aws DEV DAY
ONLINE JAPAN

2020.10.20

C - 3

アンチパターンから学ぶクラウドアプリケーションとアジャイル開発の相乗効果

株式会社アンチパターン
代表取締役 小笹佑京

In Partnership with intel

自己紹介

小笹 佑京(Ozasa Yuki)

株式会社アンチパターン
代表取締役

ソフトウェアエンジニア兼経営者

Twitter: @yukiozasa

In Partnership with intel

株式会社アンチパターンについて

Vision:
日本のソフトウェアエンジニアを憧れの職業へ

事業:

1. ソフトウェア開発支援
2. ソフトウェアエンジニア特化型コワーキングスペース
3. ソフトウェアエンジニア教育

In Partnership with intel

本セッションのゴール

- ソフトウェア開発におけるアジリティの重要性について再認識する。
- 登壇者の経験談からアンチパターンを知り、現場の状況と照らし合わせてもらう。
- より良い開発の現場が増えることを期待。

Agenda

- ソフトウェアの時代におけるアジリティの重要性について
- アジリティ向上に資するクラウドとは
- クラウドアプリケーションのアンチパターン(主に登壇者の経験から)
- クラウドとアジャイル開発の相乗効果
- 総括

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

ソフトウェアの時代における アジリティの重要性について

In Partnership with intel

至るところにソフトウェアがある時代

In Partnership with intel

ソフトウェアを扱う企業が評価されている

世界の時価総額順位

	2020/10/07	2010/03/31
1st	Apple Inc.	PetroChina
2nd	Microsoft	Exxon Mobil
3rd	Amazon.com	Microsoft
4th	Alibaba Group	ICBC
5th	Facebook	Apple Inc.

※YahooFinance/Financial Timesのデータを基に登壇者が作成
https://stocks.finance.yahoo.co.jp/us/ranking/
http://media.ft.com/cms/15951a1e-4899-11df-9a5d-00144feab49a.pdf

aws DEV DAY
ONLINE JAPAN

“ソフトウェアファースト”の モノづくりに転換することも必要だ

2020年3月24日
トヨタ自動車株式会社
豊田章男氏

https://monoist.atmarkit.co.jp/mn/articles/2003/25/news070.html

In Partnership with intel

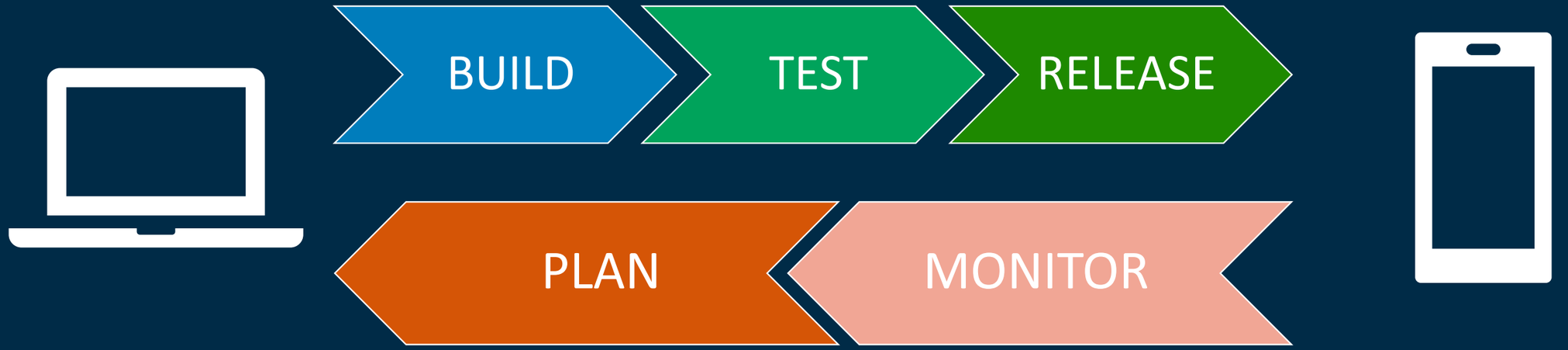
ソフトウェアがVUCAを加速させる

■ソフトウェアの特徴

- 無形で柔軟性が高い

- Volatility（変動性・不安定さ）
- Uncertainty（不確実性・不確定さ）
- Complexity（複雑性）
- Ambiguity（曖昧性・不明確さ）

ソフトウェア時代におけるビジネスのあり方



ソフトウェア開発のアジリティを高めていくことが競合優位性になる

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

アジリティ向上に資するクラウドとは

In Partnership with intel

クラウドとは

クラウドサービスプラットフォームから**インターネット経由**で

- コンピューティング
- データベース
- ストレージ
- アプリケーション

をはじめとした、

さまざまな IT リソースを**オンデマンド**で利用することができるサービスの総称

引用元: https://aws.amazon.com/jp/cloud/

In Partnership with intel

クラウドの活用によって得られる主なメリット

豊富なマネージド
サービス

スケーラビリティ

わずか数分の
デプロイ

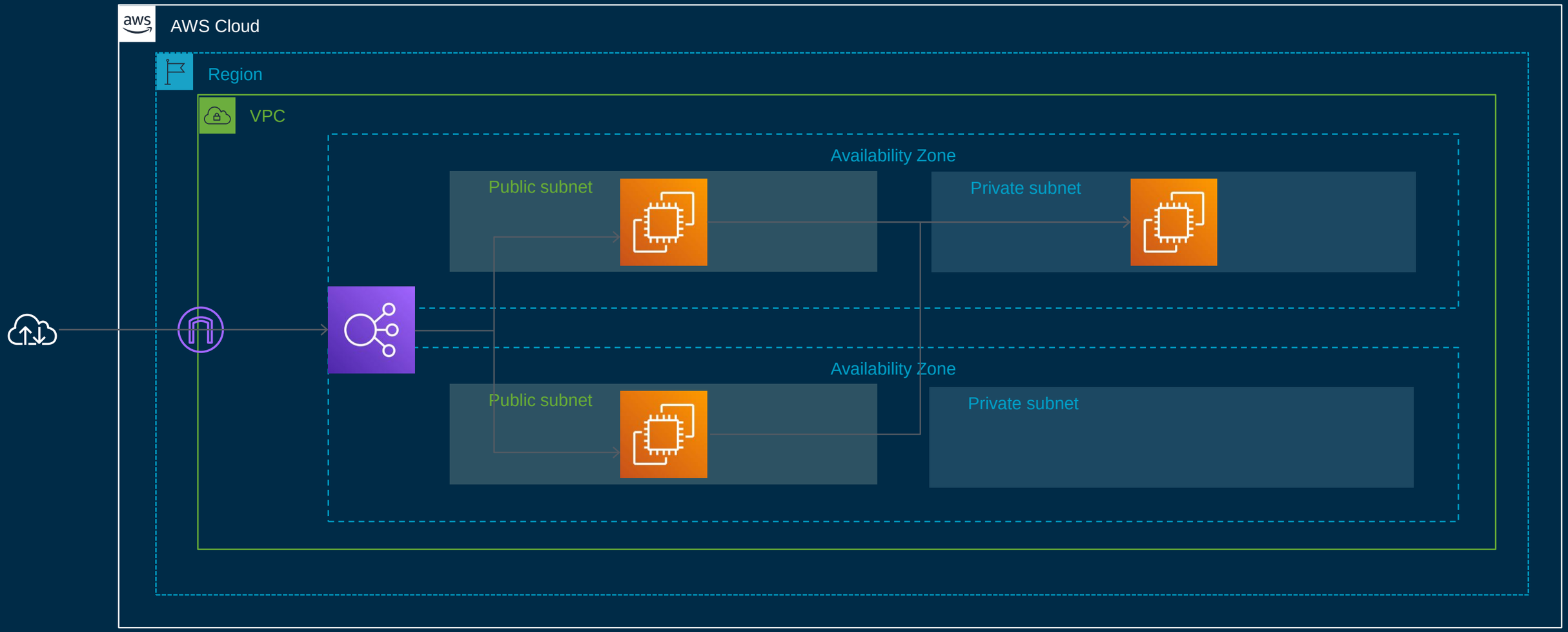
In Partnership with intel

aws DEV DAY
ONLINE JAPAN

クラウド活用アンチパターン

In Partnership with intel

パターン1: EC2インスタンスを使いがち



In Partnership with intel

パターン1: EC2インスタンスを使いがち

- 外注していたシステムの内製化プロジェクトリーダーを担当。
- クラウドベンダーのマネージドサービスを活用せず、EC2上に様々なアプリケーションを構築している。
- 「マネージドサービスを利用すること=ベンダーロックイン」という考え方が土台にある。
- 開発者の**技術力がクラウド活用のキャップ**になってしまっており、馴染みのある技術/自分の理解している範囲の技術を選定してしまう。

このような困難がありました 😢

- アクセス過多による負荷を分散するためにスケールインスケールアウトの計画を立ててスクリプトで実行。
  - アクセスの予測を立てるのに苦闘。運用コストが高い上、予測が外れた時などには・・・。

実行していたスクリプト例

スケール対象のインスタンスからAMIを作成

```
$ aws ec2 create-image --instance-id i-xxxxxxxx --name “任意のAMI名” --no-reboot

{ "ImageId": " ami-xxxxxxxx " }
```

作成したAMIをもとにインスタンスを起動

```
$ aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-name ~/.ssh/id_rsa --security-group-ids sg-xxxxxxxx --subnet-id subnet-xxxxxxxx
```

他にもこのような困難がありました 😢

- データベース(EC2)をどのリージョンのどのアベイラビリティゾーンに立てるかを検討。
  - 結果他の様々なアプリケーションもアベイラビリティゾーンが同じ方が通信コストから考えて、パフォーマンスが高いのでは、ということで耐障害性が低下など・・・。

クラウドネイティブなアーキテクチャへの変更

- Auto Scalingの活用やサーバレス(Lambda, SQS, etc.)の導入により、アプリケーションのスケーラビリティを確保

- データベースサービスにRDSを選定するなど、マルチAZ構成に変更し耐障害性を確保



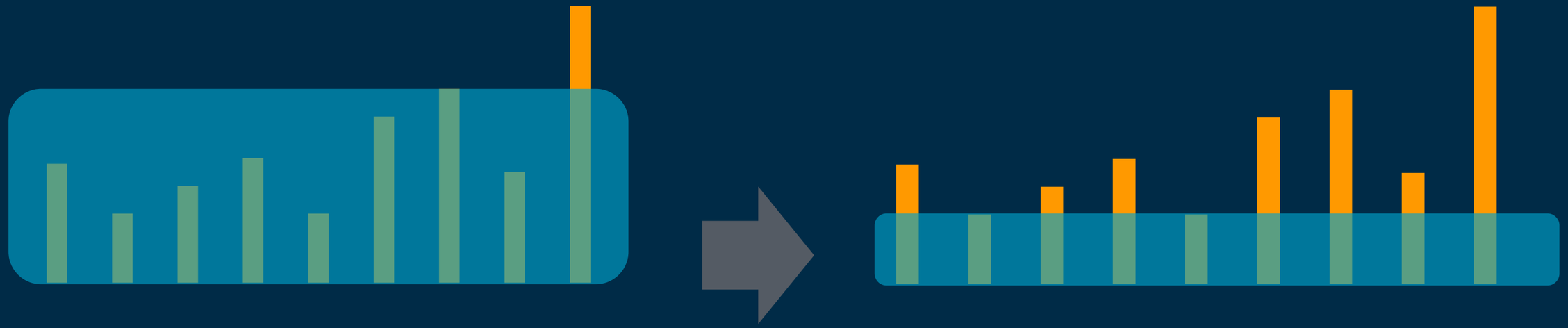
クラウドベンダー(巨人)の肩に乗る

In Partnership with intel

パターン2: リザーブドインスタンス買いがち

- 社内のシステム開発に関する購買にも携わっていた時のこと。
- とりあえず如何に安く運用するかという議論になった際、予算確保の兼ね合いなどもあり、予測性がつきやすいという考えのもと、**ほとんどのインスタンスに対してリザーブドインスタンスの購入を検討**。
- いつどのインスタンスタイプを購入したのか管理するのが煩雑になり、運用コストが増大。(Reserved Instance Reportingはまだなかった)
- 折角買ったから活かさないといけない、という**コンコルド効果**が働き、**リザーブドインスタンスを活かす目的でアーキテクチャを選定**。

クラウドの考えに沿ってない利用は避けるべき



もったいないRIの買い方(=予測/調達)

効果的なRIの買い方(=実績/利用)

スケーラビリティを犠牲にしない活用を検討する

In Partnership with intel

パターン3: CI/CD環境が整っていない

- 内製化プロジェクト中のできごとです。
- Subversionでバージョン管理を実施。単一リポジトリ構成のため、複数人での開発に向いていない。
- SCPでファイルアップロード。
- 手順書にあるコマンドを、配布先の環境に合わせて、手動で書き換えて実行するようなデプロイ環境。

当時のデプロイ作業を再現した例

```
$ zip -r src.zip ./src

$ scp -i ~/.ssh/id_rsa src.zip username@remote.example.com:/remote/directory

$ ssh -i ~/.ssh/id_rsa username@remote.example.com

$ cd /remote/directory

$ unzip src.zip

$ cp -rf ./src /var/app
```

このような困難がありました 

- リリース時にトラブルが発生。
  - ビジネス上の機会損失に繋がってしまう。

- 上記もあり、**リリースに対する心理的負荷が高くなる**。
  - リリース作業をなるべく避けたくなる=デリバリーが肥大化かつ遅延しがちなど・・・
  - アジリティが低下

CI/CD環境の整備

- Git/GitHubを利用。GitHub Flowをもとにしてブランチ等の利用方法を明確化。チーム標準の開発フローを定義。

- デプロイ作業をCircleCIを活用して**自動化**。GitHubのリリースタグを切ったらデプロイがなされるように変更。

安心安全でかつ高速にソフトウェアをデリバリーする

In Partnership with intel

パターン4: 本番と大きな乖離のある検証環境

- 開発者として新規機能実装を担当。
- 複数のテーブルに跨ったデータを元に新たなデータを算出する機能が要望された。
- 検討の結果ストアドプロシージャを実装することを決定。
- 検証環境のアーキテクチャは本番と同じ構成になっていた。

データベーストリガー実装例: ロジックの実装

```
CREATE OR REPLACE FUNCTION calculation_summary() RETURNS TRIGGER
AS $summary $
    DECLARE
        tmp_summary_data      integer;
BEGIN
 -- データの抽出/summaryデータの算出/算出したデータの格納
END;
$ calculation_summary $ LANGUAGE plpgsql;
```

データベーストリガー実装例: トリガーの設定

```
-- factテーブルにinsertかupdateが走ったら関数を実行するように設定

CREATE TRIGGER calculation_summary_trigger

        AFTER INSERT OR

        UPDATE ON fact

        FOR EACH ROW

        EXECUTE FUNCTION calculation_summary();
```

このような困難がありました 😢

- 検証環境での動作を確認後、リリース。

- **検証環境と本番環境では格納しているデータ量が異なり**、リリース直後からデータベースのCPU使用率が想定以上に増大。
  それによりデータベース全体のパフォーマンスが低下。

- 協議の結果、**当該機能の提供を中止**。再度実装方法を検討することに。

検証に適う環境を準備しておかなければならない

パターン5: 顧客目線ではないサービス選定

- システムの開発リーダーとして新規プロジェクトを担当。

- 新規プロジェクトにおいては、クラウドベンターのキャンペーンを活用することが決定事項。営業/マーケティング的な支援を得られるとのことで指定の新規サービスを活用する方向で調整。

- 指定の**新規サービスに合わせたリアーキテクティング**を実行。

このような困難がありました 😢

- 障害が頻発。SLAを割るほどのパフォーマンスの低下。
- リアーキテクティングの際にモニタリング環境は構築していたものの、障害発生ポイントがマネージドサービスの領域だったため、問題解決へは技術サポートへの問い合わせが限界。
- **重大な機会損失**及び既存顧客への説明等システム以外のコストが発生。

得た教訓

- 価格やマーケティングなど、**システム以外の事案を中心にアーキテクチャを設計するべきではない**。
- **そもそもSLOを満たせない技術の選定はするべきではない**。また、PoCは入念に行うべき。

それが顧客のためになるかを問うことが大事

5つのアンチパターン

1. EC2インスタンス使いがち
2. リザーブドインスタンス買いがち
3. CI/CD環境が整っていない
4. 本番と大きな乖離のある検証環境
5. 顧客目線ではないサービス選定

顧客目線での
積極的なクラウドの活用が
アジリティ向上に寄与する

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

クラウドとアジャイル開発の相乗効果

In Partnership with intel

起きてしまうこと

クラウドの活用により、インフラが整い、顧客への価値提供が迅速にできる状態になっても、その上で動作するアプリケーションの開発にアジリティがない状態では片手落ちになってしまう。

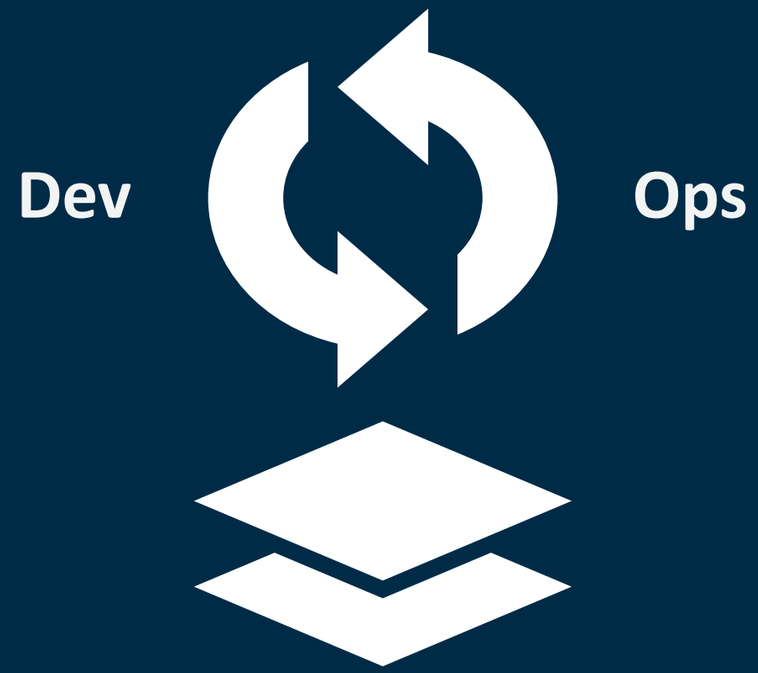
開発に6ヶ月かかったものも、
1時間あればリリースできますよ。

ソフトウェア開発全体のアジリティを高める

ソフトウェアの柔軟性を活かし、
顧客に価値を提供し続けるための開発Tips

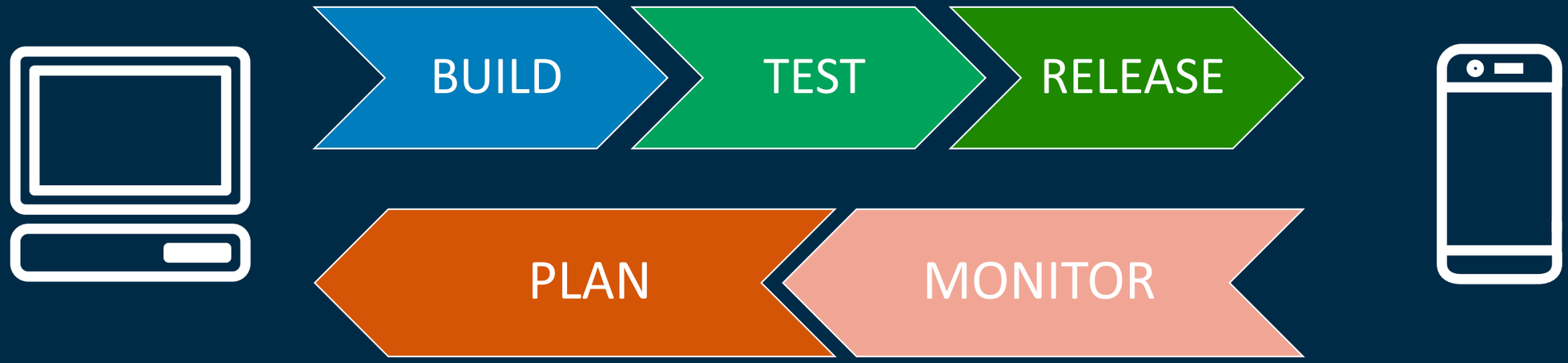
- バージョン管理
- 継続的インテグレーション
- 継続的デリバリー
- テスト駆動開発
- Infrastructure as Code
- マイクロサービス

etc…



In Partnership with intel

顧客からのフィードバックを中心に据える



In Partnership with

Agile Manifesto

プロセスやツールよりも**個人との対話**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。

引用: https://agilemanifesto.org/iso/ja/manifesto.html

Let’s Scrum

- アジャイル開発の手法としてスクラムを導入。
- 機能はなるべく細かくリリースをしていくように調整。
- 都度リファクタリングやリアーキテクティングを実行。

総括

- ソフトウェアを中心としたビジネスが今後も拡大する。
- ソフトウェア開発のアジリティが競合優位性に繋がる。
- アジリティ向上の手段としてのアジャイル開発を進める上で、クラウドの活用は必須である。
- Agile Manifestoはとても良くできている。

おわりに

- エンジニアの技術力が、ソフトウェアの価値に直結する。
  - 初学者だった私も手を抜いていたわけではなく、その時出せる最大限のアウトプットだった。

- **ソフトウェアに投資するということはエンジニアに投資することに等しい。**
  - エンジニアは自己投資を怠らない
  - 経営者はエンジニアへの投資を怠らない

aws DEV DAY
ONLINE JAPAN

Thank you!

株式会社アンチパターン
小笹 佑京 (Ozasa Yuki)

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

Please complete the session survey in the mobile app.

In Partnership with intel