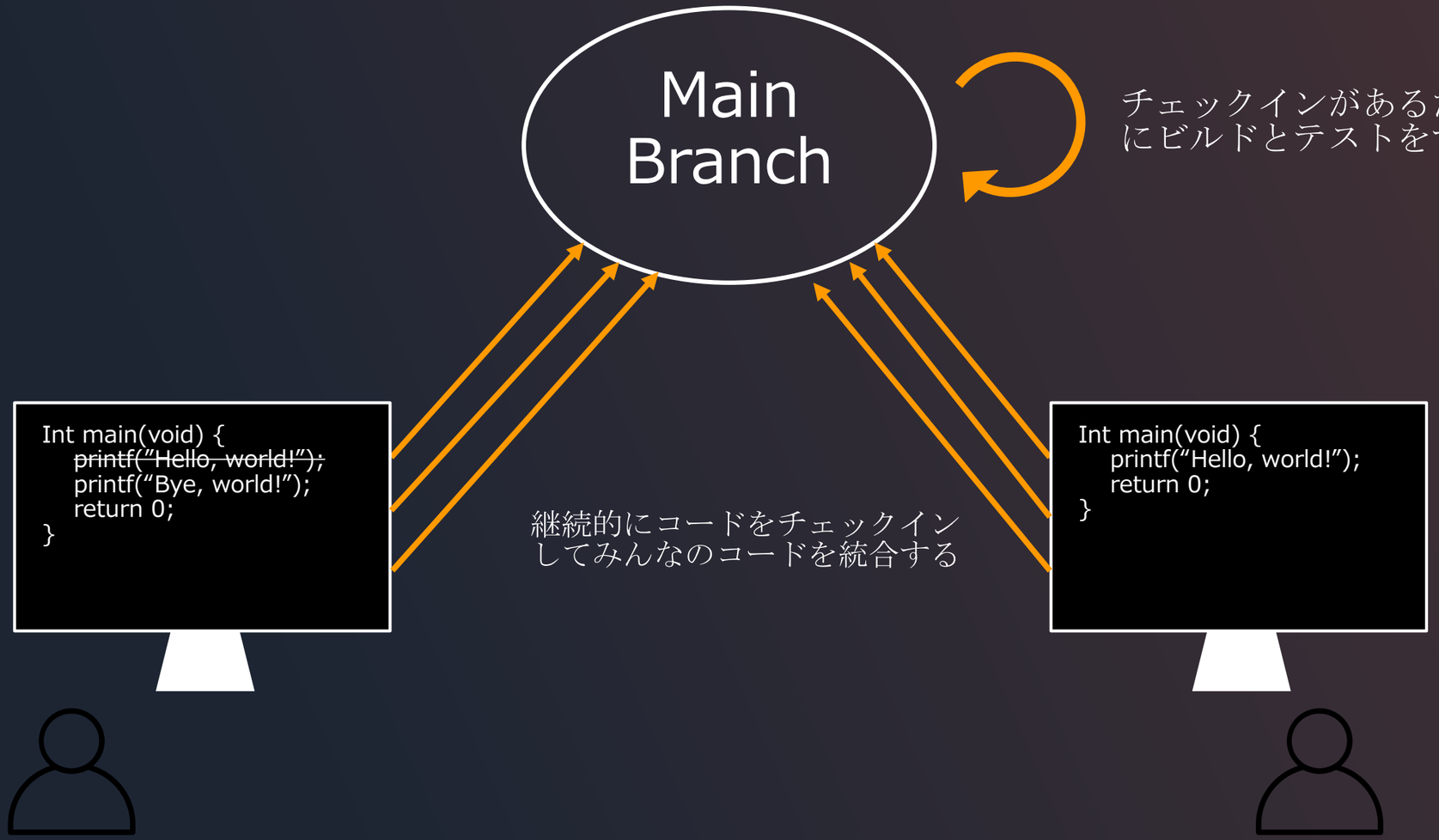
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

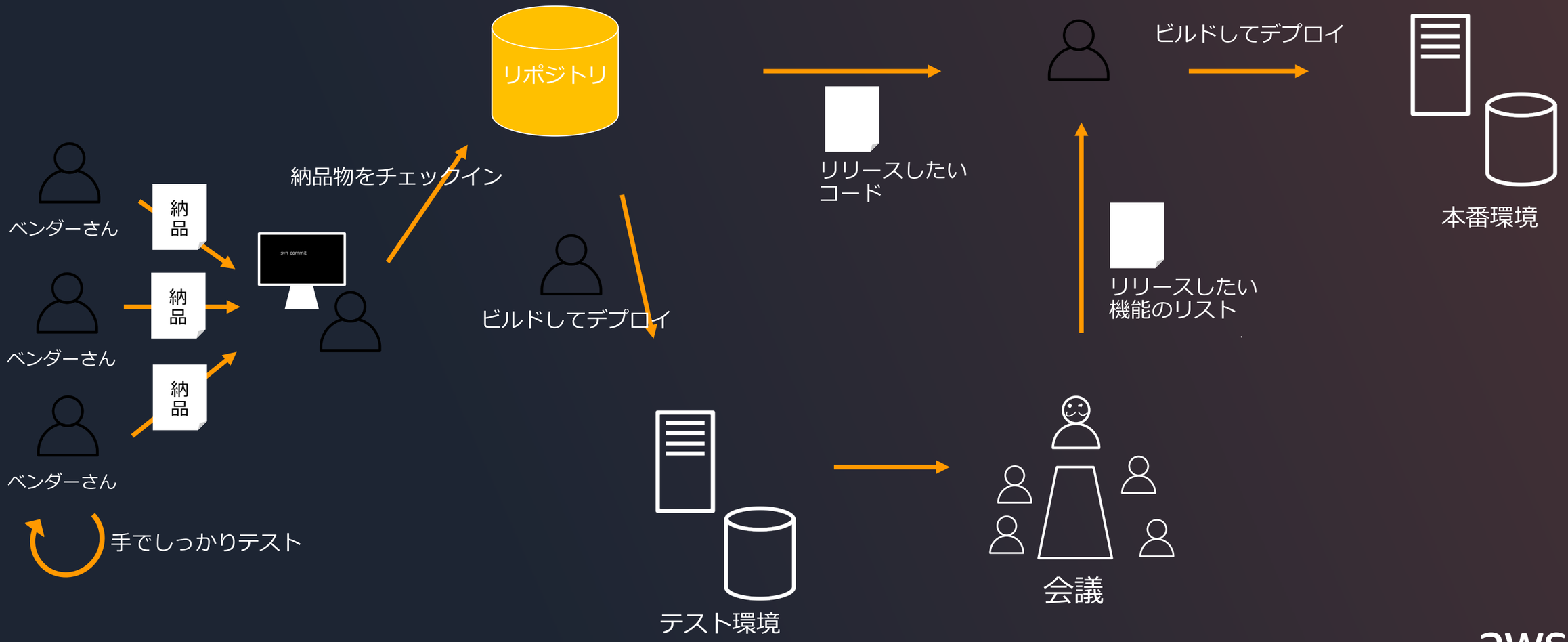
aws DevAx

継続的インテグレーション（CI）



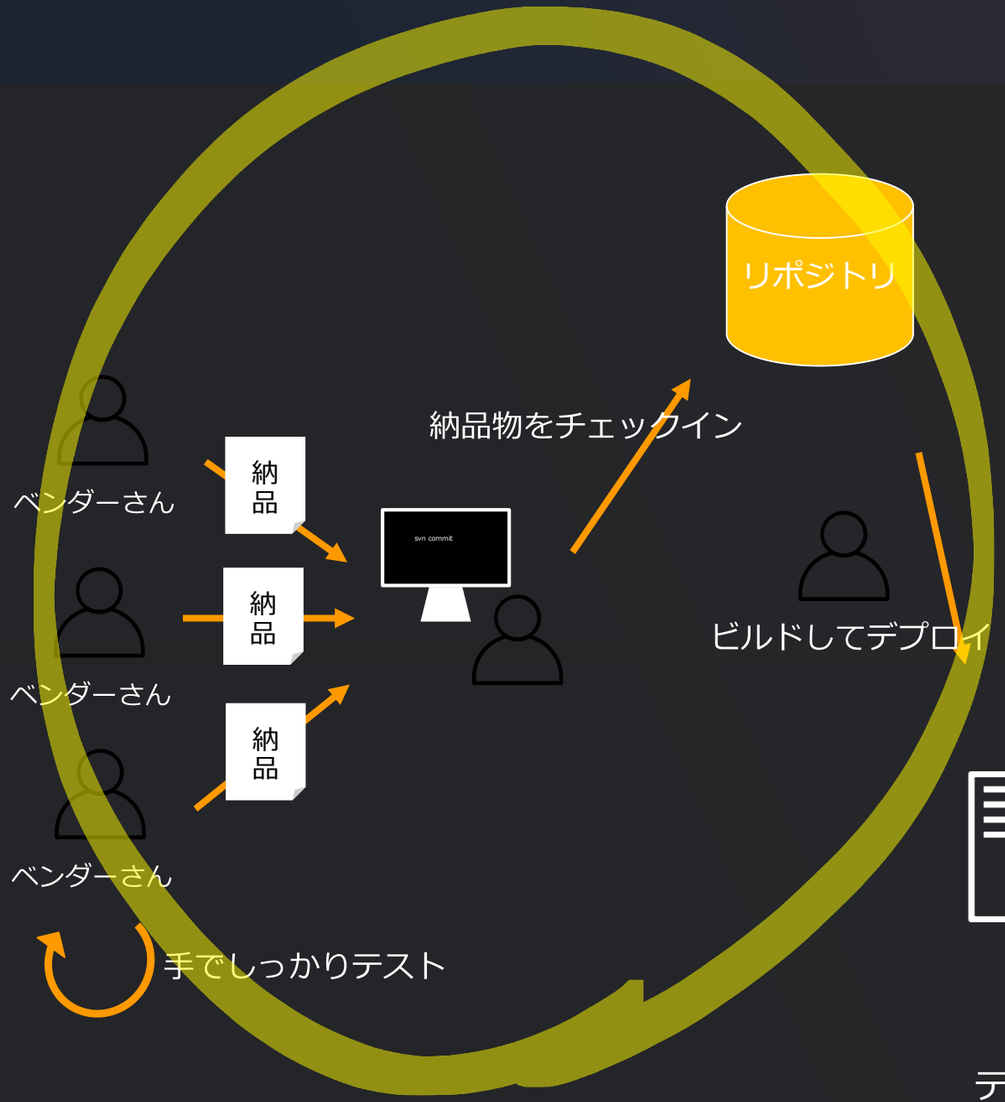
aws DevAx

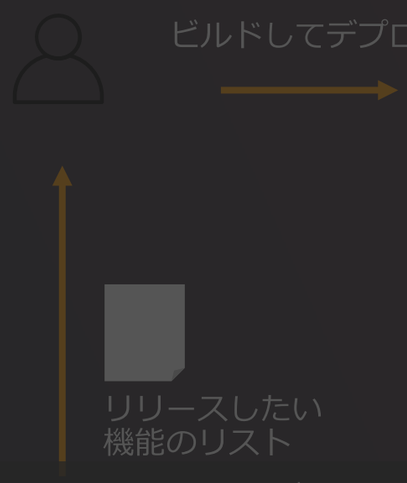
CI準備編



aws DevAx

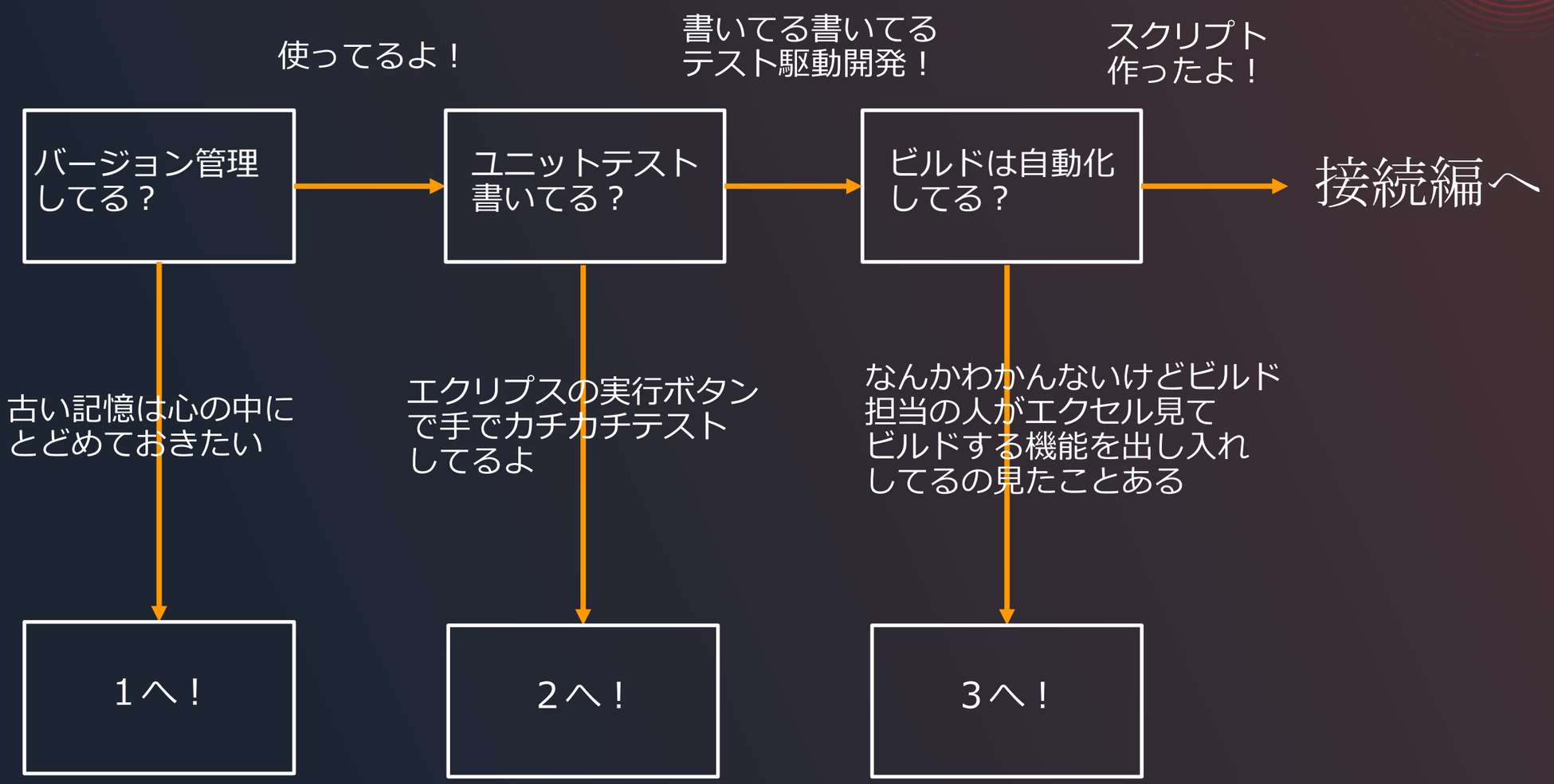
CI準備編





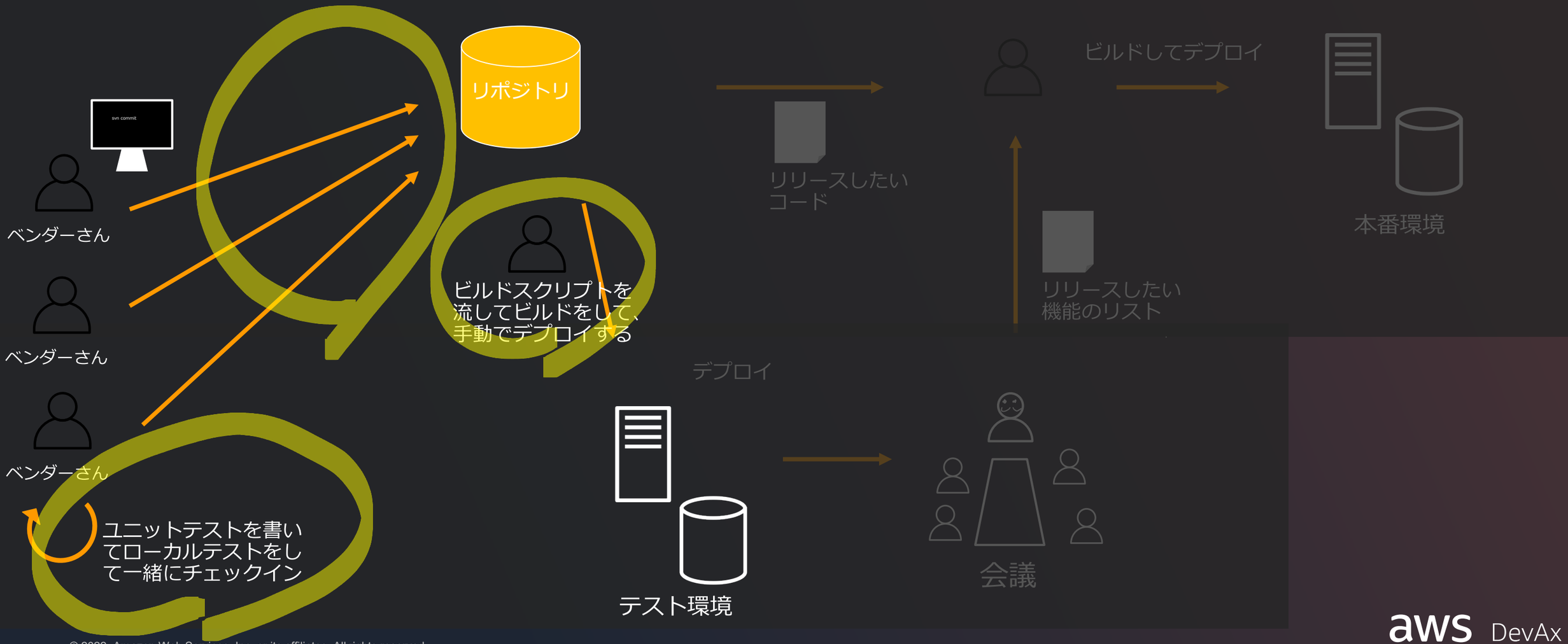
aws DevAx

CI準備編



aws DevAx

CI準備編



aws DevAx

1. コードのバージョン管理ちゃんとできてますか？

いろいろなコード管理ツールがある

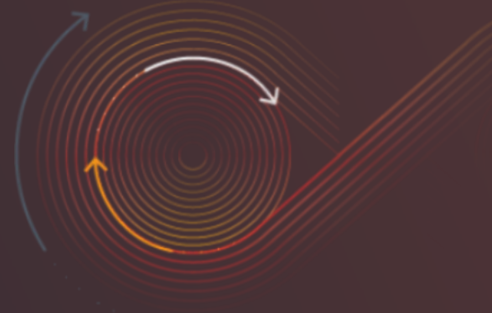
- gitベースのツール
- subversion
- mercurial
- などなど

1. コードのバージョン管理ちゃんとできてますか？

なにかまずいことが起きたときの再現や証跡のため

- 格納されたあらゆるファイルに対してバージョンを保持してアクセスできるようにする
- チームが離れた場所や別の時間帯にいても協力しあえる環境をつくる

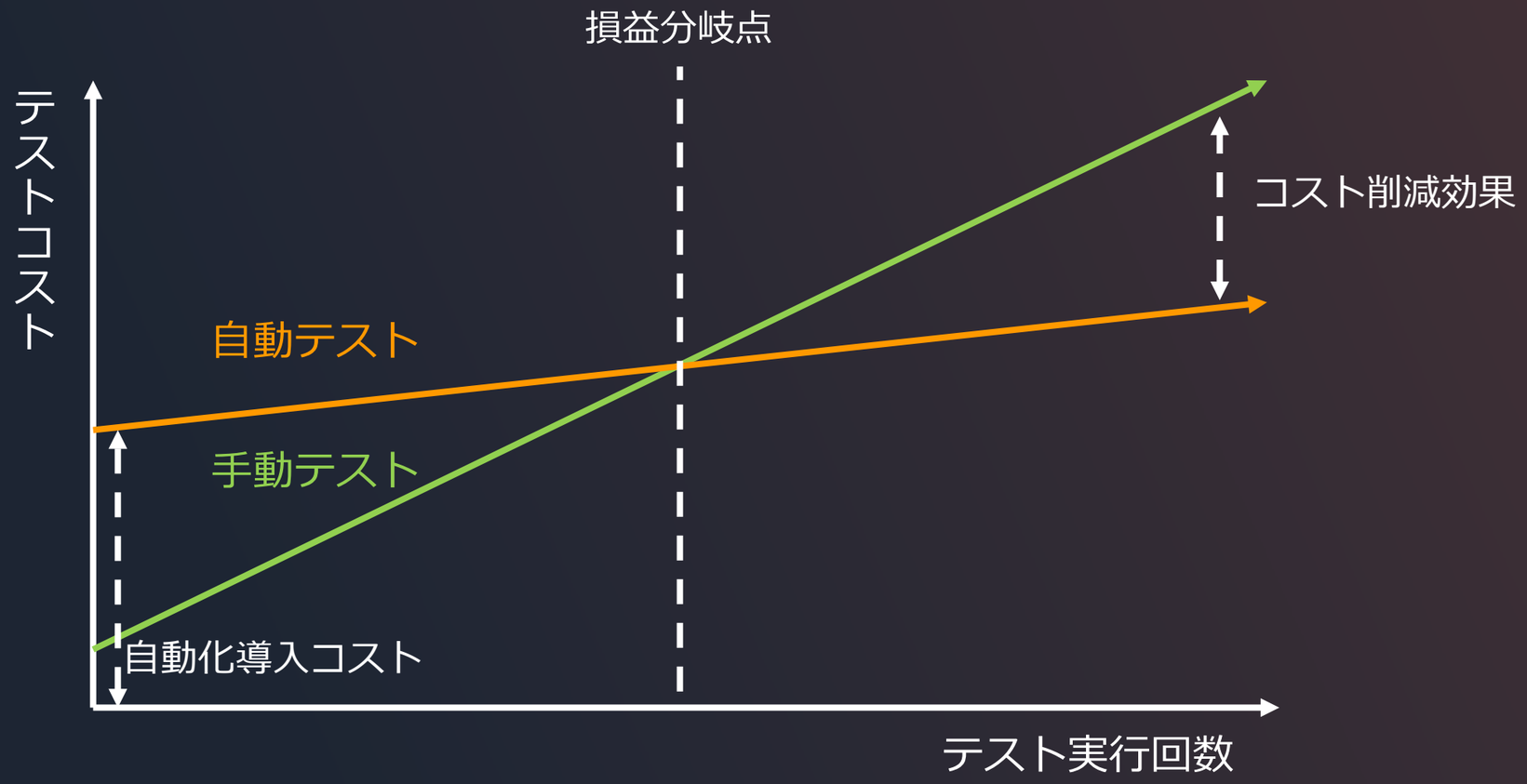
2. ユニットテストは書けていますか？



- CIでの自動テストの重要性
  - コミュニケーション手段としてのCI
  - チェックインに対する心理的安全性
- 機能追加やリファクタリングの高速化
  - アジャイルな開発現場では一度コードを書きあげたら終わりではない
  - スプリントごとに新たな機能を追加
  - メンテナンス性強化のためのリファクタリング

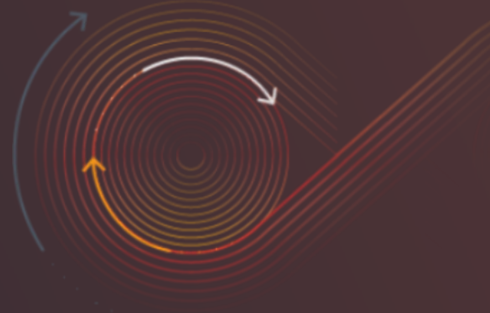
2. ユニットテストは書けていますか？

- テスト自動化は投資



aws DevAx

3. ビルドは自動化できていますか？



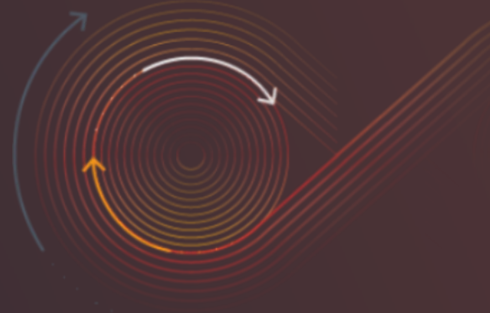
ビルドはコマンドラインから叩けるものにする

- CIツール（後述）から実施できるようにするため
- ビルドスクリプトのバージョン管理をするため
- 運用担当者との共同作業を容易にするため

テストの自動化が同時にできるようにする

- ビルドのコマンドの中でテストコードを実行する仕組みを使う
- ビルドスクリプトとテストスクリプトをまとめて実行するスクリプトを書く

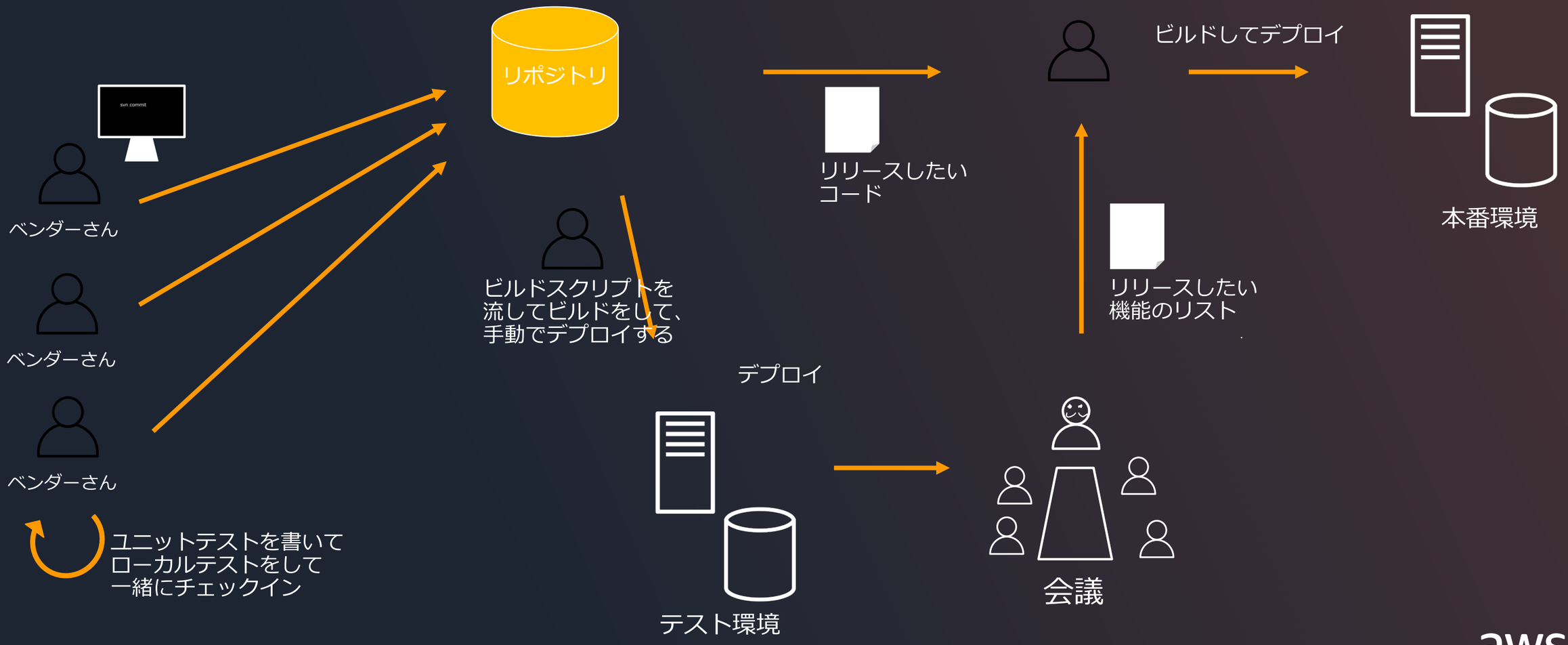
参考：CIと関連するXPのプラクティス



- 10分ビルド
- テストファーストプログラミング

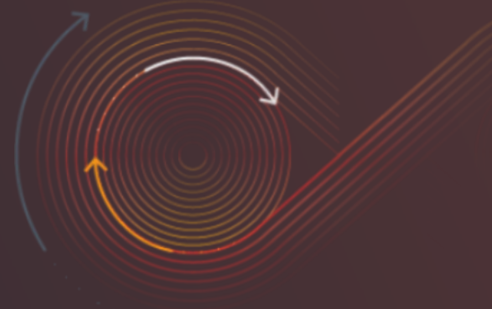
こういったつながりのあるプラクティスが組み合わさって
現在のCIの概念が複合的に完成された

CI準備編



aws DevAx

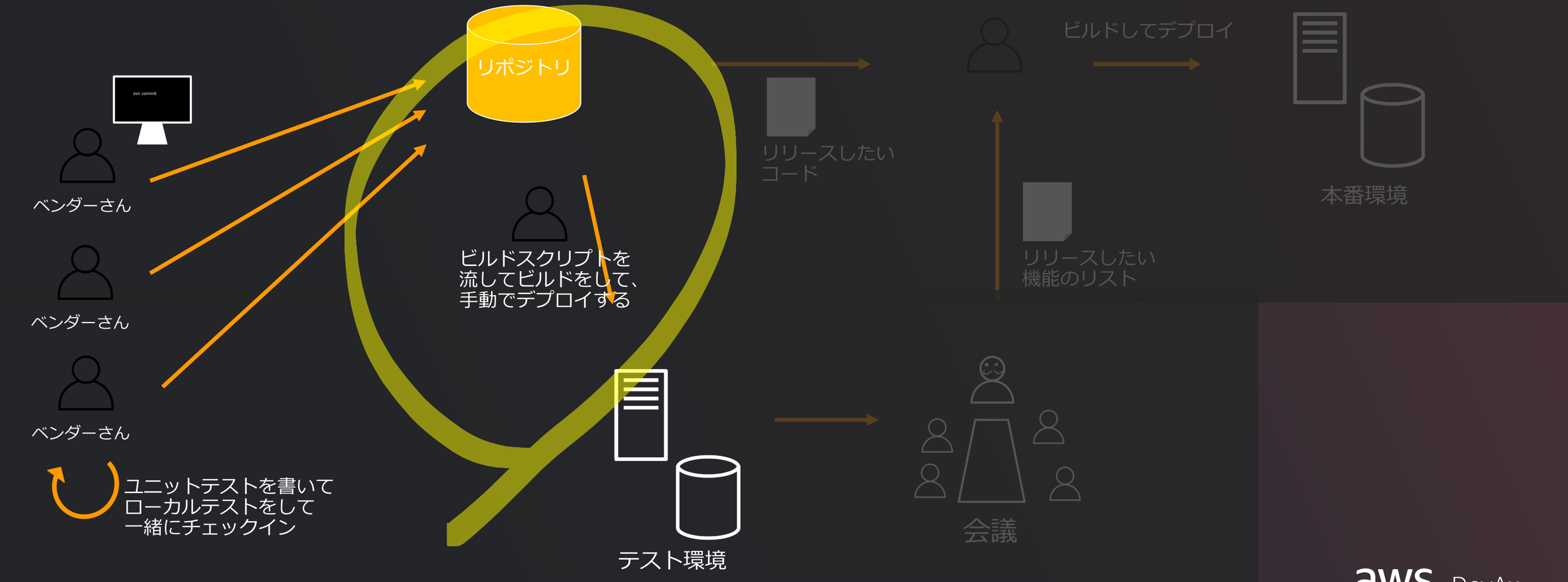
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

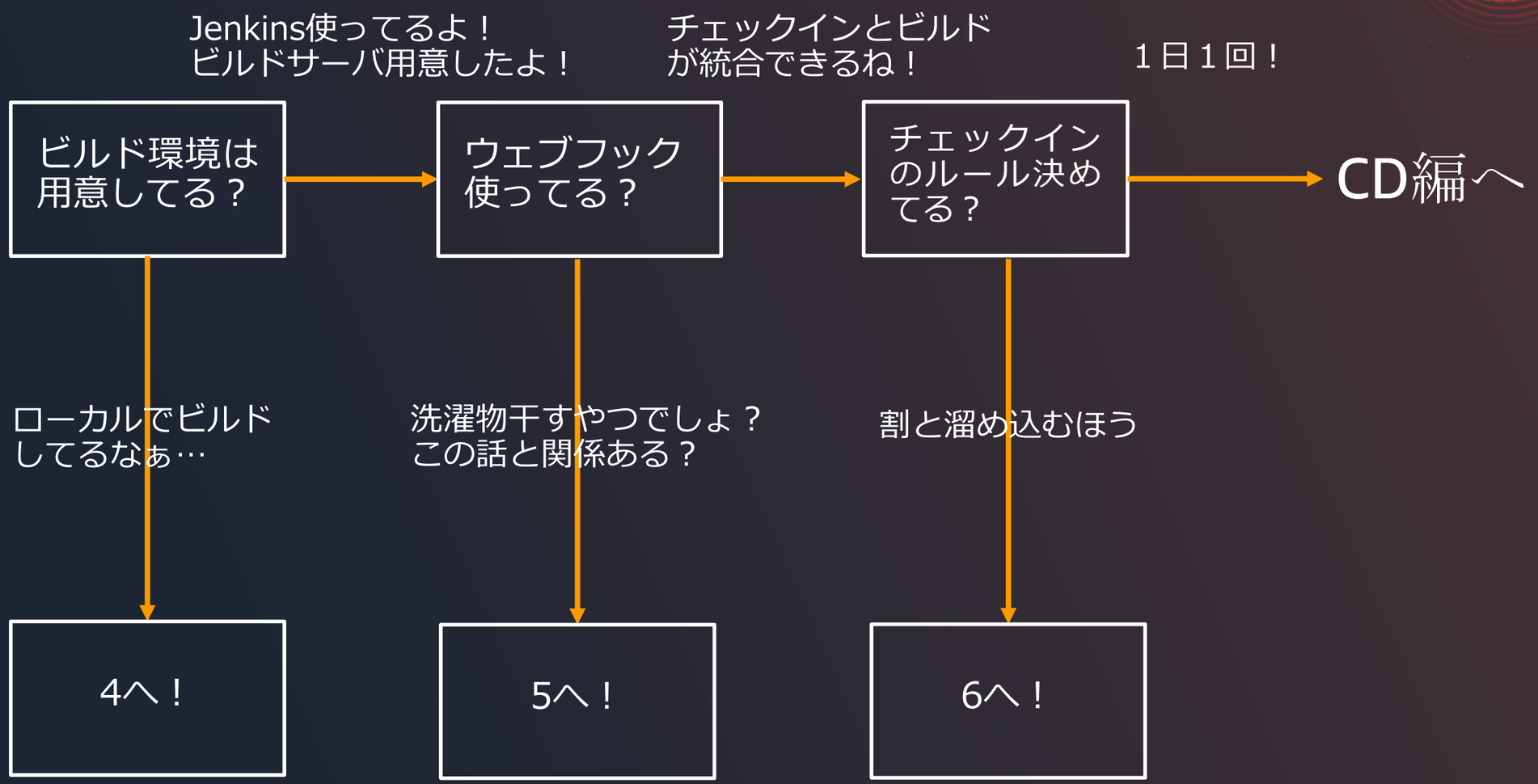
aws DevAx

CI接続編



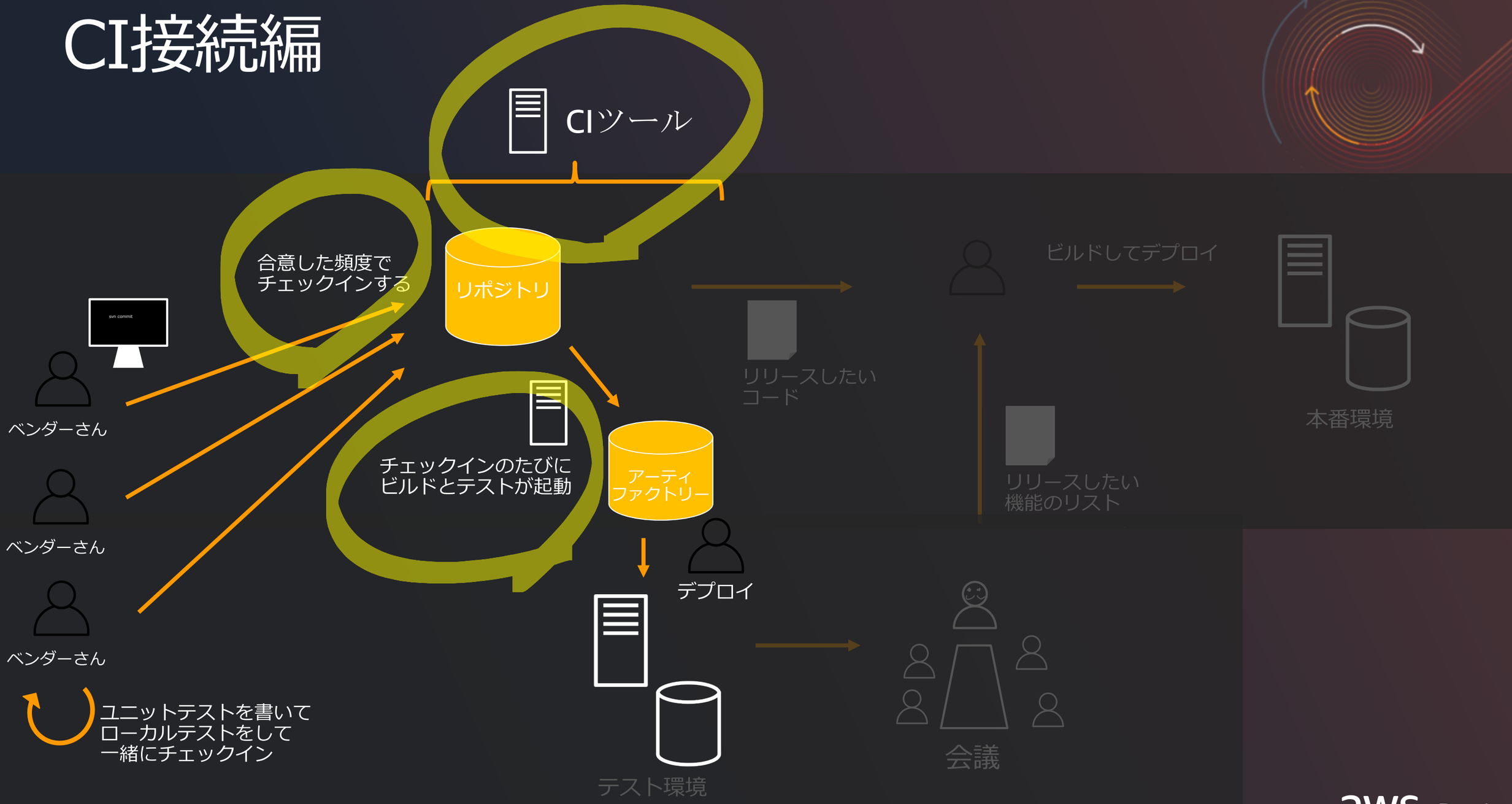
aws DevAx

CI接続編



aws DevAx

CI接続編



aws DevAx

4. ビルドとテストをできる環境はありますか？

- ビルドとテストが行えるような環境（=ビルドサーバ）
  - 前の工程で作成したビルド・テストのスクリプトを実行する環境
- ビルドした成果物を溜め込む場所（=成果物リポジトリ）
  - バイナリであればアーティファクトリ
  - コンテナであればレジストリ

4. ビルドとテストをできる環境はありますか？

- ここまで準備ができればCIツールを導入しても大丈夫！

- CIツールがやってくれること：
  - ビルドやテストを自動で行うための段取りの簡素化
  - 実際の自動ビルド/テストの実施
  - 成果物の保持
  - チェックインを行うたびに起こるビルドやテストのモニタリングとロギング
  - バージョン管理ツールとの連携

- CIツールがやってくれないこと：
  - ユニットテストを書く
  - ビルドスクリプトを書く
  - コードをチェックインする

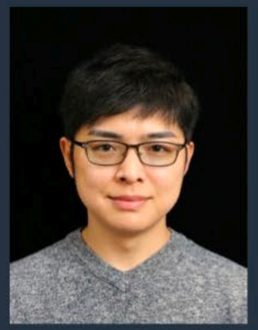
開発者のための開発者による Web セミナーシリーズ

AWS DevAx::connect 3rd 「rethink CI/CD」(前編)

rethink CI/CD

4/7	4/14	4/14	4/21	4/28
CI/CD再入門	CI/CDツール	CI/CD可視化	トランクベース	スペシャル




毎週木曜 16:00-18:00

令和も早や 4 年。私たちは「CI/CD」をできているのか

aws DevAx

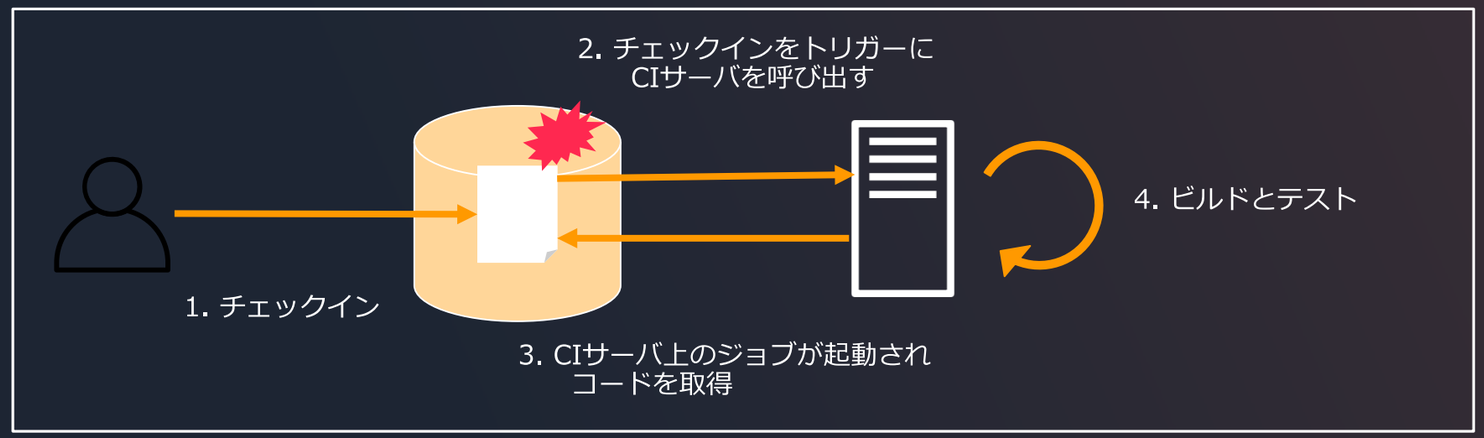
aws DevAx

5. Webhookは使っていますか？

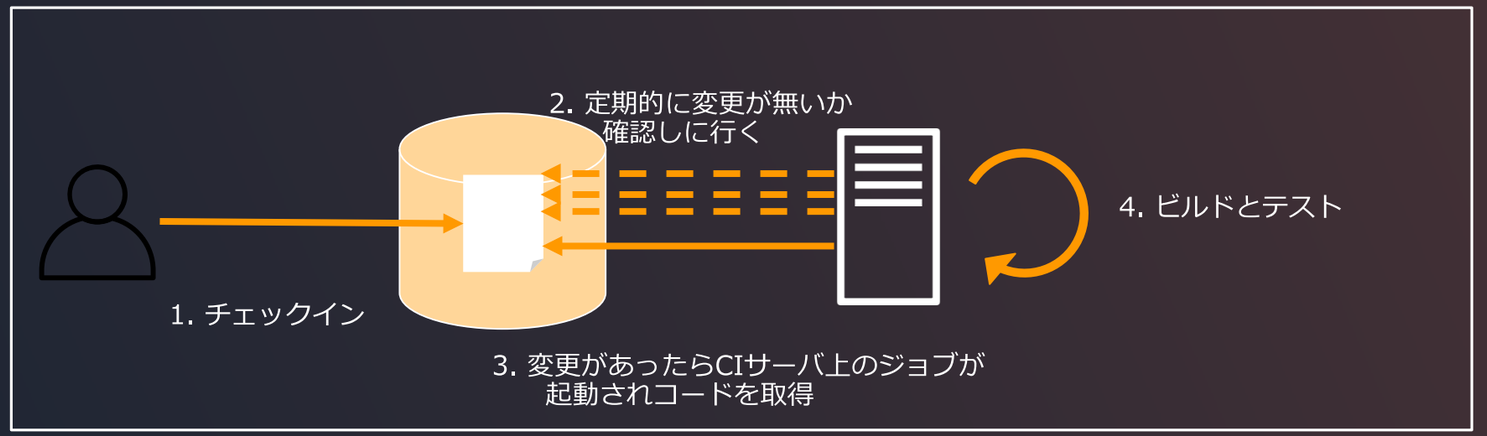
- コードのチェックインなどのアクションがトリガーになる
- URLを指定してPOSTをリクエストすることができる
  →チェックインをトリガーにビルドを起動するときに使う
- CIツールからPULLで変更の確認をすることもできる

5. Webhookは使っていますか？

チェックインをトリガーに後続プロセスを起動

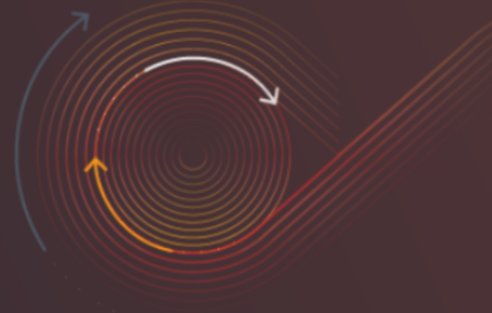


CIサーバからの定期チェックで後続プロセスを起動



aws DevAx

参考：ビルドのプラクティス

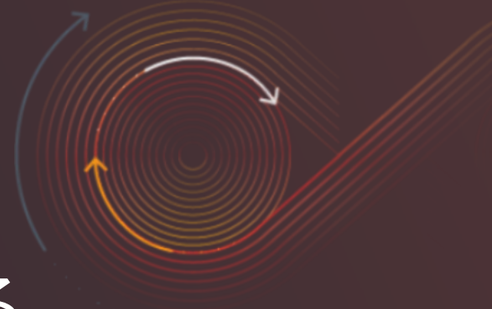


- ビルドが壊れているときにはチェックインするな
- ビルドが壊れているのに家に帰ってはならない
- 常に前のリビジョンに戻す準備をしておくこと
- リバートする前にタイムボックスを切って修正する
- 自分が変更してビルドが壊れたら全てに対して責任を取れ

- 継続的デリバリー 第三章:継続的インテグレーションより

ビルドが壊れていることを異常事態であると同時に、
割と起こるということも認識しておく

6. チェックインのルールを決めていますか？



- チェックインの頻度はいろいろな要因に左右される
  - チームの開発規模
  - 個人の開発規模
  - コンポーネント分割の規模・結合度合い
  - コードレビューの方針
  - 他のアジャイルプラクティスとの併用
    - テスト駆動開発
    - ペアプログラミング

- 絶対にメインブランチに1日1回以上チェックインしないとCI/CDのコンセプトが全く使えない！というわけでもない

参考：コード管理戦略

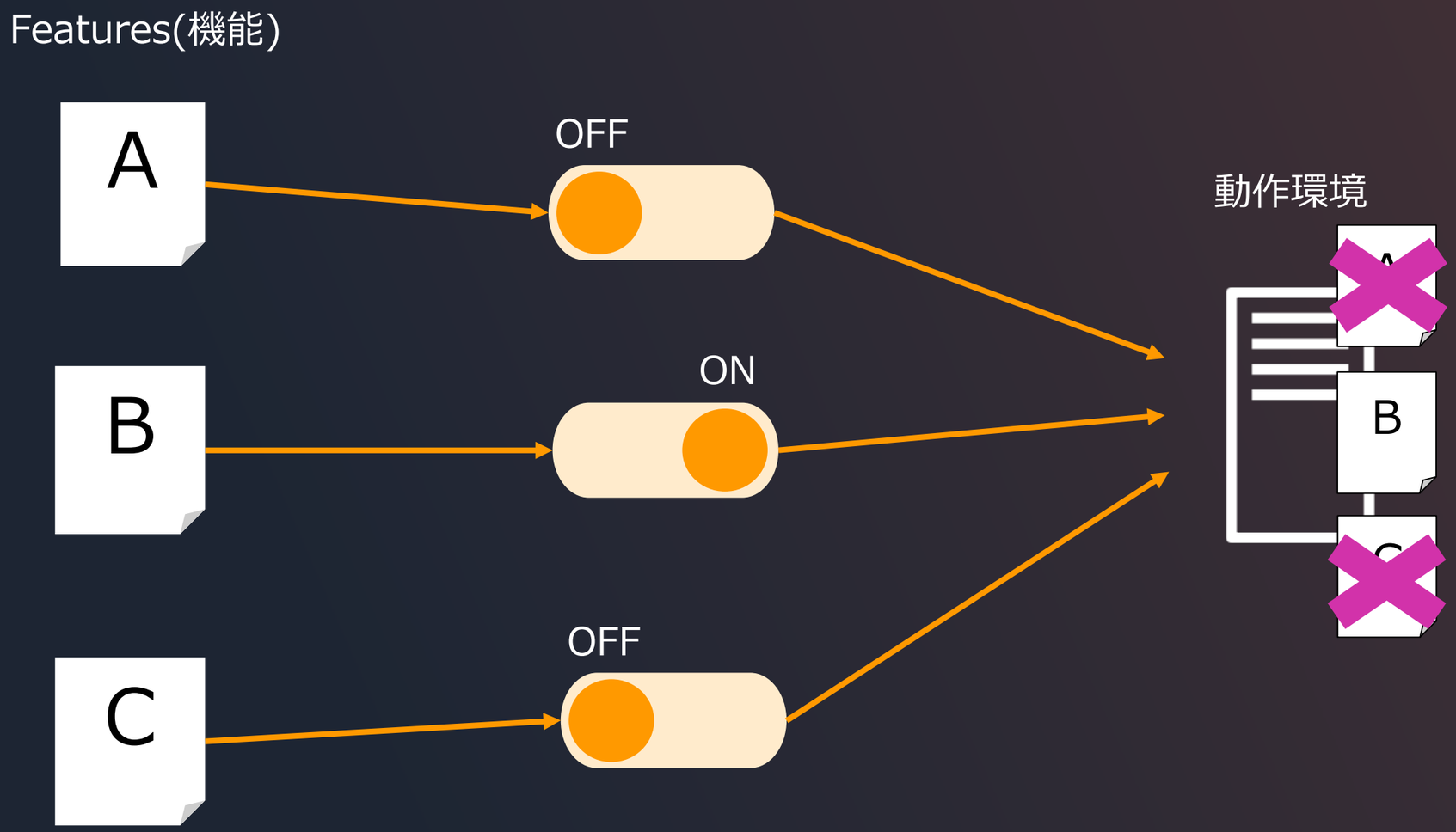
- メインラインインテグレーション
- フィーチャーブランチ
- Git Flow
- GitHub Flow
- Pull Request

aws DevAx

6. チェックインのルールを決めていますか？

- 最も高頻度でインテグレーションをするには？
  - メインライン（トランク・メインブランチ）に直接インテグレーションをして、一日以上の作業が統合されずにローカルレポジトリにあってはならない
  - チェックインのたびにビルドシステムが起動され、ビルドとテストをする
  - Feature Toggleやkey stone interfaceでの隔離などの実装隠蔽をおこなう

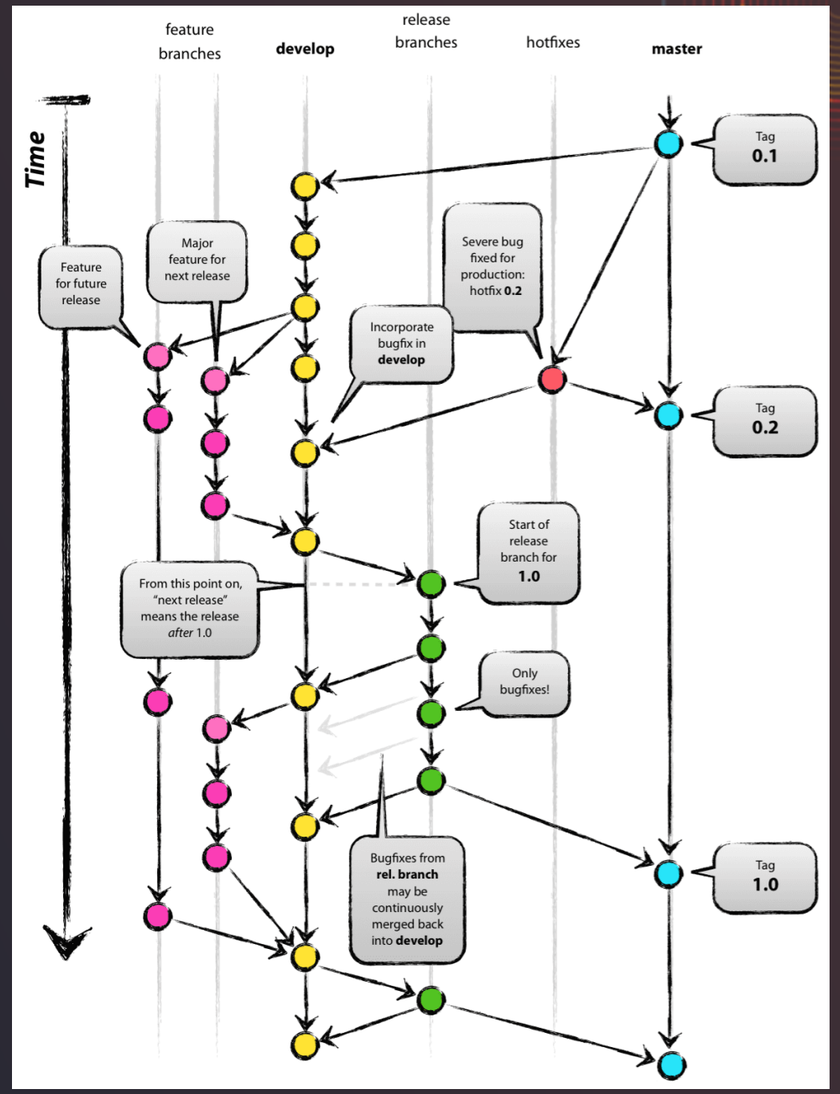
参考：Feature Toggle



aws DevAx

6. チェックインのルールを決めていますか？

- 大きなビルド単位では大掛かりなコード管理戦略を取らざるをないこともよくある
- 小さなビルド単位ではCIに直結できるようなシンプルな戦略を取りやすい



割と複雑なバージョン管理戦略として有名なgit-flow

参考：https://nvie.com/posts/a-successful-git-branching-model/

aws DevAx

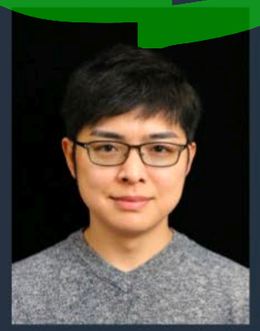
開発者のための開発者による Web セミナーシリーズ

AWS DevAx::connect 3rd 「rethink CI/CD」(前編)

rethink CI/CD

4/7 CI/CD再入門	4/14 CI/CDツール	4/14 CI/CD可視化	4/21 トランクベース	4/28 スペシャル



?

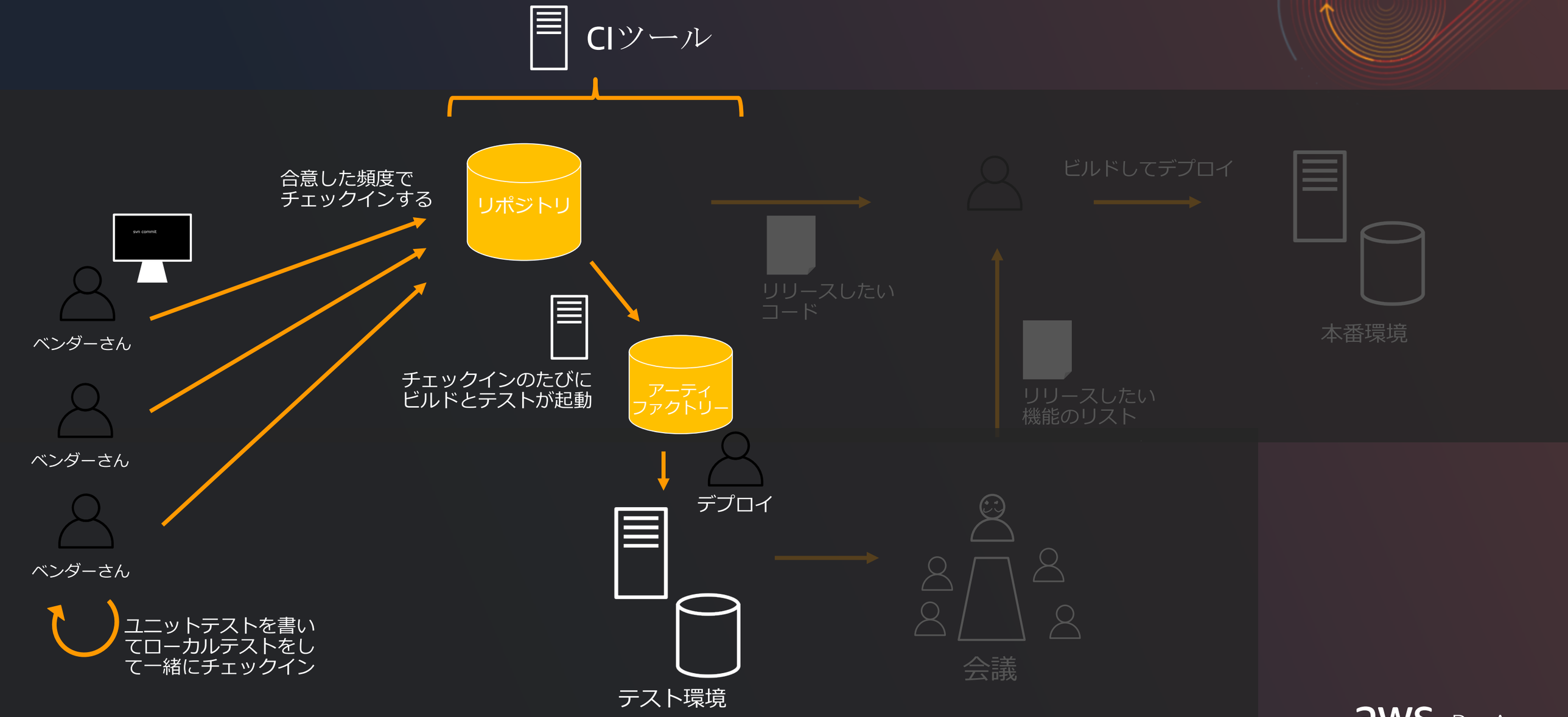
毎週木曜 16:00-18:00

令和も早や 4 年。私たちは「CI/CD」をできているのか

aws DevAx

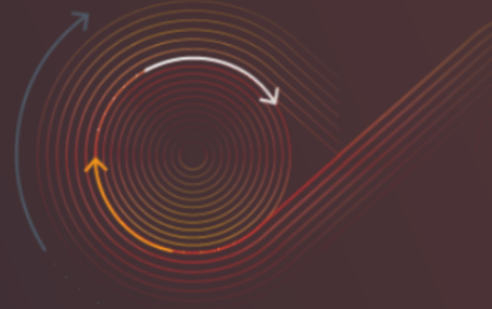
aws DevAx

ここまで：CI編まとめ



aws DevAx

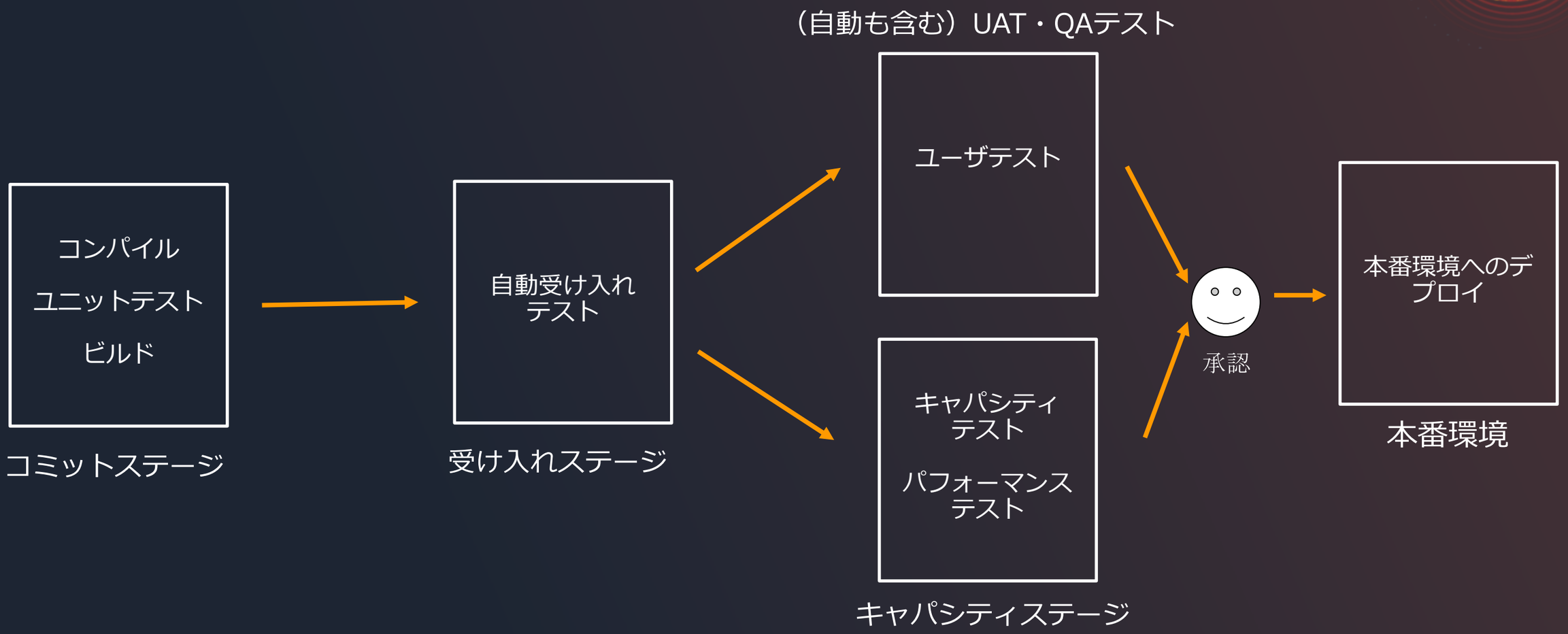
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

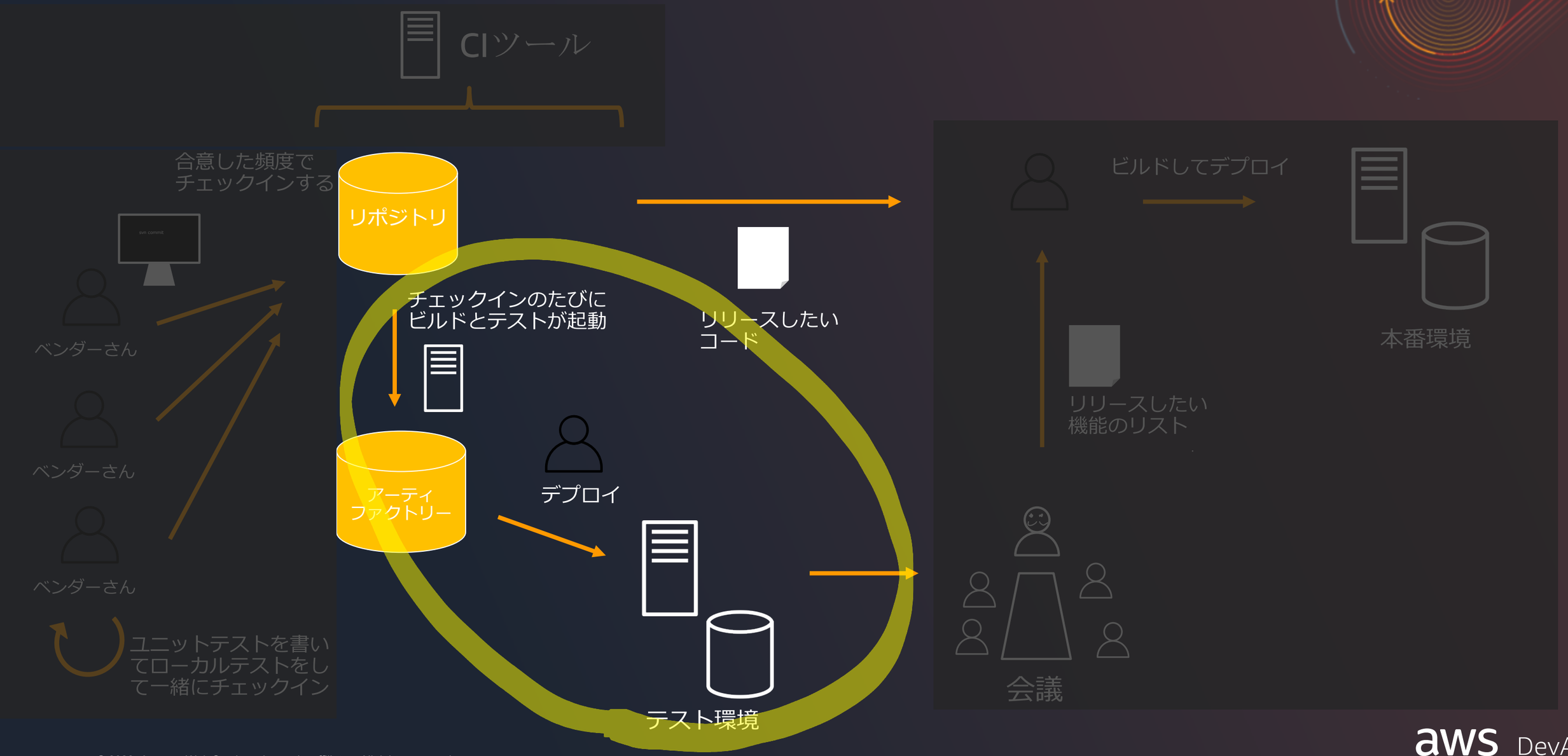
aws DevAx

継続的デリバリー：デプロイメントパイプライン



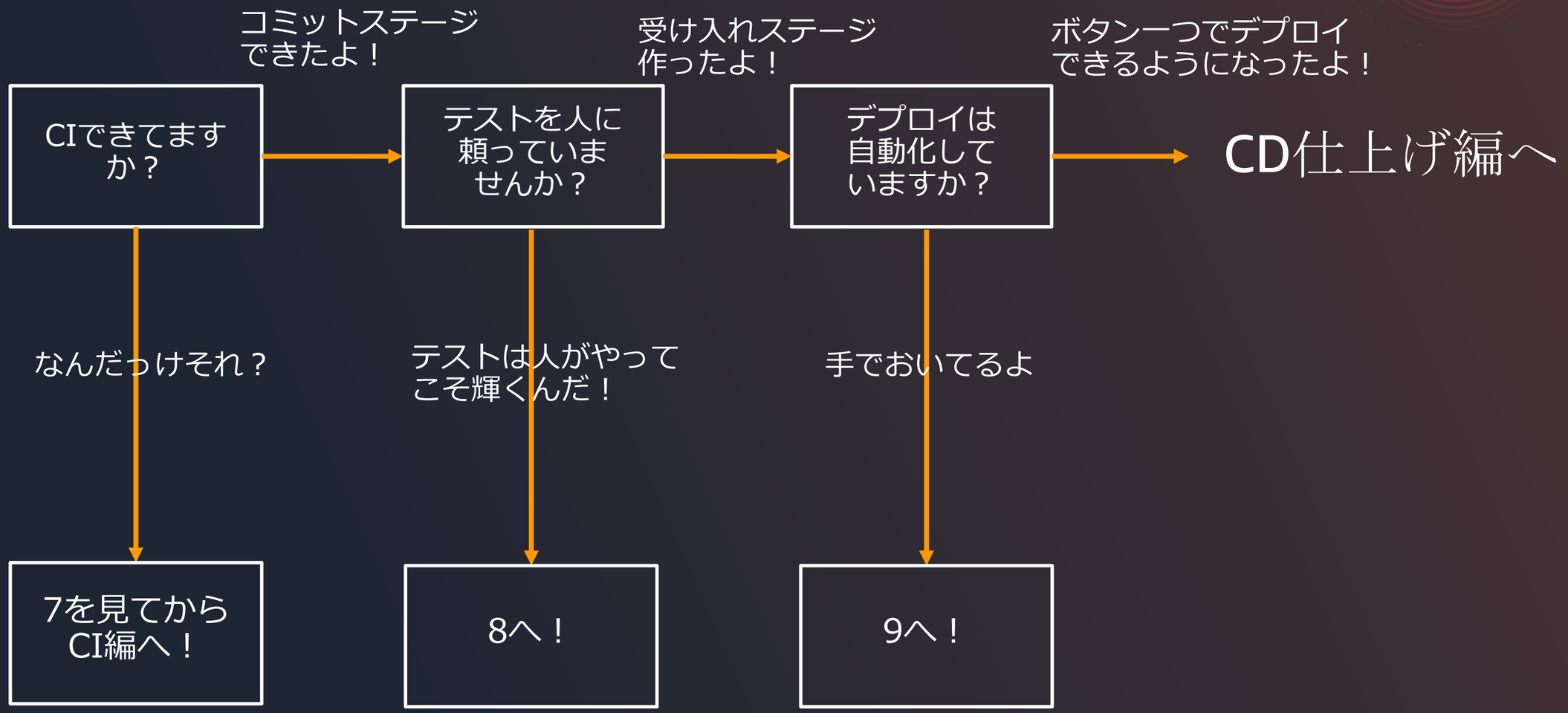
aws DevAx

CD：ステージ構築編



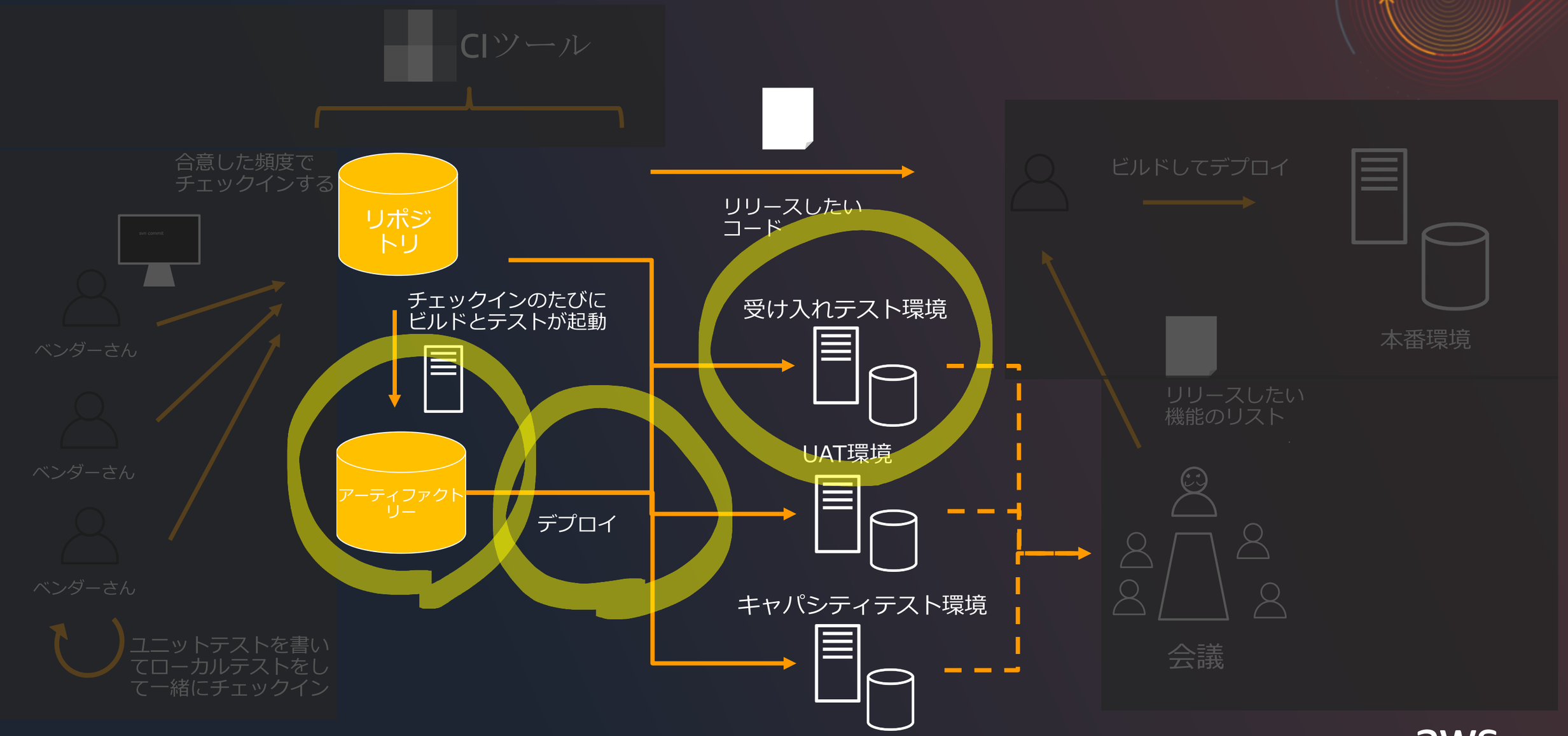
aws DevAx

CD：ステージ構築編



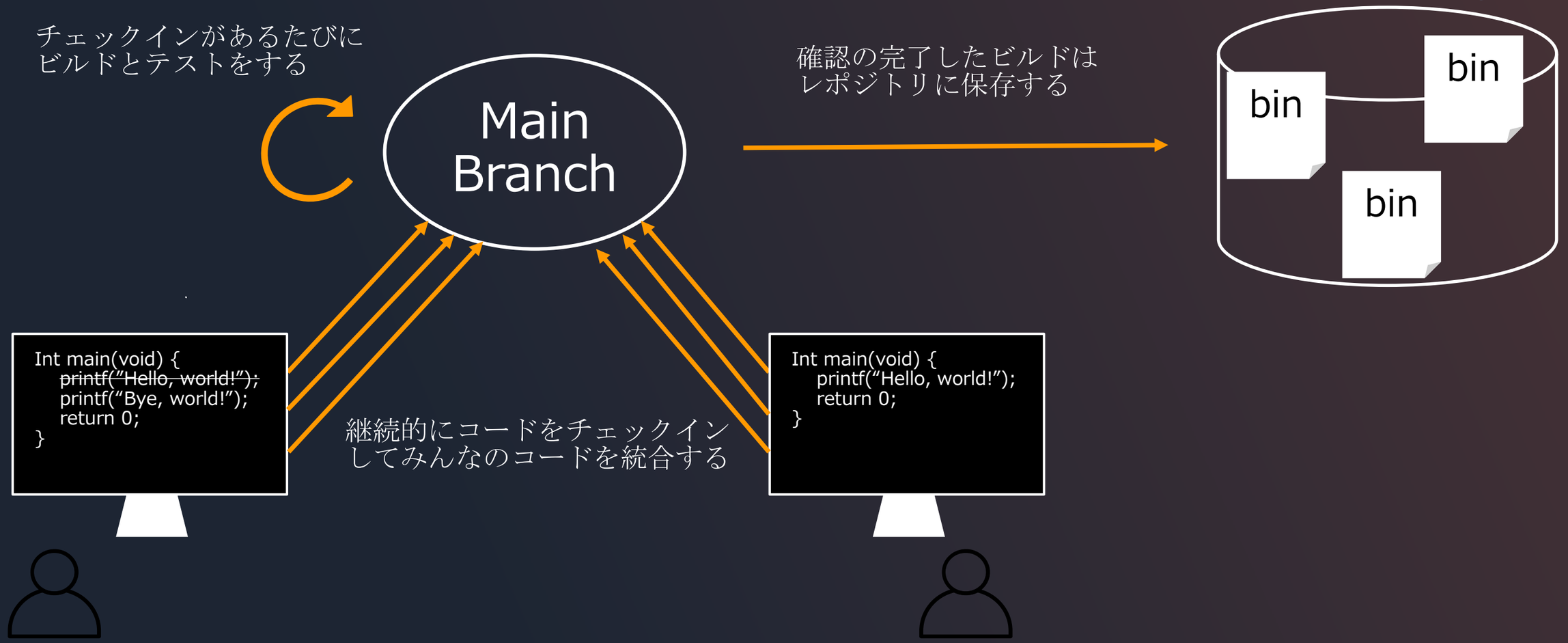
aws DevAx

CD：ステージ構築編



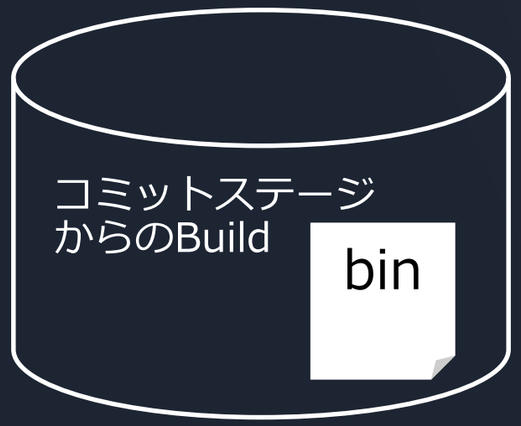
aws DevAx

7. CIできてますか？
- コミットステージ



aws DevAx

8. テストを人に頼ってませんか？
- 受け入れステージ



自動受け入れ
テスト

OK!

大抵の場合
人間系のテスト

通らなければ先のステージ
には行けない

直してチェックインし直し

NG…

aws DevAx

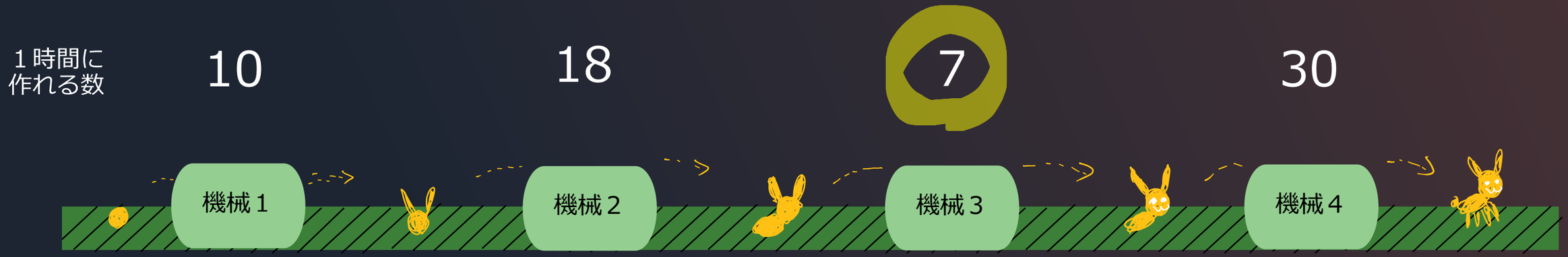
参考：制約理論（Theory of Constraints）

- 工場のスループットを最大限に上げるために考えられた理論
- 今ではありとあらゆる”制約”のある系を管理するための理論として有名
- 適用分野：
  生産・サプライチェーン
  ロジスティクス・会計・営業
  プロジェクト・研究開発・ITなど
- 簡単に言うと、ボトルネックになるリソースを最大限使うように周りが調整するという理論

aws DevAx

参考：制約理論

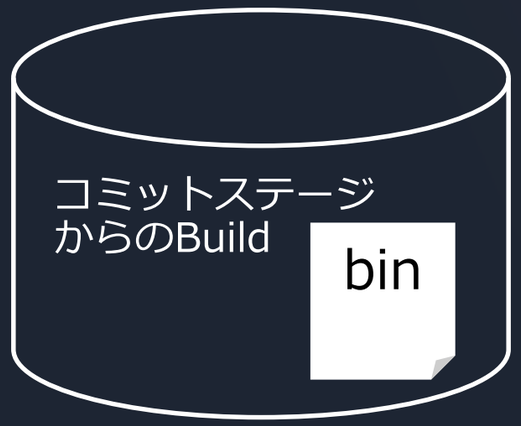
例：ベルトコンベアで作られるうさぎのぬいぐるみ



スループット（１時間で作れるうさぎの数の最大）：7

aws DevAx

8. テストを人に頼ってませんか？
- 受け入れステージ



自動受け入れ
テスト

OK!

大抵の場合
人間系のテスト

通らなければ先のステージ
には行けない

直してチェックインし直し

NG…

aws DevAx

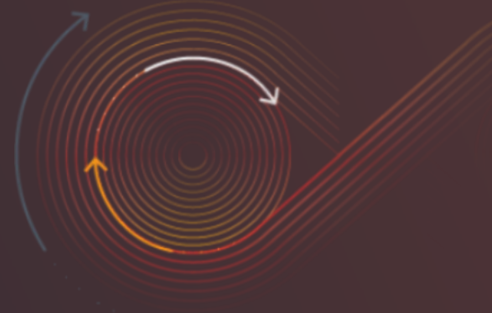
8. テストを人に頼ってませんか？
- 受け入れステージ

- 基本的には本番環境と同等の環境で行う

- 今のテストの状況から逆算する
  - テスターやユーザーが毎回手動でやっているテストが自動化出来ないか
  - リグレッションテストは十分出来ているか

- 受け入れ基準
  - ふるまい駆動開発（BDD）

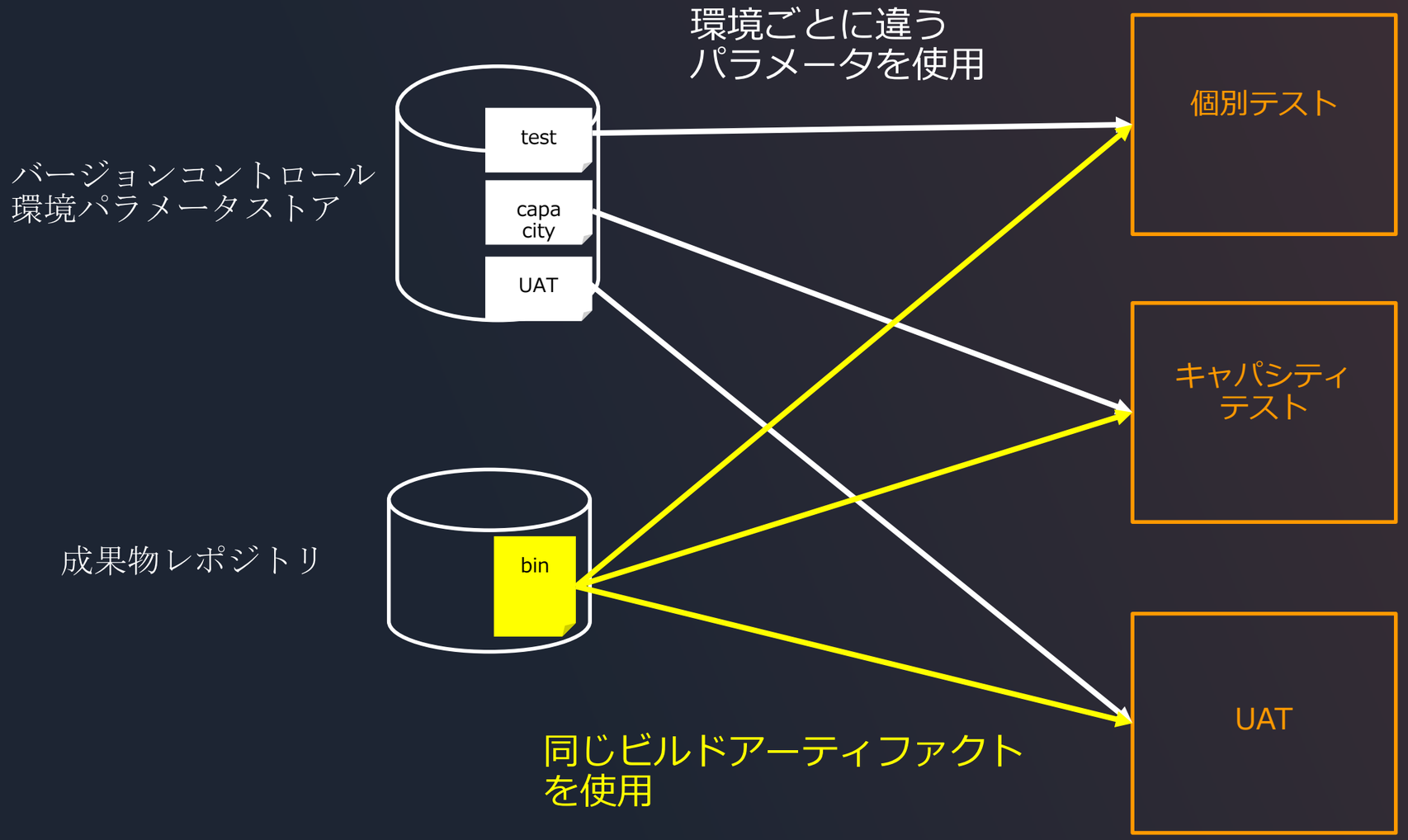
9. デプロイは自動化してますか？



- 具体的な実装については開発で使っている言語や
  フレームワーク、ツールセットによる

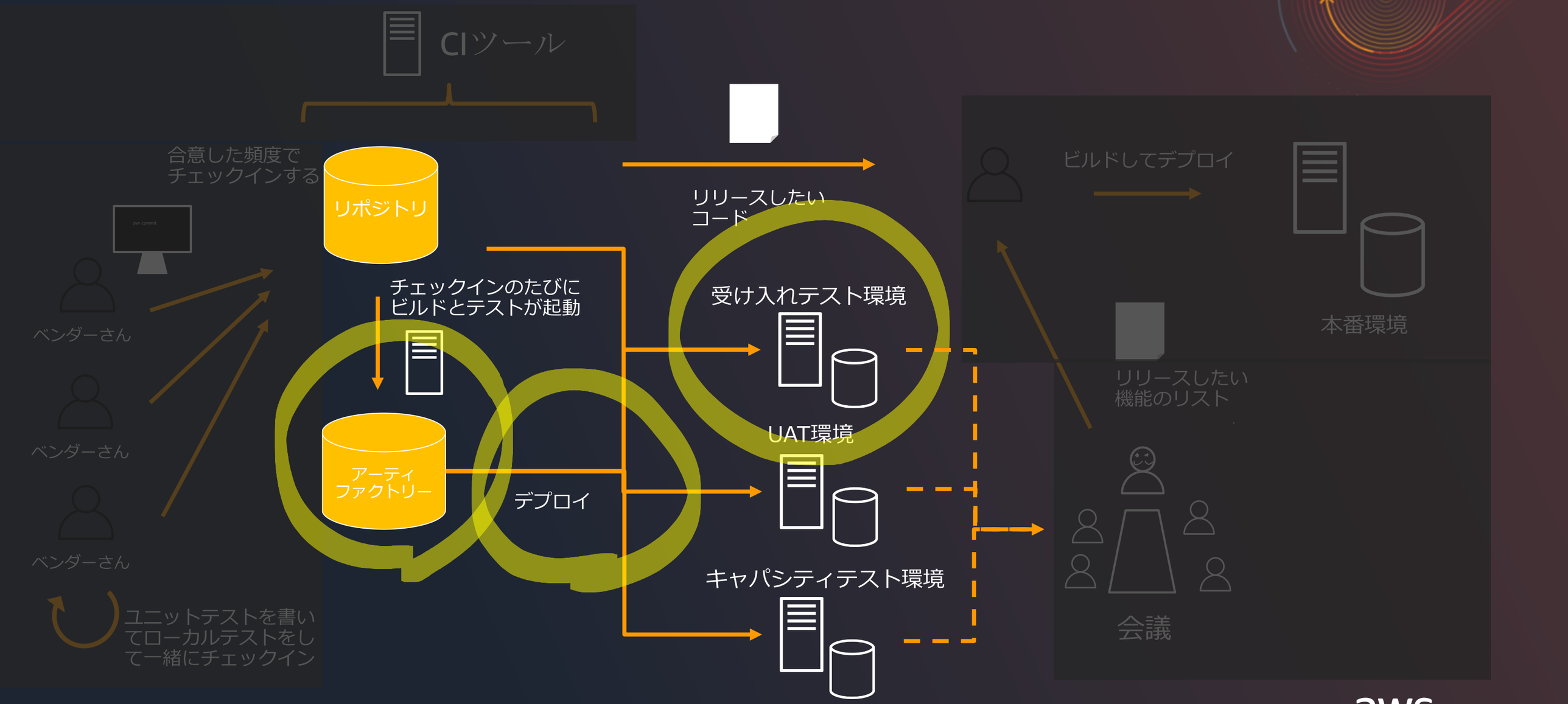
- 自分の使っているフレームワークを簡単にデプロイできる
  ツールが無いか調べてみよう！

9. デプロイは自動化してますか？



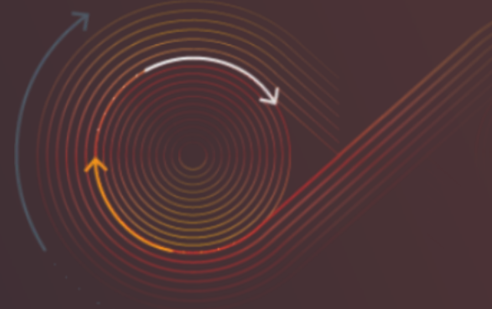
aws DevAx

CD：ステージ構築編



aws DevAx

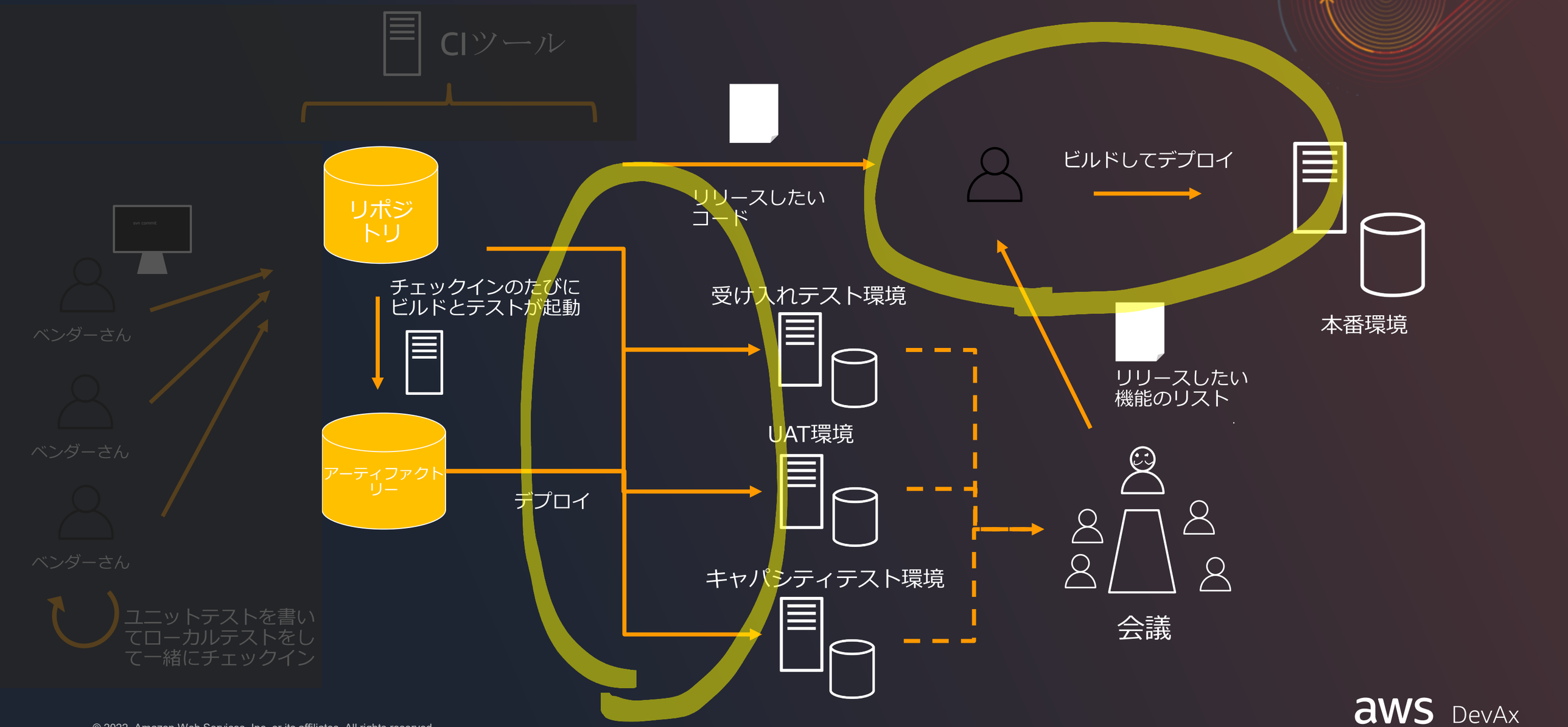
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

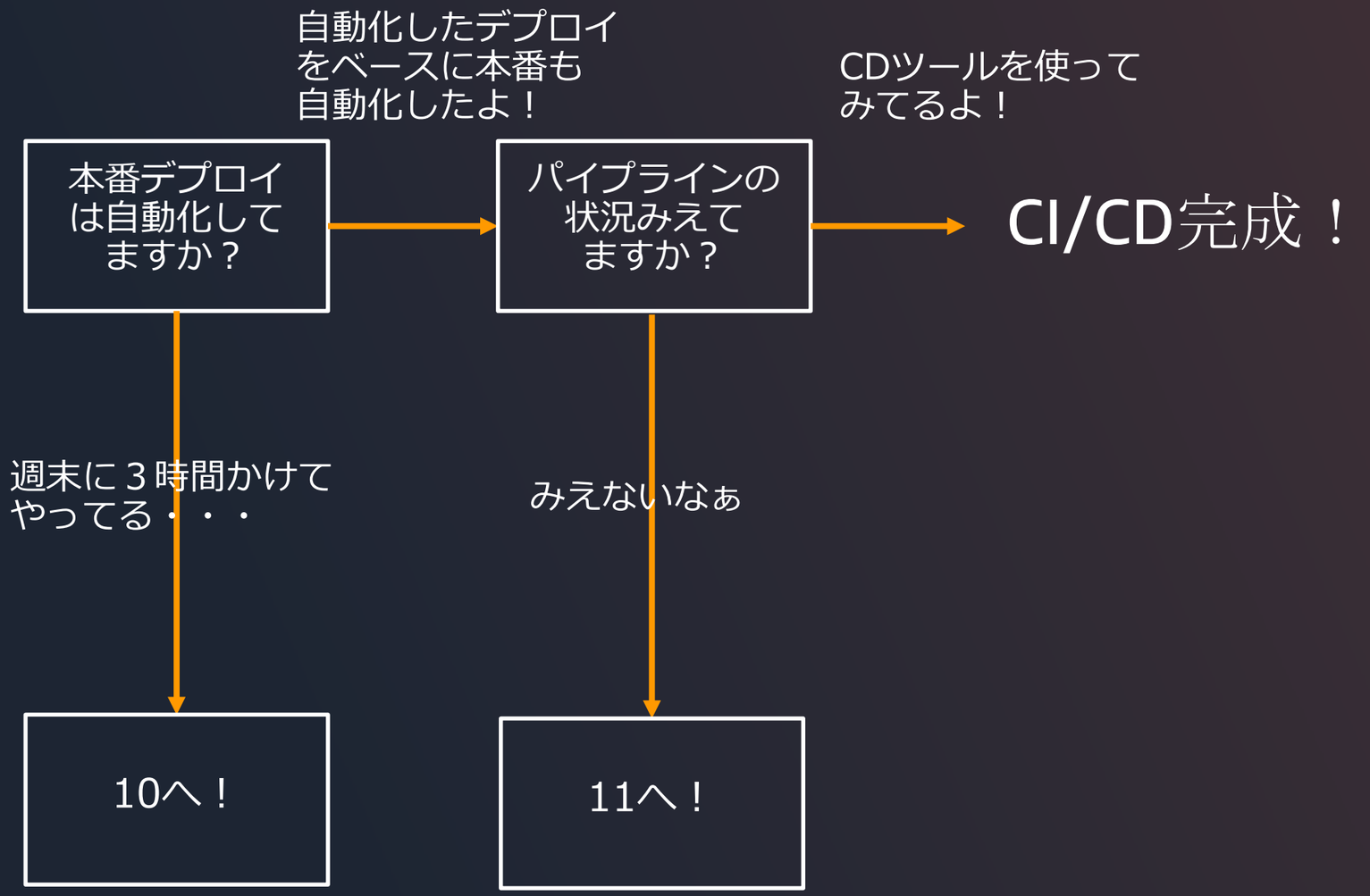
aws DevAx

CD：パイプライン構築編



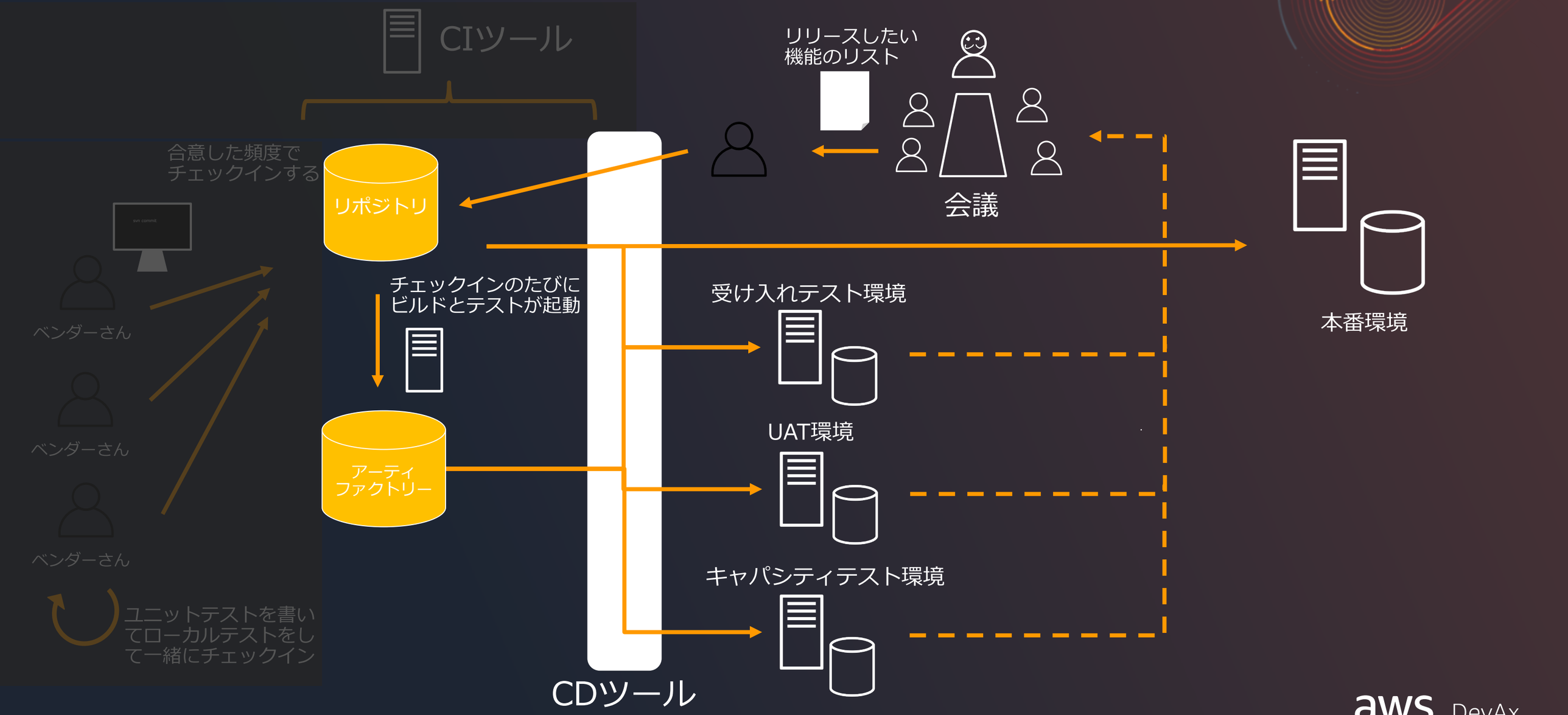
aws DevAx

CD：パイプライン構築編



aws DevAx

CD：パイプライン構築編



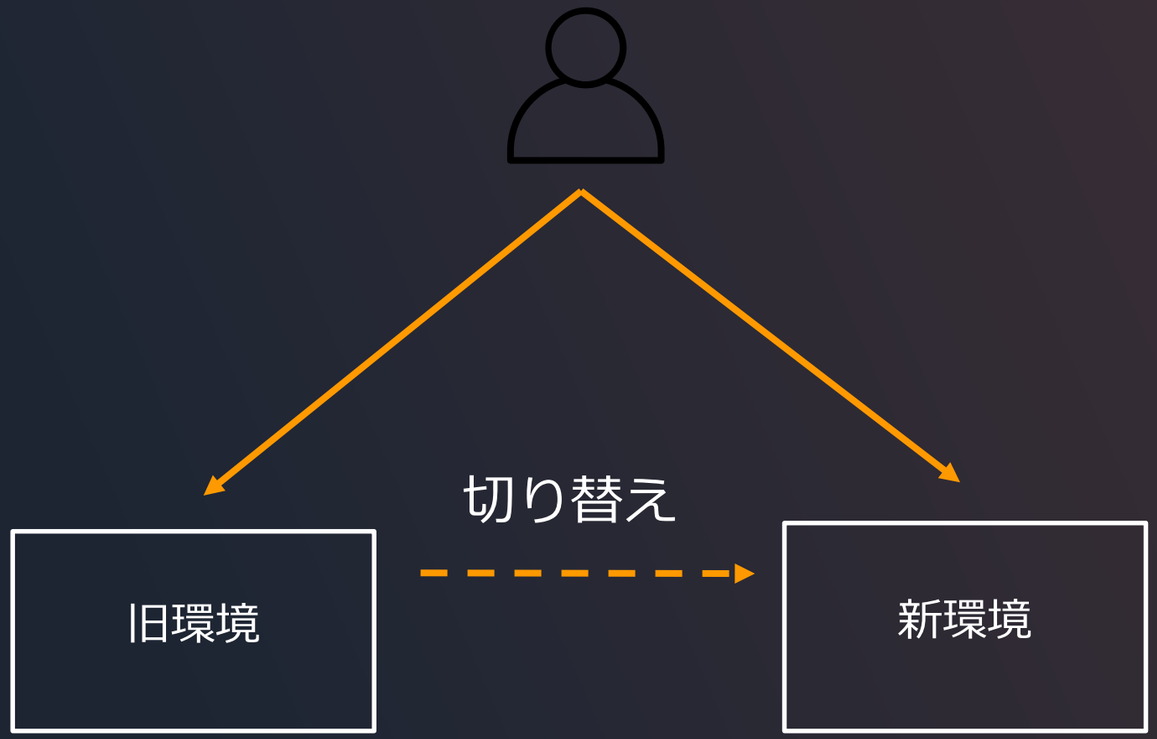
aws DevAx

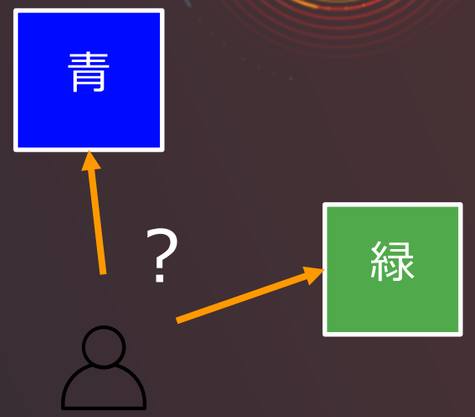
10. 本番環境には安全に素早くデプロイ出来ていますか？

- 本番環境でも前の環境と同じやり方でデプロイをする
- デプロイ担当者がデプロイプロセスの策定に関わる
- ロールバック・ダウンタイムのないリリース
  - ブルーグリーンデプロイ
  - カナリアリリース

参考：いろいろなデプロイの方法

カナリアリリース





ブルー・グリーン
デプロイ

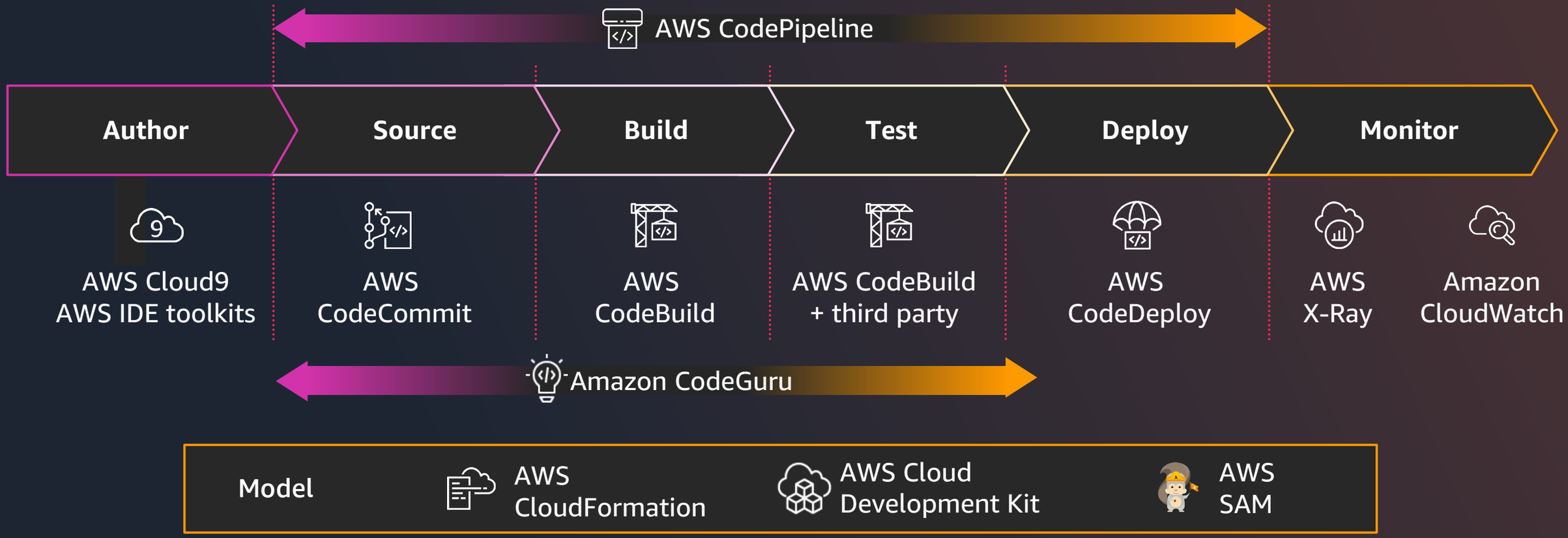
Blue-Green：一度に旧から新へ切り替え

カナリア：少しだけ新環境にトラフィックを流して確認

aws DevAx

11. パイプラインの状況が見えていますか？

参考：AWSサービスでのパイプライン構築



aws DevAx

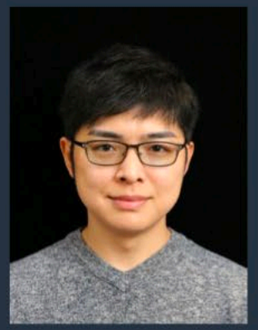
開発者のための開発者による Web セミナーシリーズ

AWS DevAx::connect 3rd 「rethink CI/CD」(前編)

rethink CI/CD

4/7	4/14	4/14	4/21	4/28
CI/CD再入門	CI/CDツール	CI/CD可視化	トランクベース	スペシャル



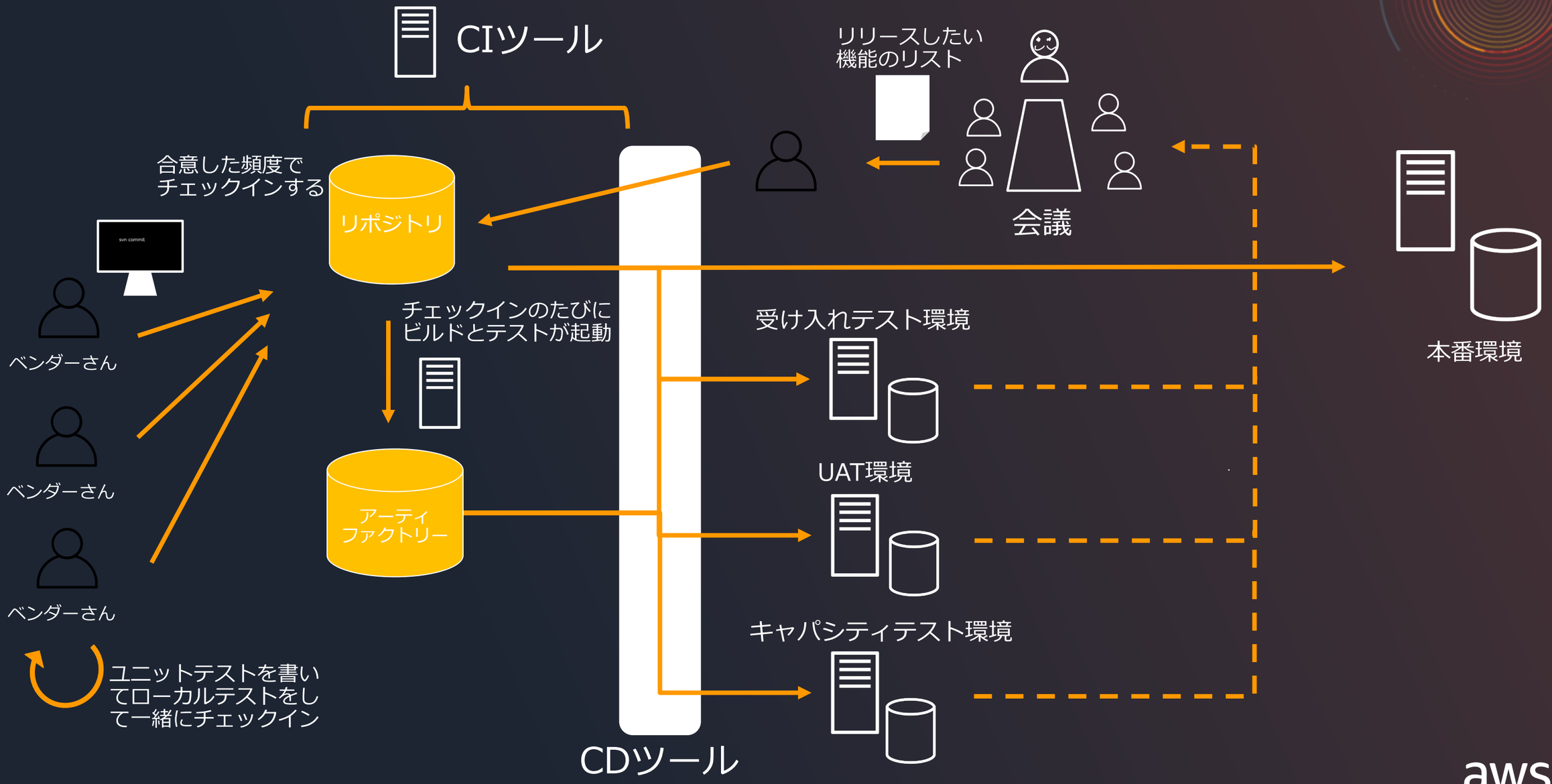

毎週木曜 16:00-18:00

令和も早や 4 年。私たちは「CI/CD」をできているのか

aws DevAx

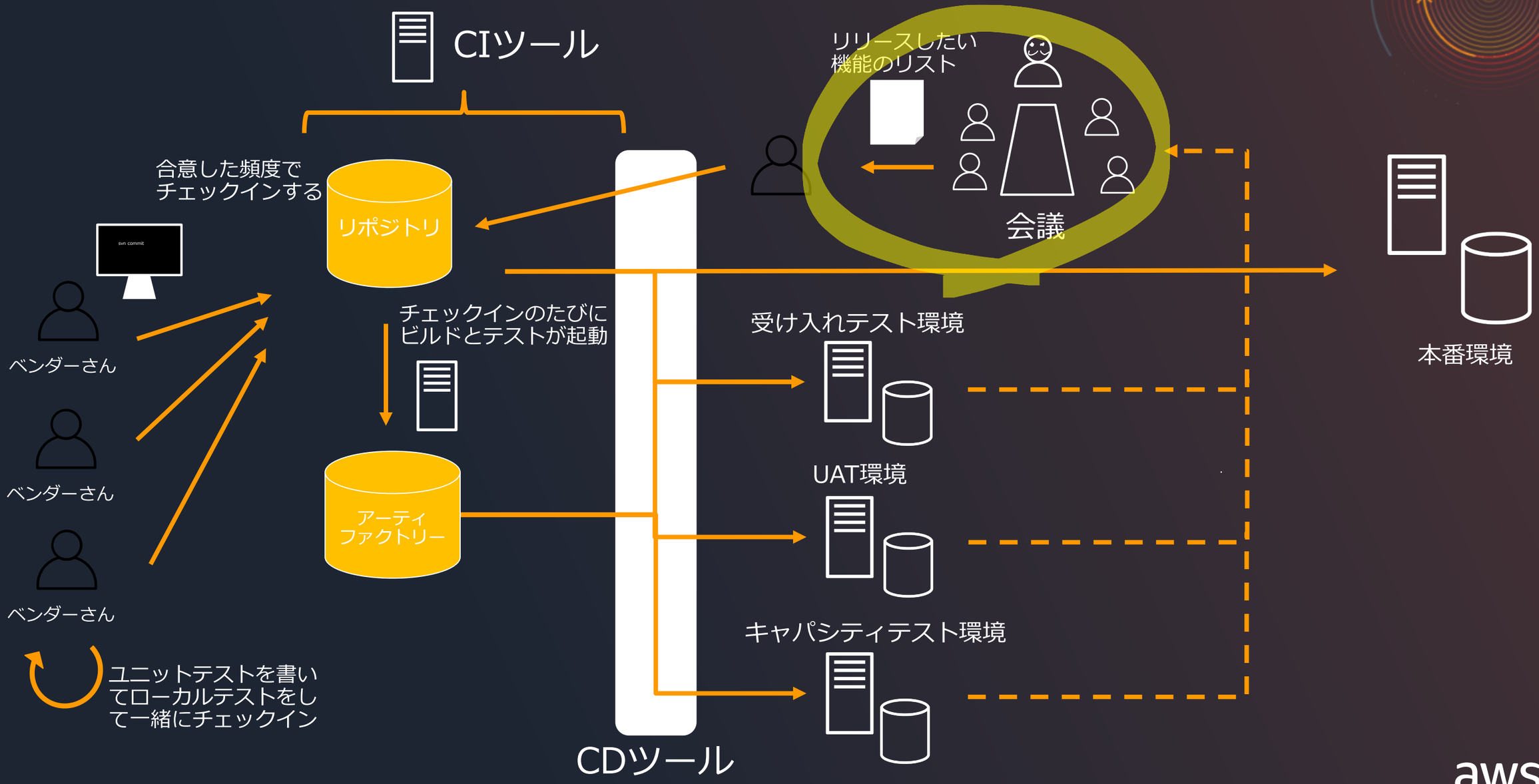
aws DevAx

CD：パイプライン構築編



aws DevAx

CD：パイプライン構築編



aws DevAx

参考：継続的デプロイ

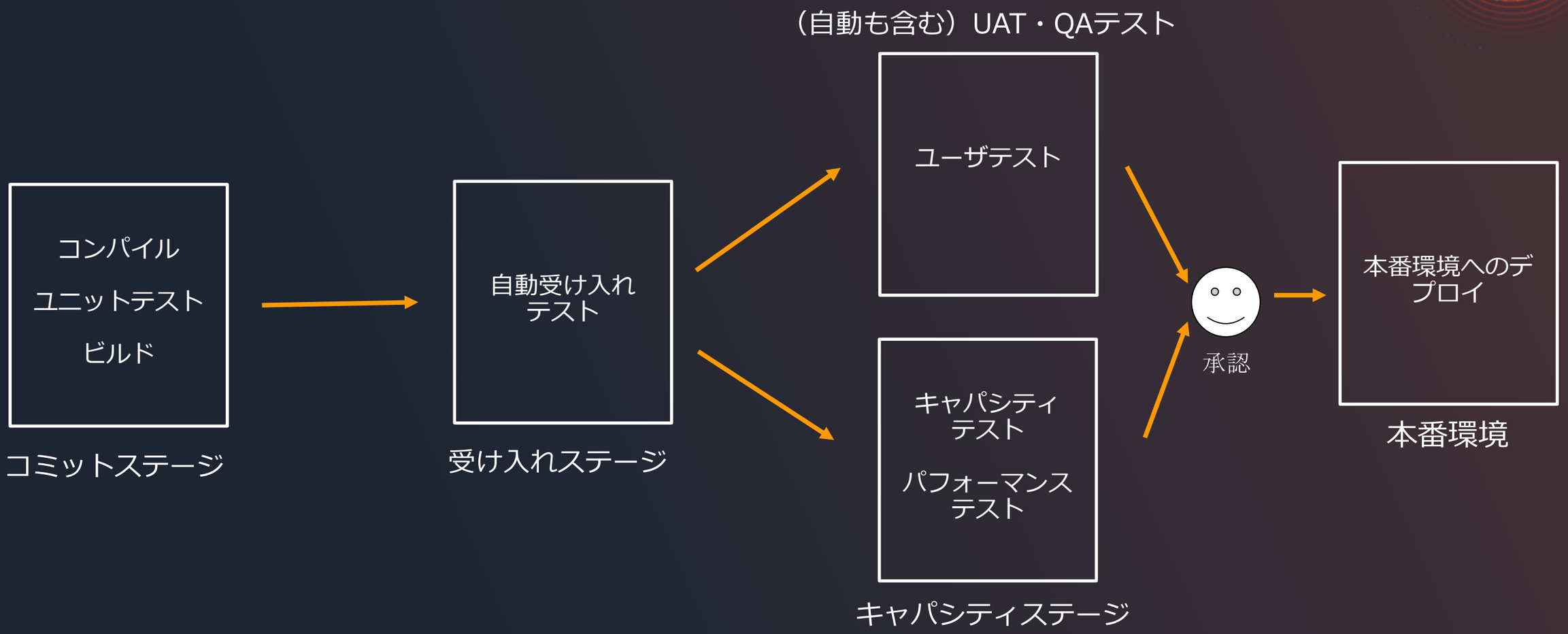
CI：継続的インテグレーション

CD：継続的デリバリー

別のCD：継続的デプロイ

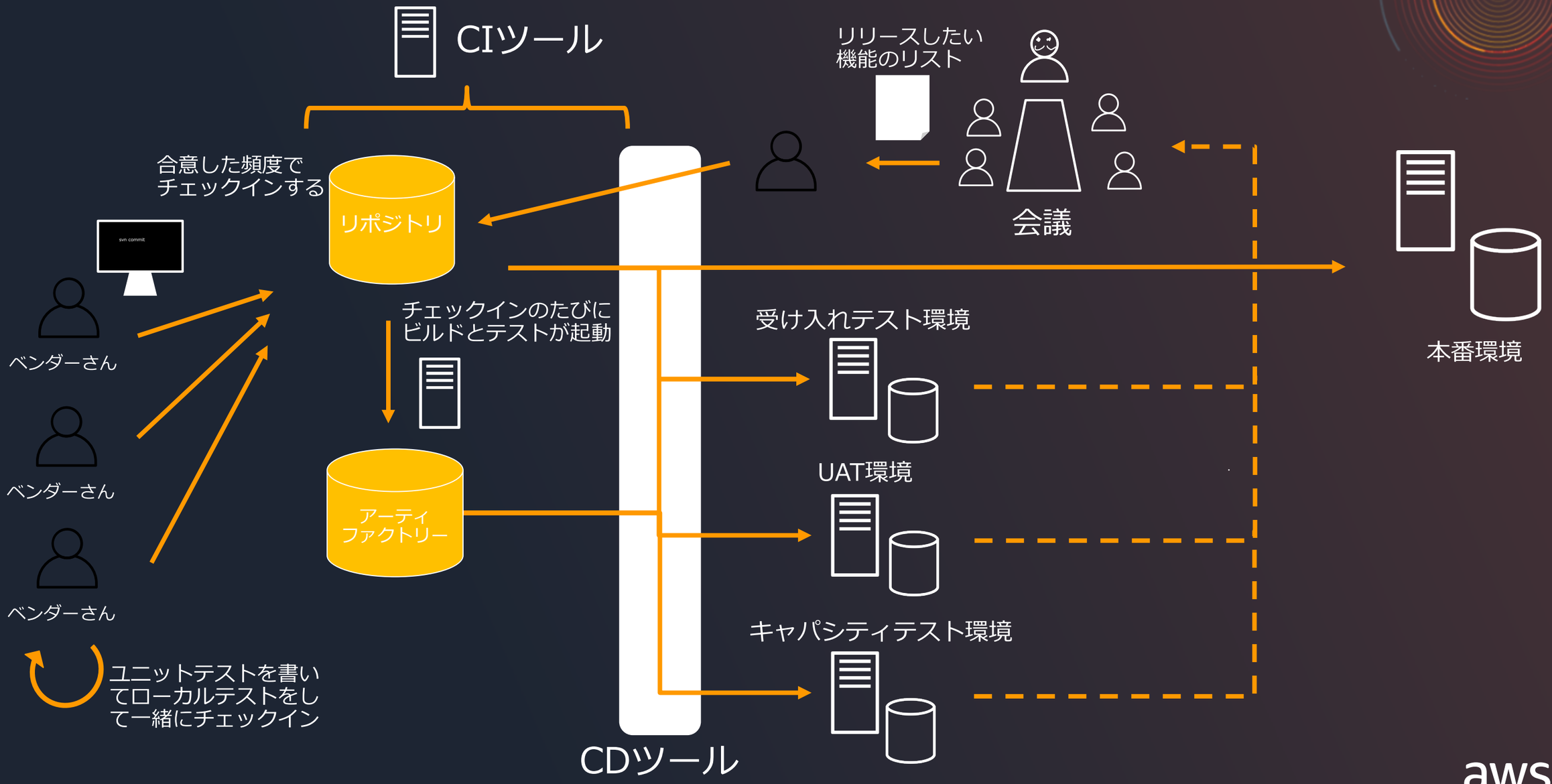
aws DevAx

継続的デリバリー：デプロイメントパイプライン



aws DevAx

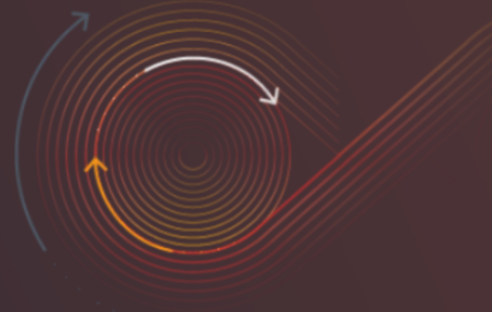
CD：パイプライン構築編



aws DevAx

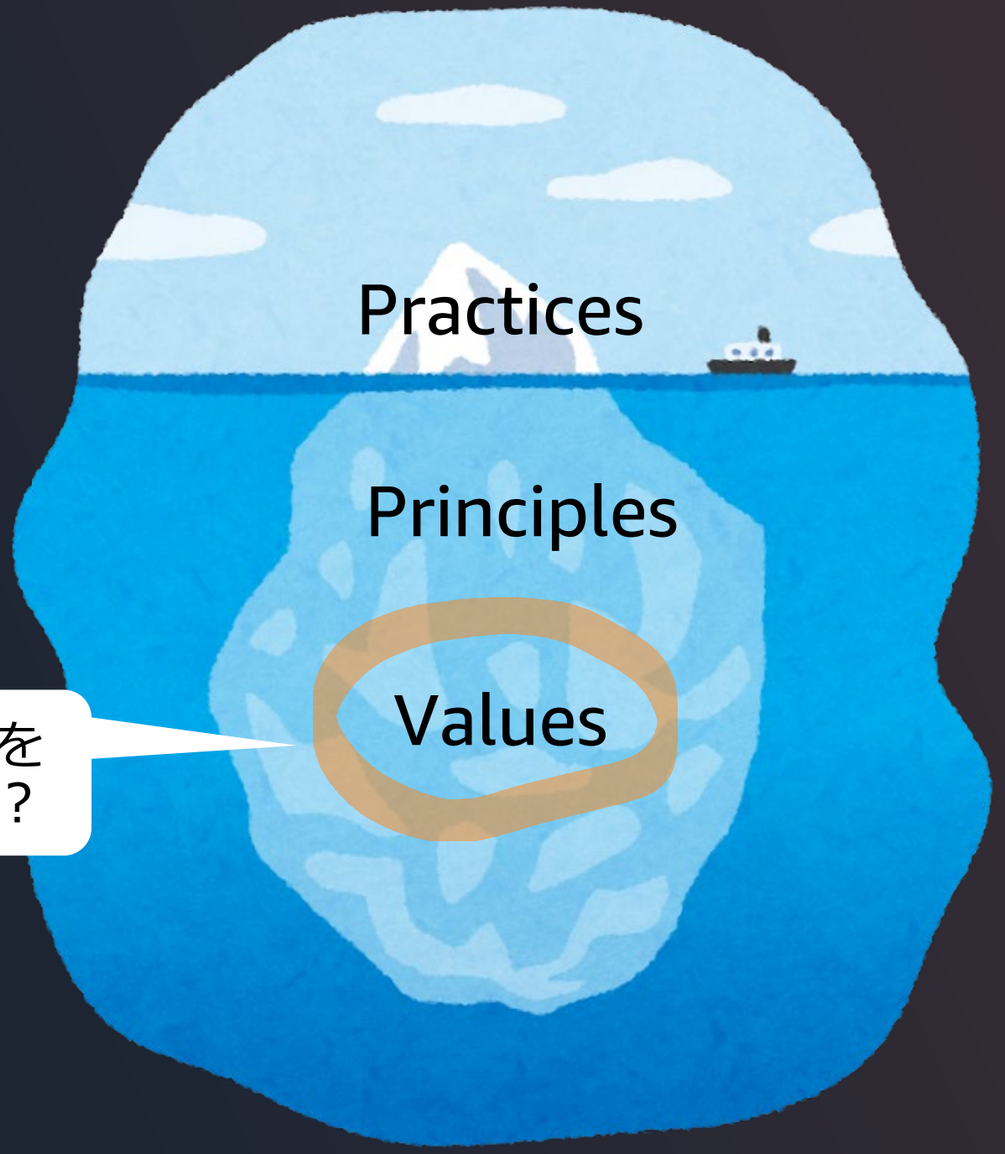
まとめ

CI/CDとは



- 継続的インテグレーション（CI）
  - 継続的にコードをチェックインして開発者の最新のコードがマージされた状態にしておくこと
  - 安全にマージを行うために自動ビルドや自動テストを構築すること
- 継続的デリバリー/デプロイ（CD）
  - デプロイメントパイプラインを構築すること
  - パイプラインの可視化をすること

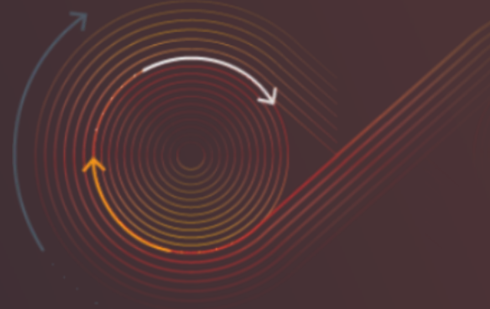
価値・原則・プラクティス



あなたの組織はなにを
価値としていますか？

aws DevAx

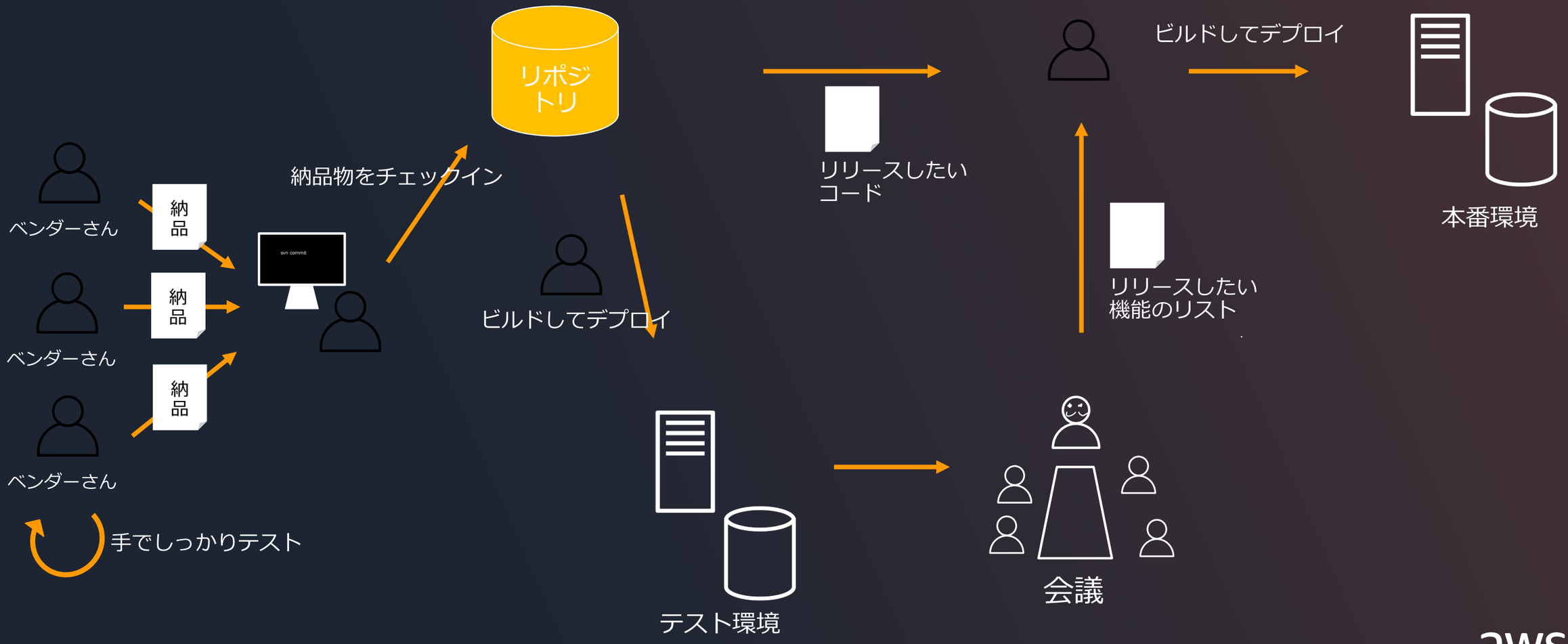
CI/CDの導入



- CI準備編
  1. コードのバージョン管理はできていますか？
  2. ユニットテスト書いていますか？
  3. ビルドは自動化出来ていますか？
- CI接続編
  4. ビルドとテストを自動化する環境はありますか？
  5. Webhookは使っていますか？
  6. チェックインのルールを決めていますか？
- CDステージ構築編
  7. CIできていますか？
  8. テストを人に頼っていませんか？
  9. デプロイは自動化していますか？
- CDパイプライン構築編
  10. 本番環境のデプロイは安全に素早く出来ていますか？
  11. パイプラインの状況が見えていますか？

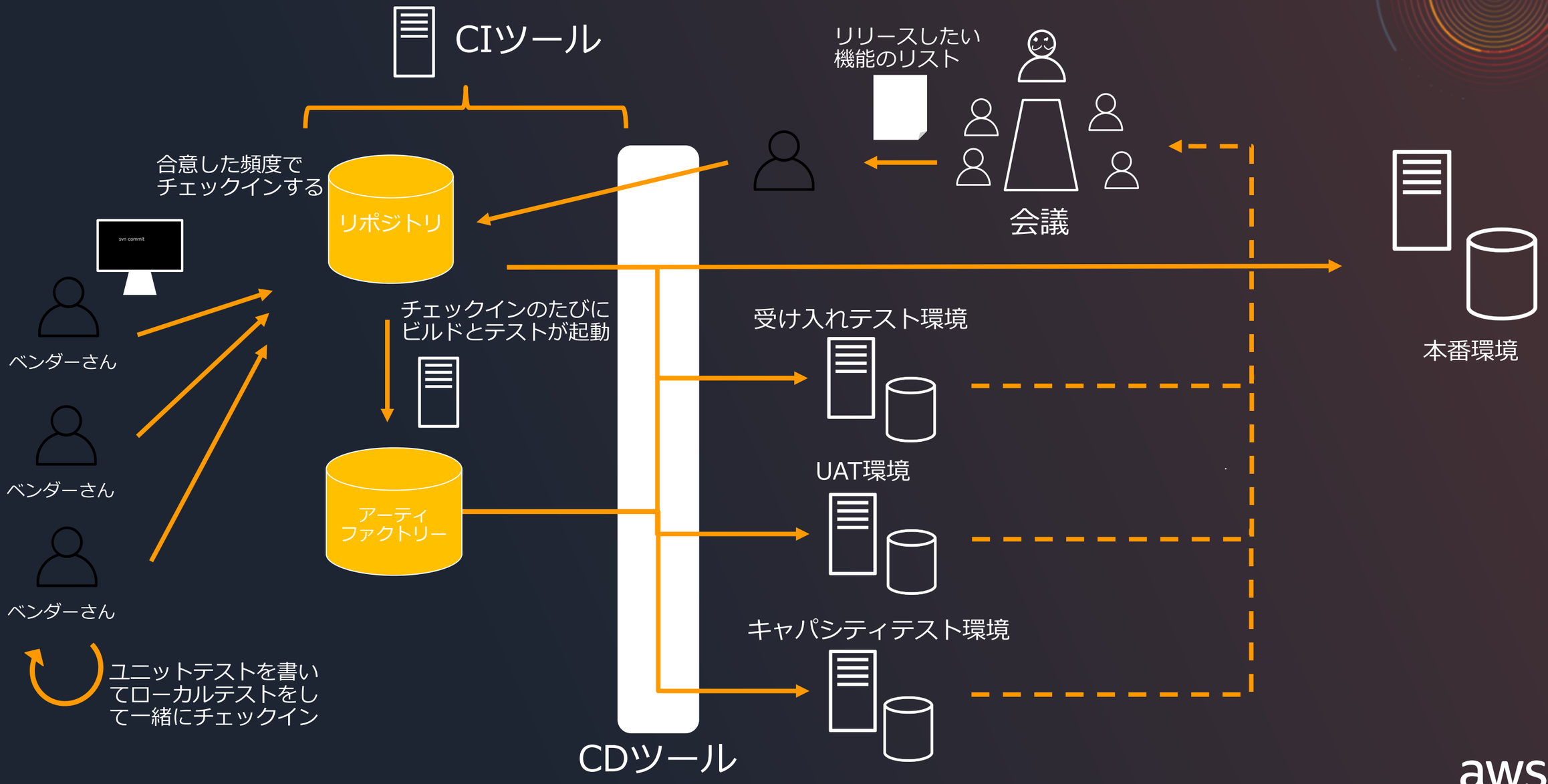
aws DevAx

CI/CD導入前



aws DevAx

CI/CD導入後



aws DevAx

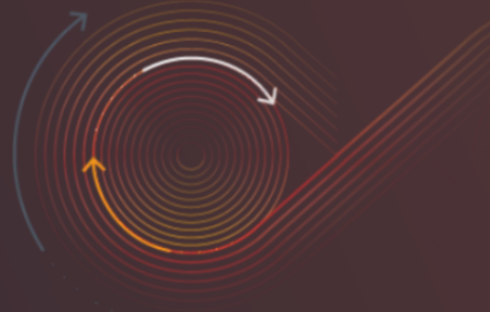
Thank you!

Yuji Nomura

Solutions Architect
Amazon Web Service Japan G.K.

aws DevAx

参考文献



- 継続的デリバリー 信頼できるソフトウェアリリースのための ビルド・テスト・デプロイメントの自動化
  - www.amazon.co.jp/dp/4048707876
- エクストリームプログラミング
  - www.amazon.co.jp/dp/B012UWOLOQ
- CONTINUOUS DELIVERY Blogs
  - https://continuousdelivery.com/
- martinFowler.com
  - https://martinfowler.com/
  - https://martinfowler.com/articles/continuousIntegration.html
  - https://martinfowler.com/articles/branching-patterns.html
- アジャイルソフトウェア開発宣言
  - https://agilemanifesto.org/iso/ja/manifesto.html
  - https://agilemanifesto.org/iso/ja/principles.html

aws DevAx