



# DevOps Hands-on

## Service Dev Engineer TAIKEN

# ハンズオンについて

できるだけ参加者の皆さんにDevOpsの環境を実際に触っていただく予定でしたが、ライセンスやユーザ登録の都合で直に触っていただくのを断念しました。

Zoom越しにDevOpsの環境を操作するか、  
参加者の皆さんに意見をもらってスタッフが操作するかたちで  
疑似的に体験いただくかたちとさせていただきます。

# サービス開発エンジニア体験

サービス/プロダクトの開発に欠かせない  
アプリ開発とDevOpsを体験してみませんか？

9月 SPAハンズオン

10月 APIハンズオン

11月 モバイルハンズオン



12月 DevOpsハンズオン



# スタッフ紹介

TIS株式会社

テクノロジー & イノベーション本部

テクノロジー & エンジニアセンター

伊藤 清人@会津若松

世古 雅也@会津若松

藤田 佳樹@東京



# お願い

Zoomで名前（ニックネームも可）を分かるようにしてください。  
オンライン開催なのでリアクションは大きな動きをお願いします！  
周りの音が入り込まないようにお願いします。

今後の改善等に活用したいので  
ハンズオン終了後のアンケートにご協力をお願いします。

# DevOpsのやり方を学ぶハンズオン

# ハンズオンのゴール

DevOpsのやり方を体験するがゴールです。

DevOpsは改善活動がメインなので、こうすればDevOpsだ、といったものはありません。  
チームやプロジェクトの状況に合わせて技術面と文化面から改善活動に取り組むのがDevOpsです。

このハンズオンではDevOpsの活動をやりやすくするために構築した環境を使って、  
主に技術面からDevOpsのやり方を体験していきます。

皆さんが作業しただけにならず、皆さんにDevOpsのやり方を持ち帰ってもらえるように頑張ります！

# ハンズオンの進め方

あるサービス開発チームに参画するような状況を想像してください。

そのチームはDevOpsに取り組んでいて、すでにDevOps活動をしやすい環境を構築済みです。

DevOpsの概要を学習し、チームメンバとしてDevOps活動に参加できるように開発～デプロイ～運用の流れを一通り体験しましょう。

スタッフが既存チームメンバ、皆さんが新しくチームに参加したメンバとしてハンズオンを進行します。



# ハンズオンのスケジュール（全体120分）

オープニング（15分）



DevOps概要（25分）

DevOps入門  
DevOps活動をしやすい環境  
5分休憩



DevOpsハンズオン（60分）

ハンズオンの題材  
チーム開発  
CI/CD  
5分休憩  
モニタリング  
インシデント管理  
Epona



クロージング（10分）

アンケート（10分）

# DevOps入門

# DevOpsの歴史

6 Minutes DevOps: DevOps の歴史

<https://channel9.msdn.com/Blogs/livedevopsinjapan/6min-DevOps-1>

リリースサイクル、早く、頻繁に、安全に

アジャイル開発、1～2wで動作するアプリ

→エンドゲーム（QA、受入テスト）

→QAをチームに、テスト自動化

→インフラ引き渡し

→インフラ自動化、Webオペレーション（クラウド）



# 10 deploys per day (2009年)

10 deploys per day

<https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr>

155 people clipped this slide

10 deploys per day  
Dev & ops cooperation at Flickr

John Allspaw & Paul Hammond  
Velocity 2009

1 of 78

10+ Deploys Per Day: Dev and Ops Cooperation at Flickr

950,967 views

12 people clipped this slide

Dev and Ops

17 of 78

10+ Deploys Per Day: Dev and Ops Cooperation at Flickr

950,967 views

61 people clipped this slide

1. Automated infrastructure
2. Shared version control
3. One step build and deploy
4. Feature flags
5. Shared metrics
6. IRC and IM robots

1. Respect
2. Trust
3. Healthy attitude about failure
4. Avoiding Blame

76 of 78

10+ Deploys Per Day: Dev and Ops Cooperation at Flickr

950,967 views

# DevOps

明確な定義はないです。

開発と運用が一丸となってプロダクト/サービスの価値を高めるための改善活動に取り組むこと。

プロダクト/サービスの価値を高める

→BMLループ (Build/Measure/Learn)

"体系的" に開発サイクルを回して "効果的" に学びを得るには

<https://techlife.cookpad.com/entry/2018/02/10/150709>



# DevOpsのパフォーマンス

エリート DevOps チームであることを Four Keys プロジェクトで確認する

<https://cloud.google.com/blog/ja/products/gcp/using-the-four-keys-to-measure-your-devops-performance>

デプロイの頻度

- 組織による正常な本番環境へのリリースの頻度

変更のリードタイム

- commit から本番環境稼働までの所要時間

変更障害率

- デプロイが原因で本番環境で障害が発生する割合 (%)

サービス復元時間

- 組織が本番環境での障害から回復するのにかかる時間

# DevOpsをはじめ

## バリューストリームマッピング

アイデアが生まれてから、利用者の手元に価値が届けられるまでの流れを可視化するツールです。

バリューストリームマッピングをやってみた

<https://dev.classmethod.jp/articles/value-stream-mapping/>

<よくある話し>

承認プロセスが多く待ち時間が多い。

チームが分かれているため引き渡しに時間がかかる。

手作業でやっているので時間がかかる。

×必要ですか？

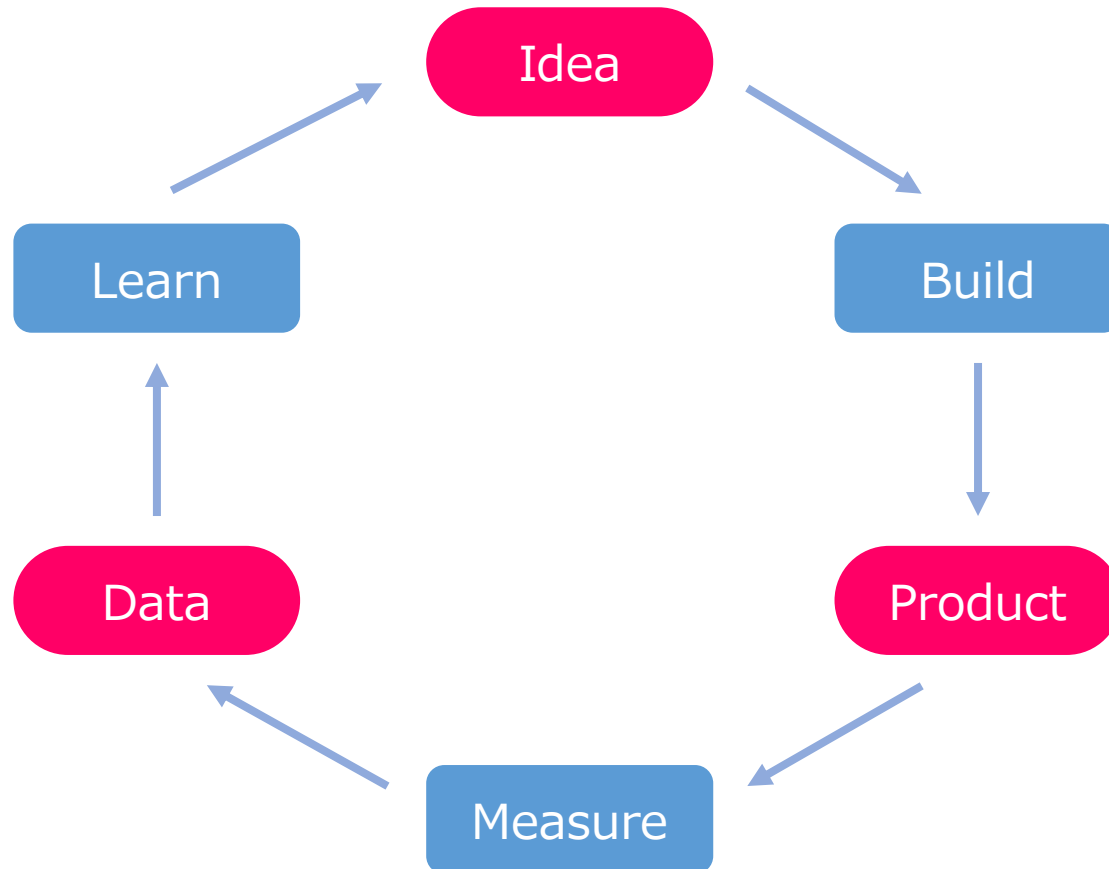
○価値を生み出していますか？

質問タイム

# DevOps活動をしやすい環境

# DevOps活動

BMLループを出発点にして考えてみましょう。



## DevOps

明確な定義はないです。

開発と運用がー丸となってプロダクト/サービスの価値を高めるための改善活動に取り組むこと。

プロダクト/サービスの価値を高める

→BMLループ (Build/Measure/Learn)

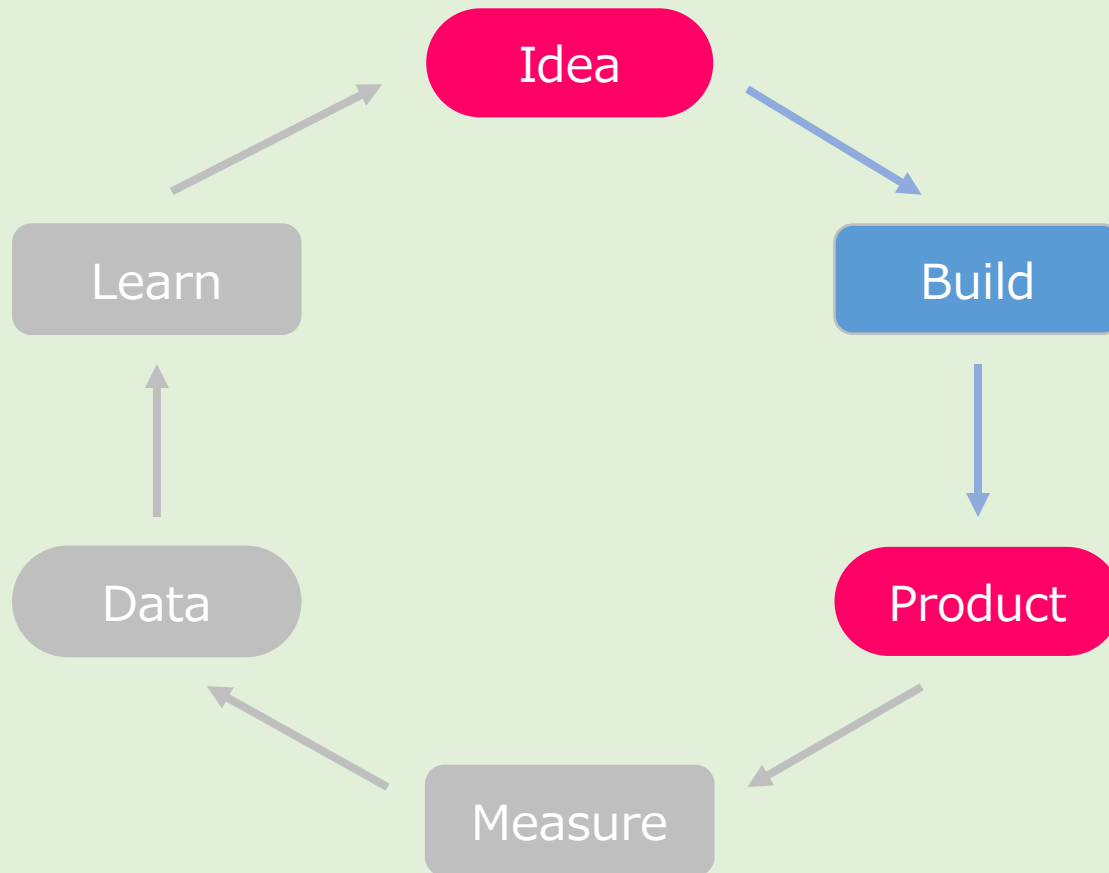
"体系的" に開発サイクルを回して "効果的" に学びを得るには

<https://techlife.cookpad.com/entry/2018/02/10/150709>

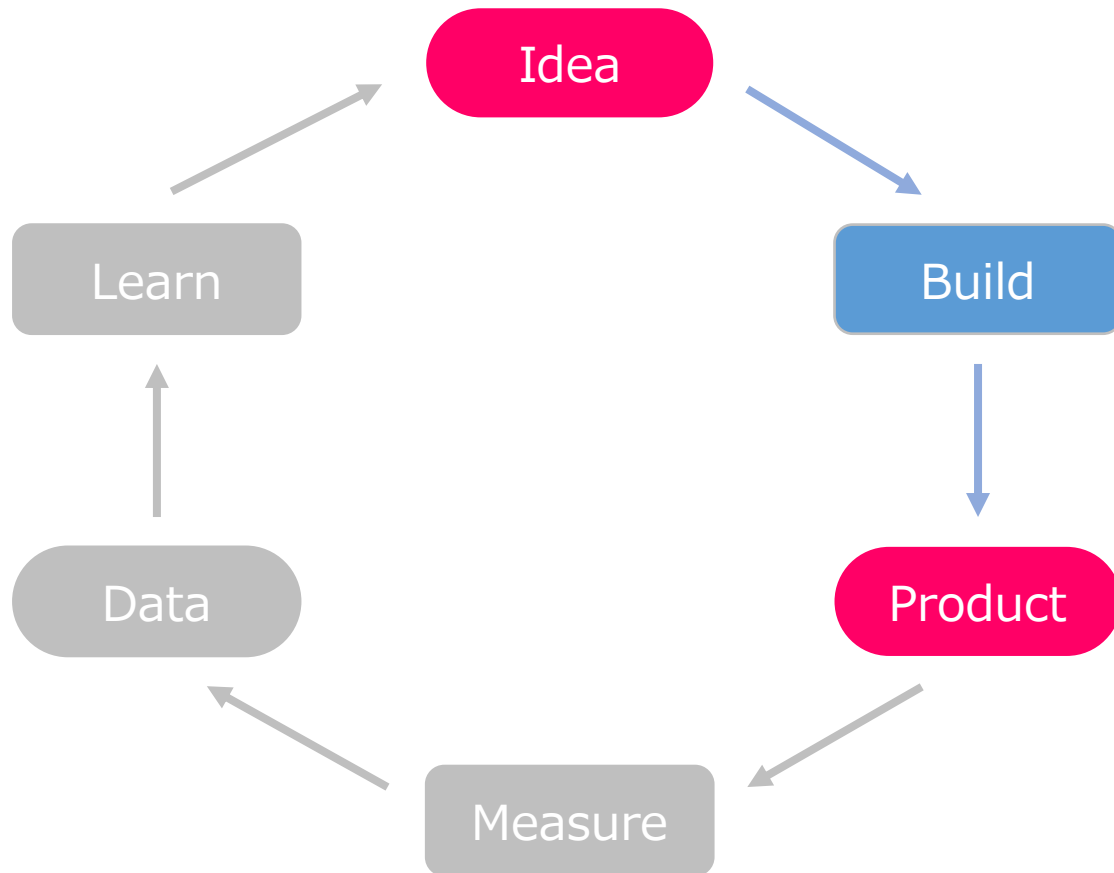
14



アイデアをユーザに届けるために何が必要ですか？



# Build



アイデアをユーザに届けるために何が必要ですか？

課題/バグ管理（ITS/BTS）

かんばん、バックログ

バージョン管理（VCS）

Git

ピアレビュー

プルリクエスト/マージリクエスト（PR/MR）

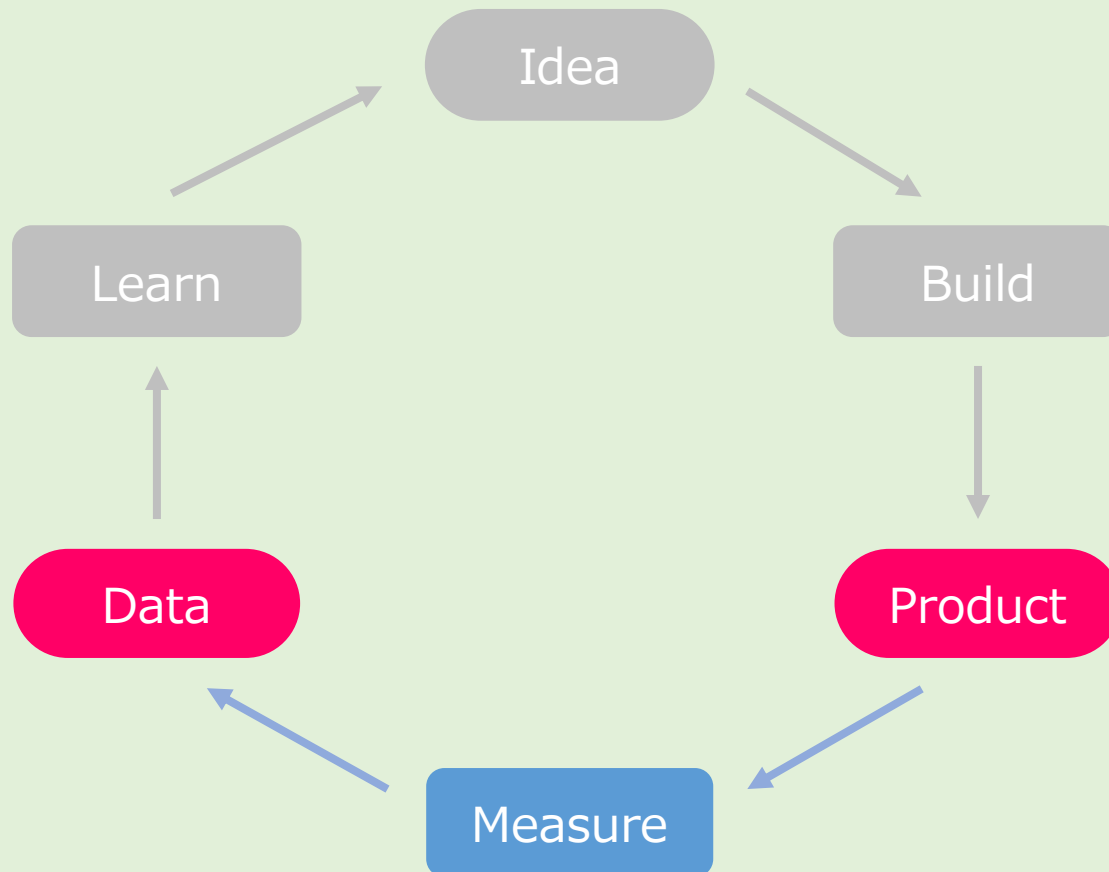
継続的インテグレーション（CI）

パイプライン（ビルド、テスト等の自動化）

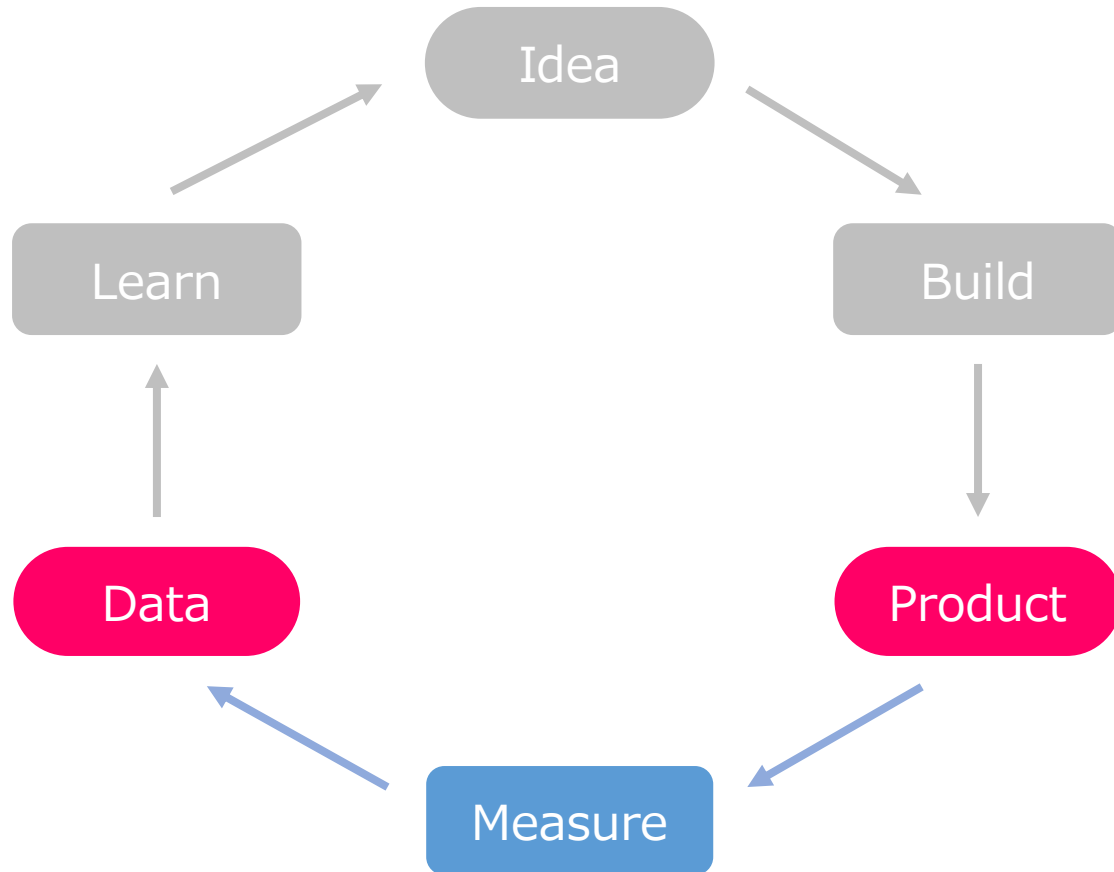
継続的デリバリ（CD）

パイプライン（無停止デプロイ、リリース承認）

利用状況を把握するために何が必要ですか？



# Measure



利用状況を把握するために何が必要ですか？

モニタリング

システム/ビジネスメトリクスの可視化、ログの検索

インシデント管理

アラート通知、アサイン管理、エスカレーション

バックアップ

データのバックアップと復元

セキュリティ

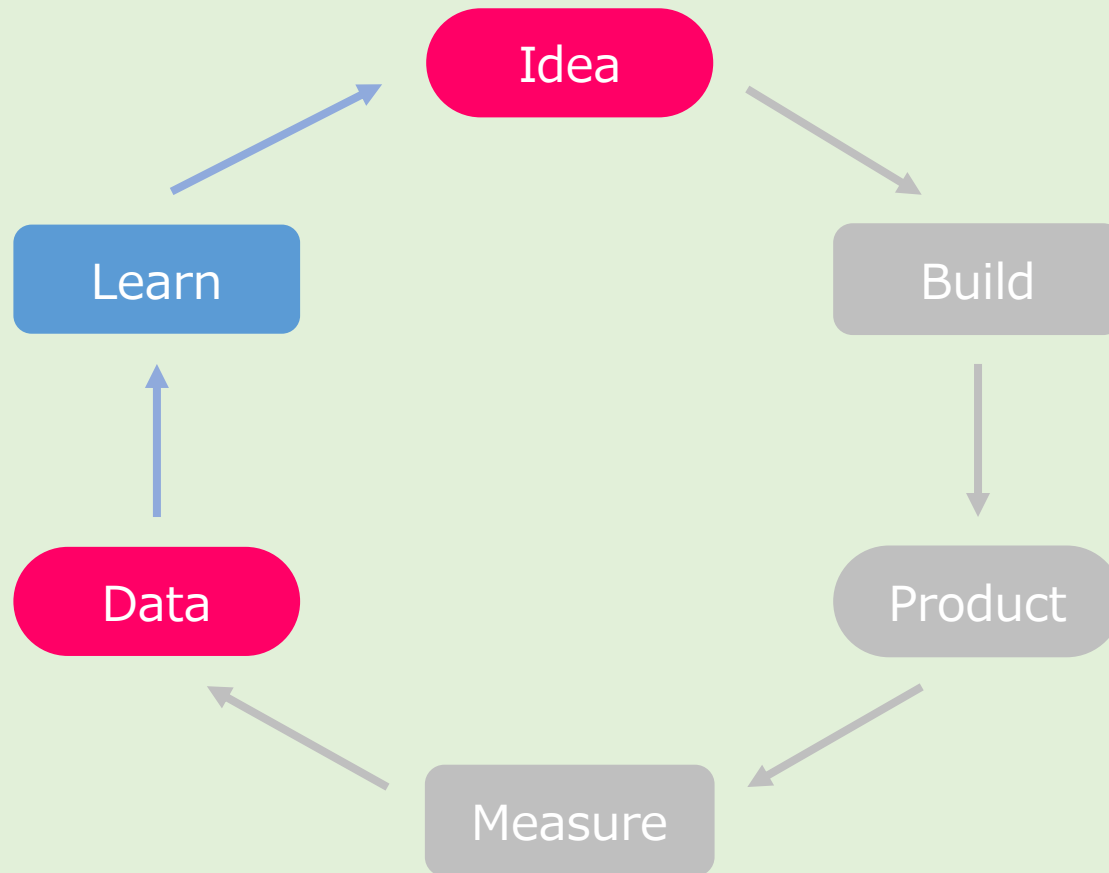
リモートから本番運用、本番運用の承認

権限管理、監査証跡

構成変更のルール違反検知

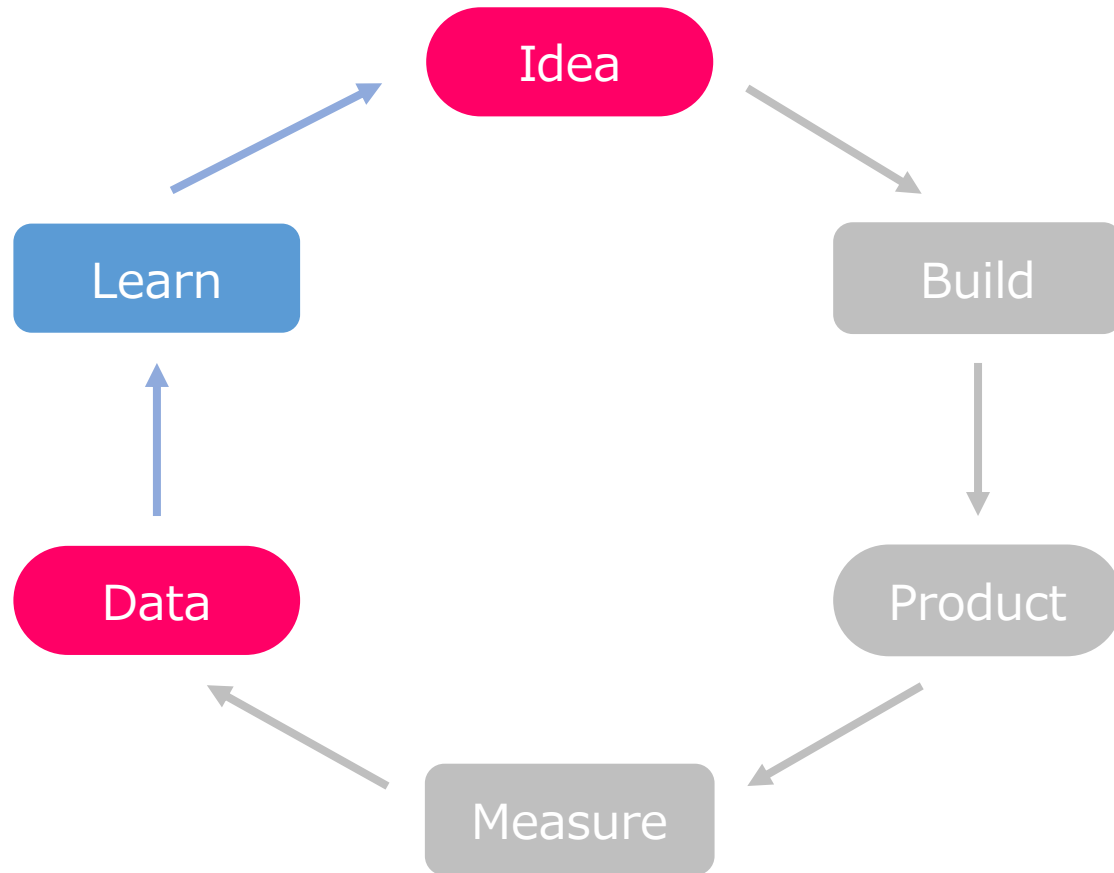
第三者の不正アクセス防止

アイデアを出すために何が必要ですか？





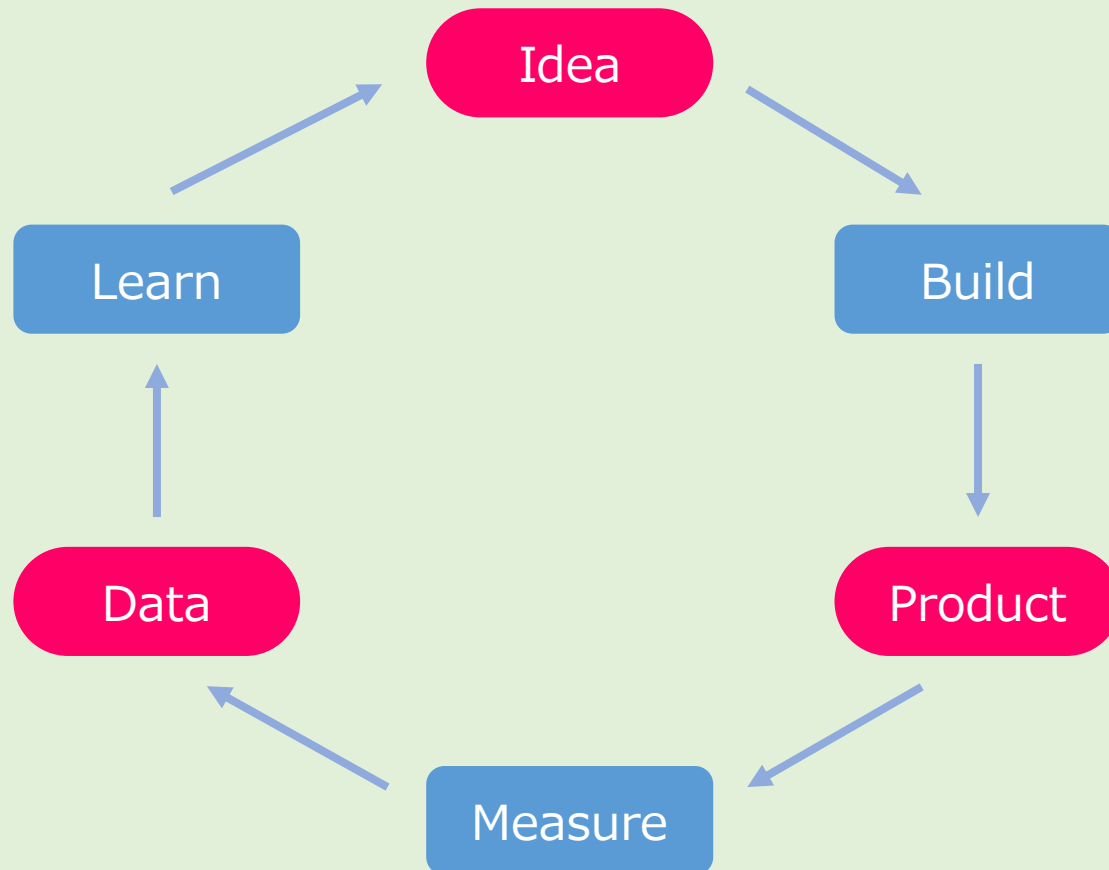
# Learn



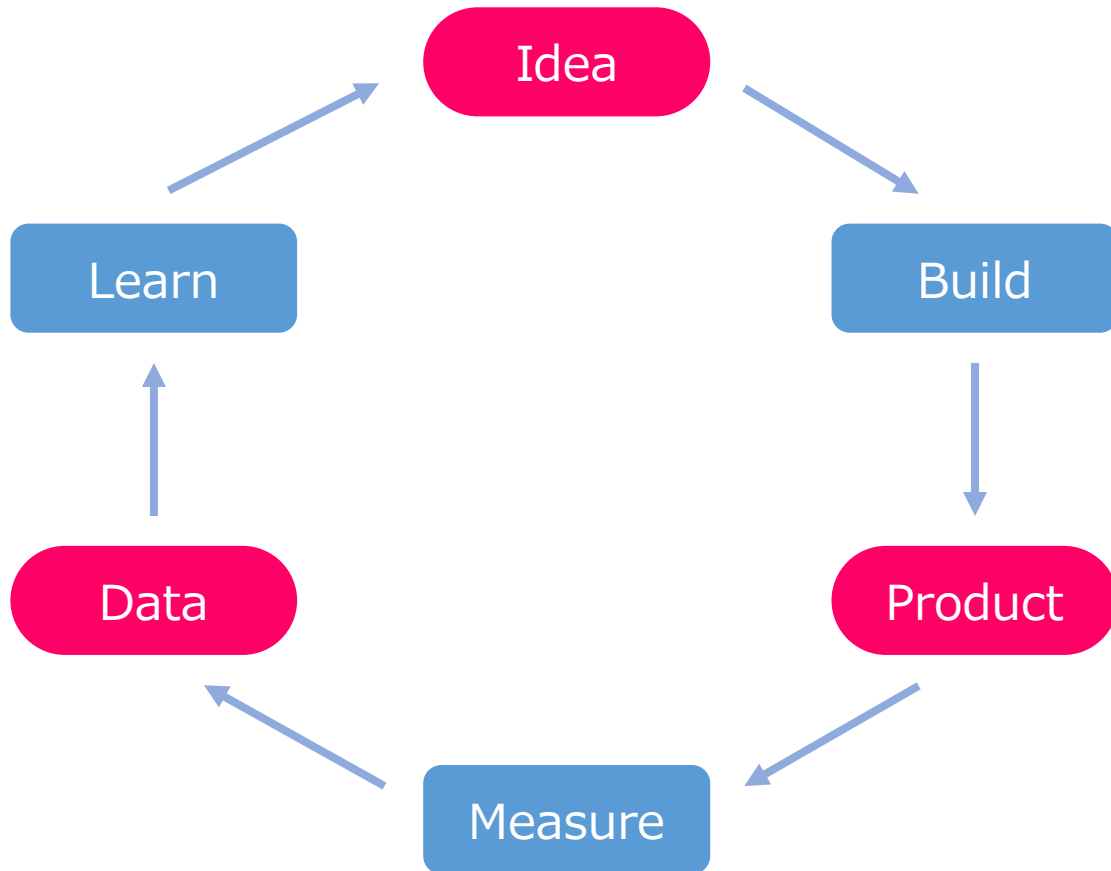
アイデアを出すために何が必要ですか？

サービスの仮説検証で失敗してしまうのはなぜか？  
Cookpadが実践する“体系的”開発サイクルの作り方  
<https://logmi.jp/tech/articles/282800>

全体を活性化するために何が必要ですか？



# BMLループ°



全体を活性化するために何が必要ですか？

## コミュニケーション

コミュニケーションハブとしてのチャット

チャットにすべての情報を集約

コミュニケーションの過程・結果を残す

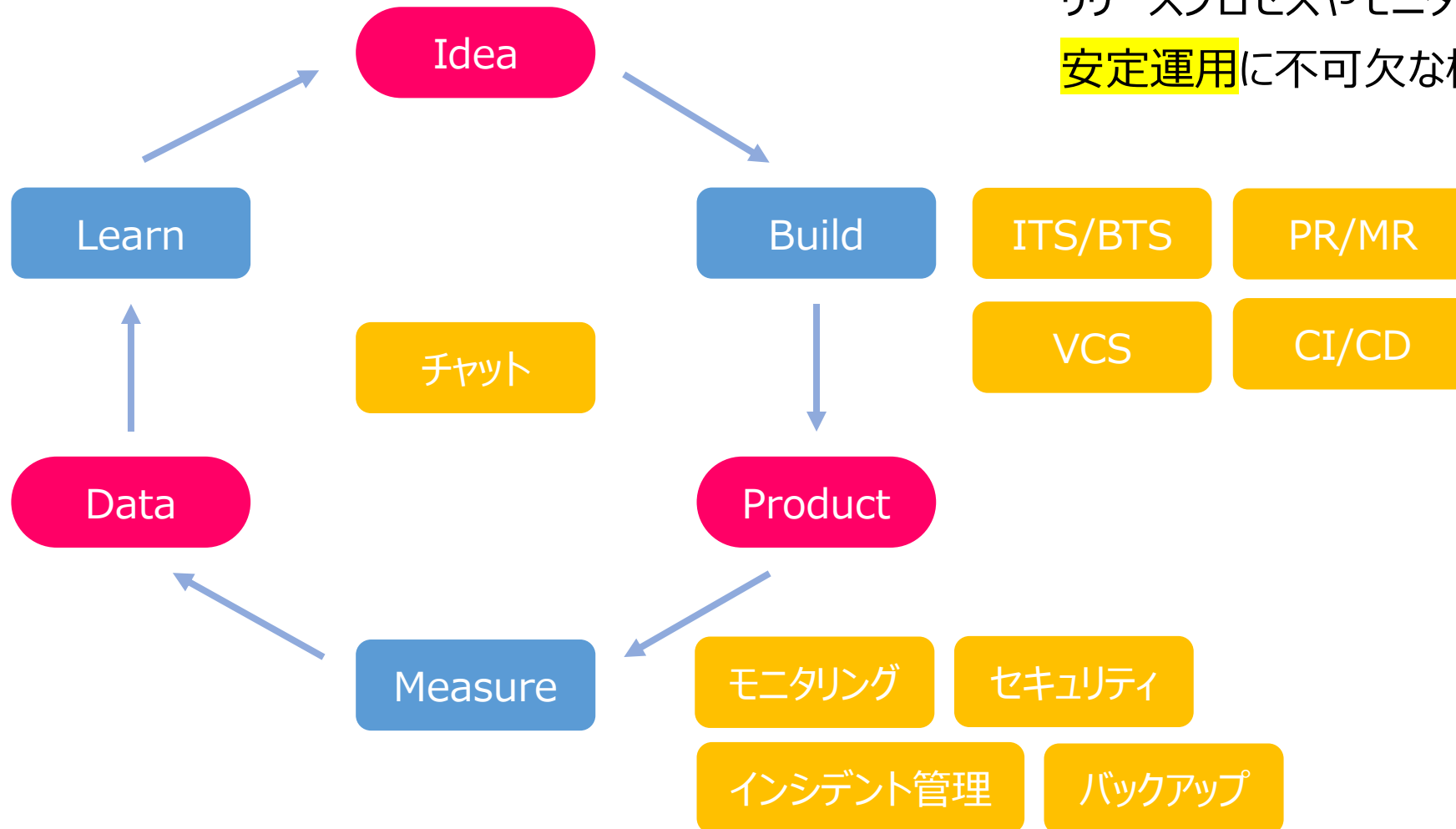
PR/MRの状態、CI/CDの結果を通知

障害通知やアラートを通知

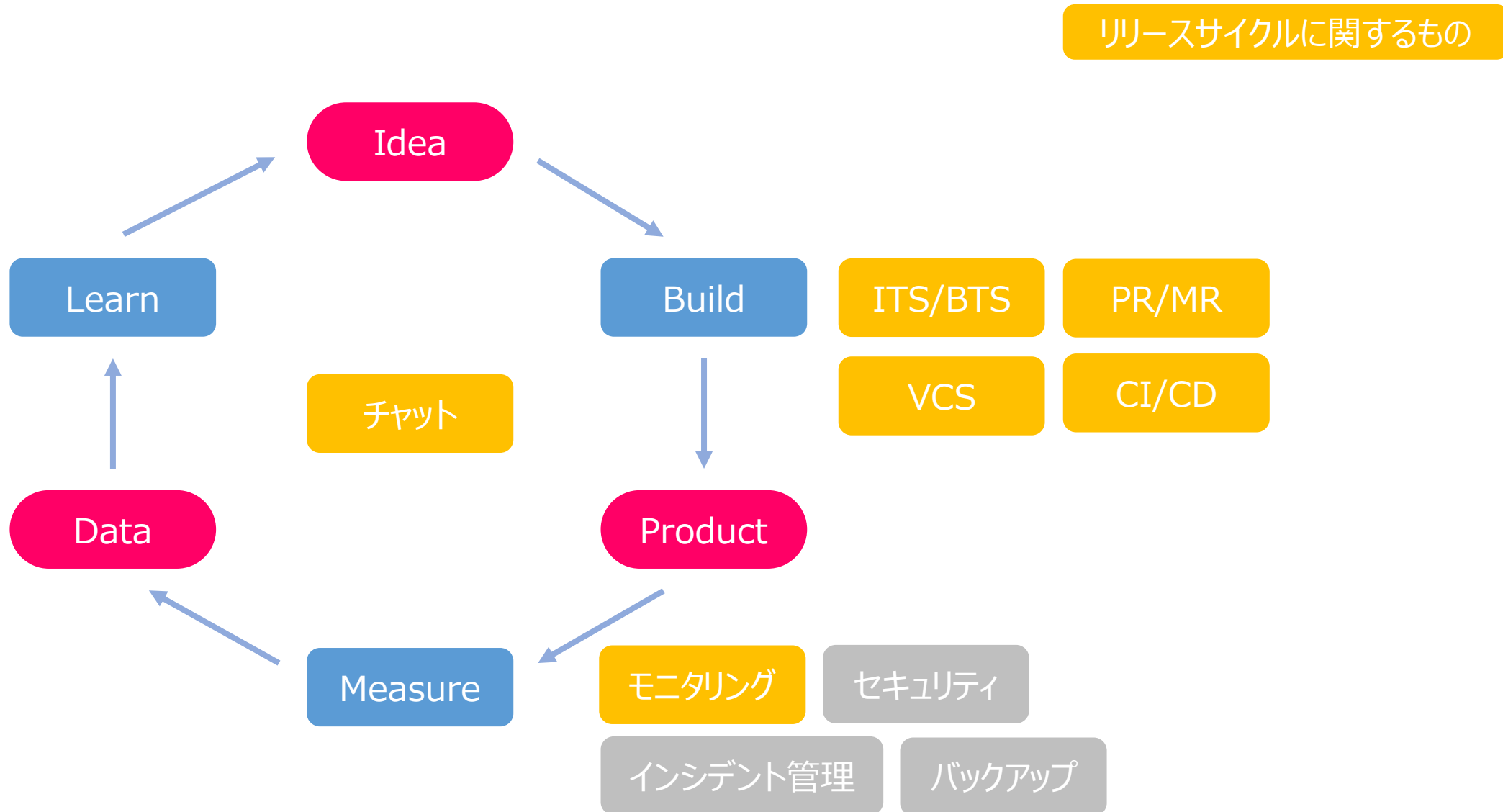
システム/ビジネスメトリクスを通知

# DevOps活動をしやすい環境

コミュニケーションハブとしてチャットに情報が集約され、  
リリースプロセスやモニタリングは自動化され、  
安定運用に不可欠な機能が備わっている環境です。

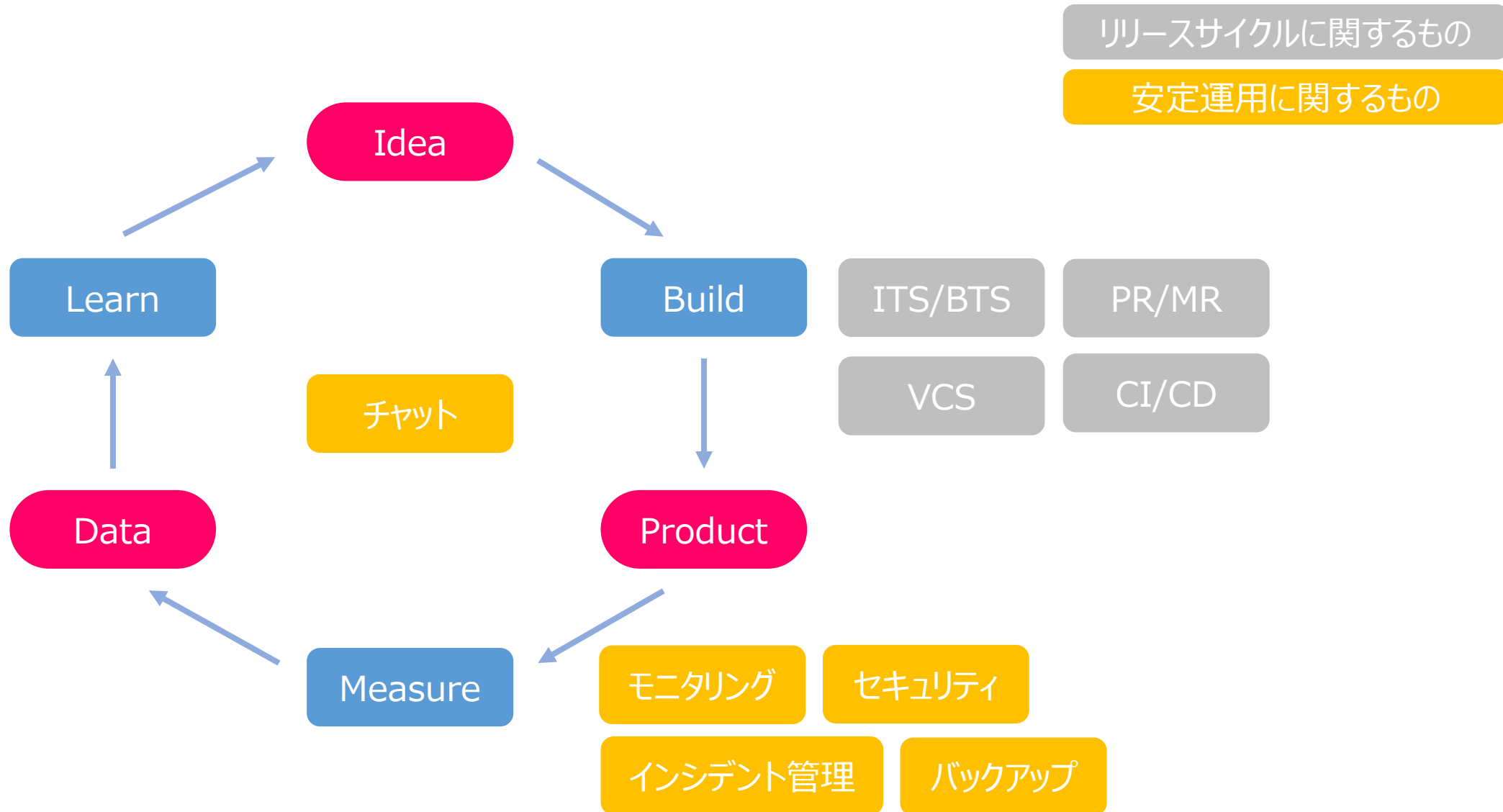


# DevOps活動をしやすい環境



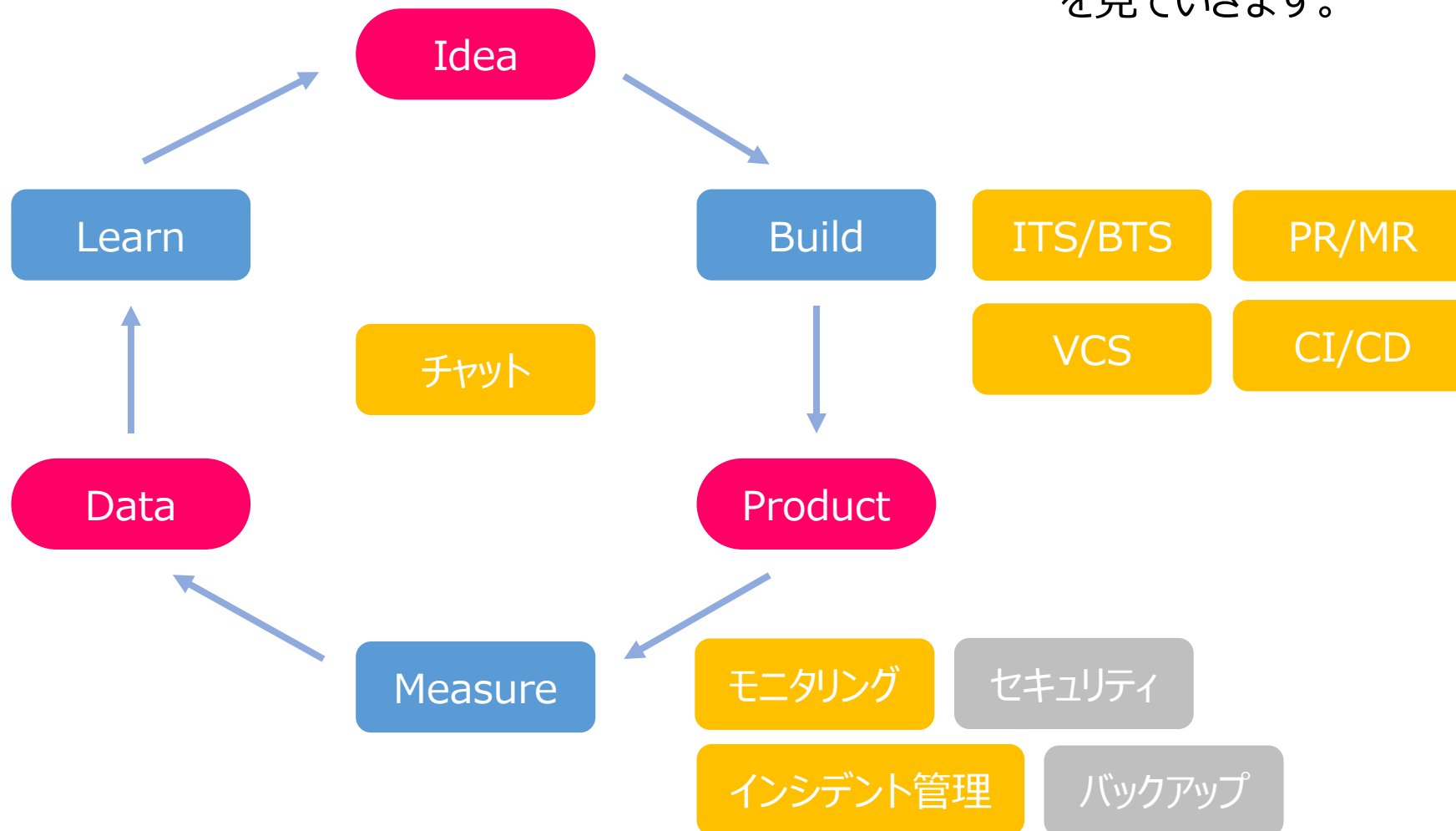


# DevOps活動をしやすい環境



# DevOps活動をしやすい環境

ハンズオンではセキュリティとバックアップ以外のところ  
を見ていきます。



質問タイム

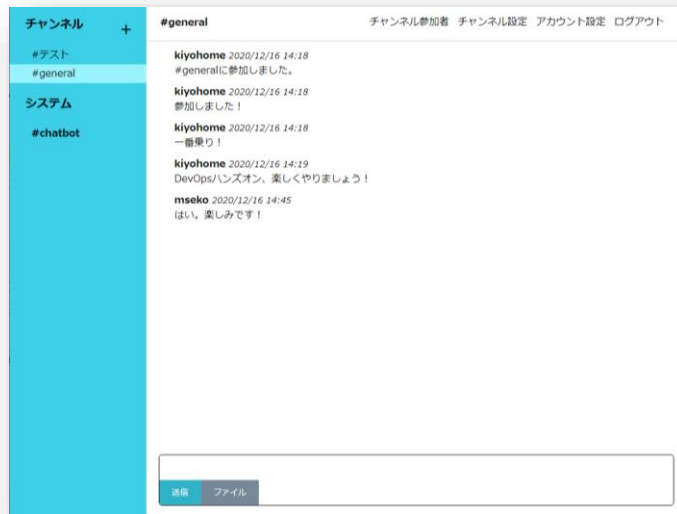
5分休憩

# ハンズオンの題材

# チャットサービス

チャットサービスを提供しているチームです。

チャットサービスはSPA + REST API構成のアプリケーションです。

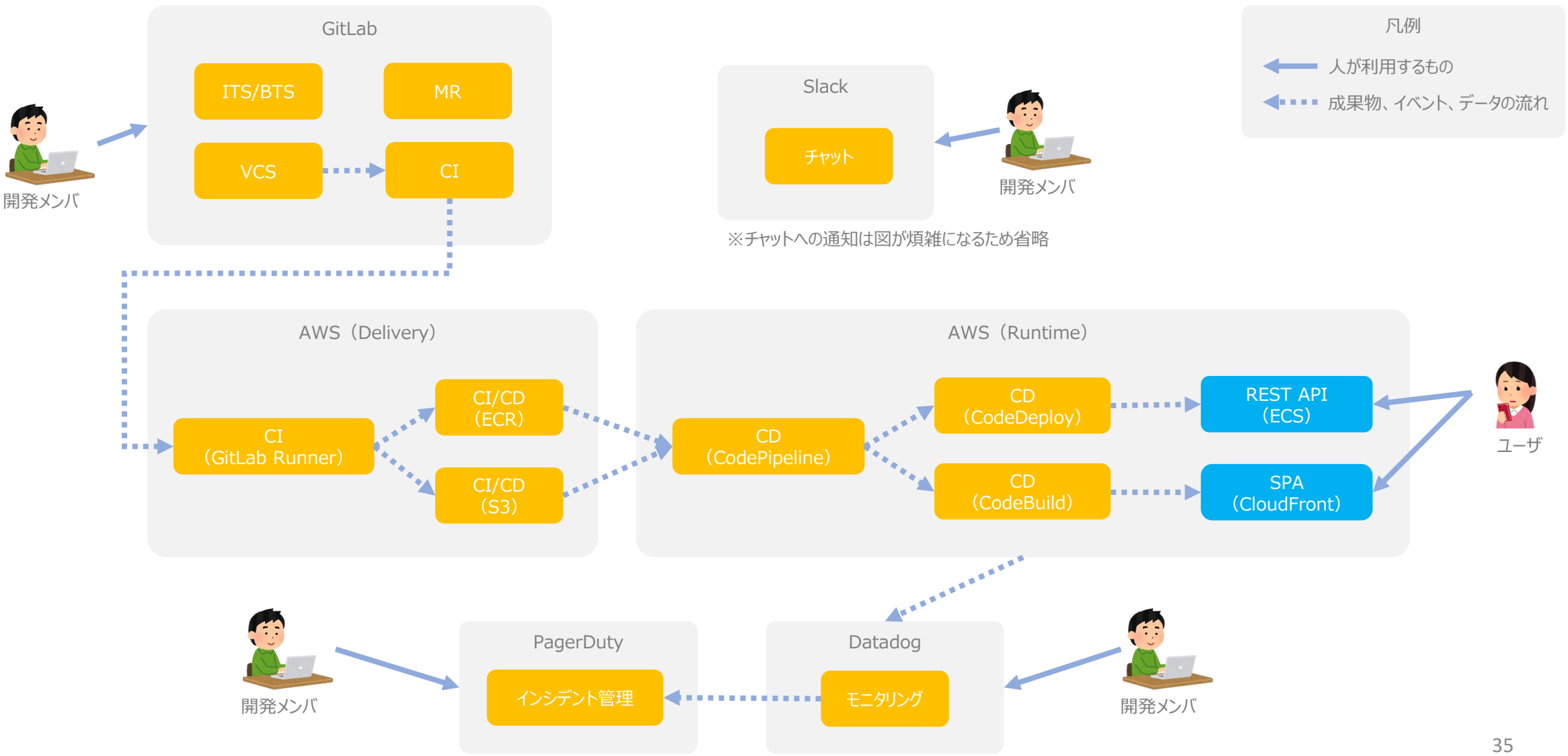


SPA + REST API構成のサービス開発リファレンス

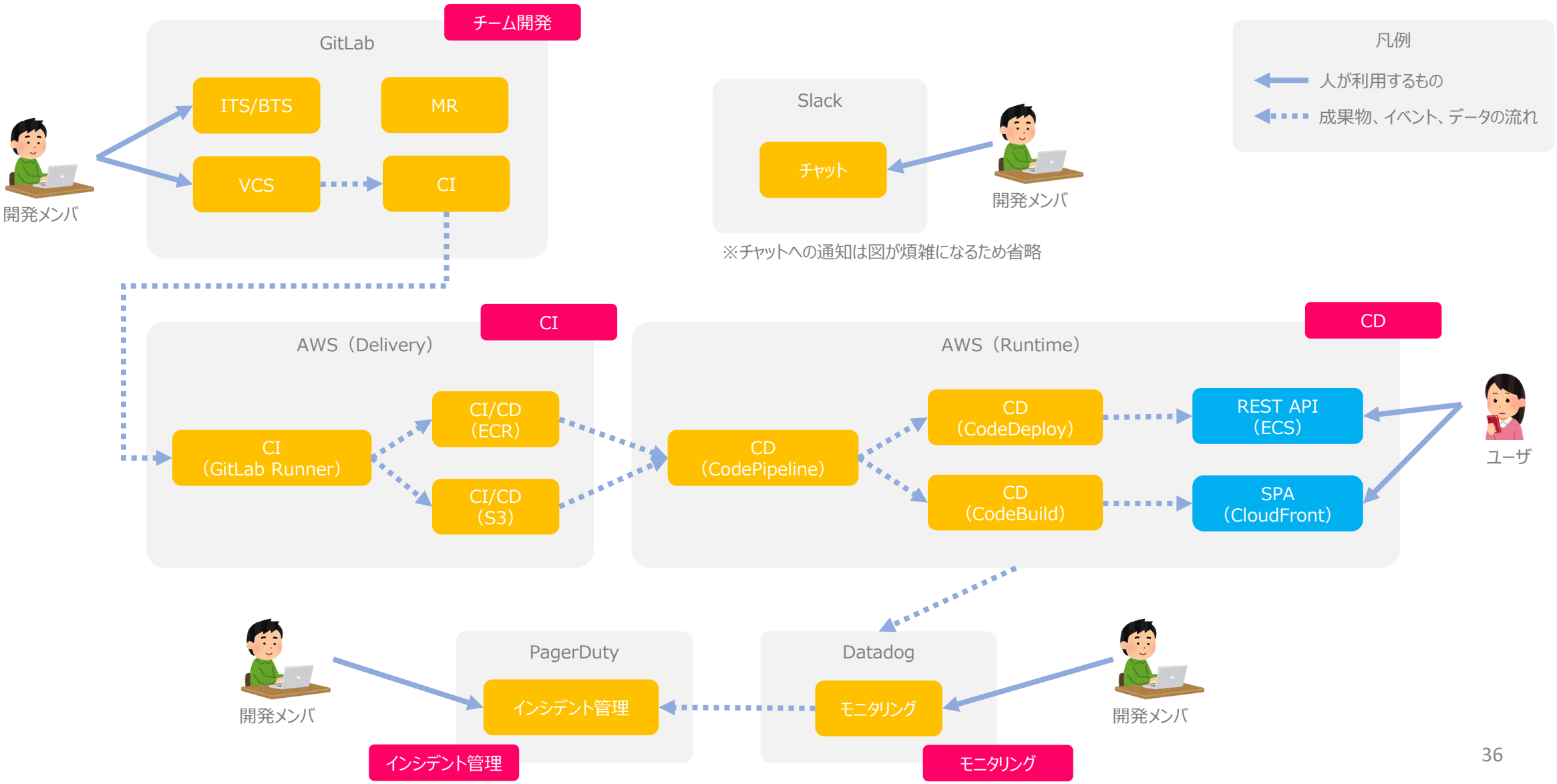
<https://fintan.jp/?p=5952>

→コード例にあるチャットサービスを使っています。

# チャットサービスのDevOps環境



# チャットサービスのDevOps環境





# チーム開発

# チーム開発

開発を加速するためには作業の可視化が大切です。

作業の可視化によく使われるのが作業をチケットとして管理する方法です。

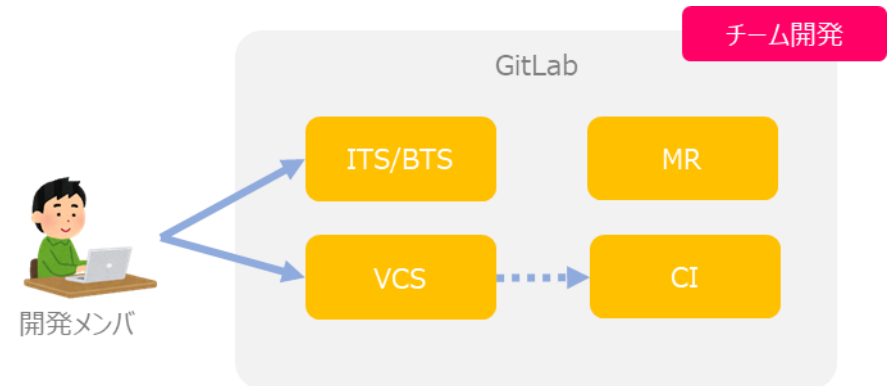
日々の作業をチーム全体で可視化し共有する方法として、

アジャイル開発では「かんばん」を用いることが多いです。

またアジャイルなプロジェクト運営の方法として「スクラム」が有名です。

チーム開発には[GitLab](#)を使用します。

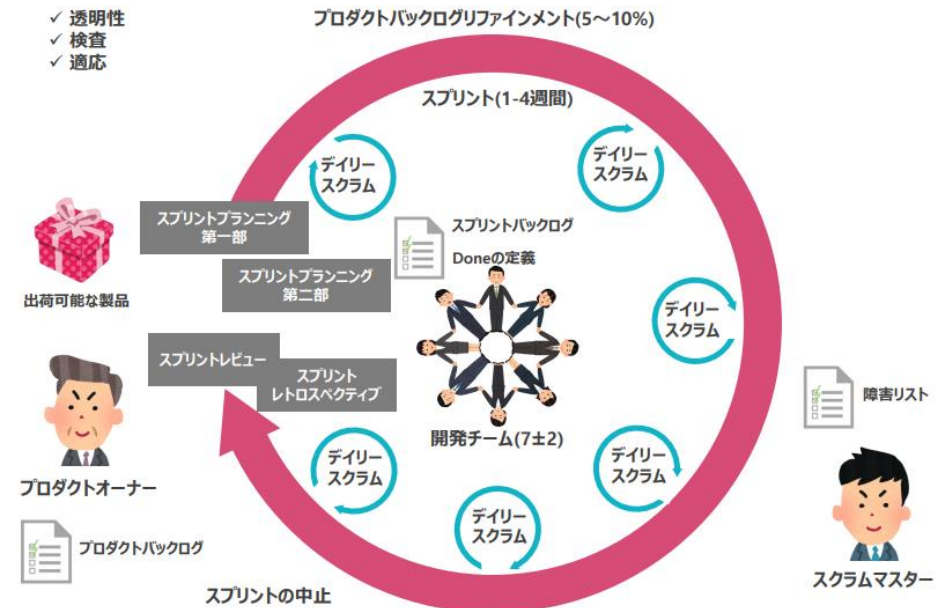
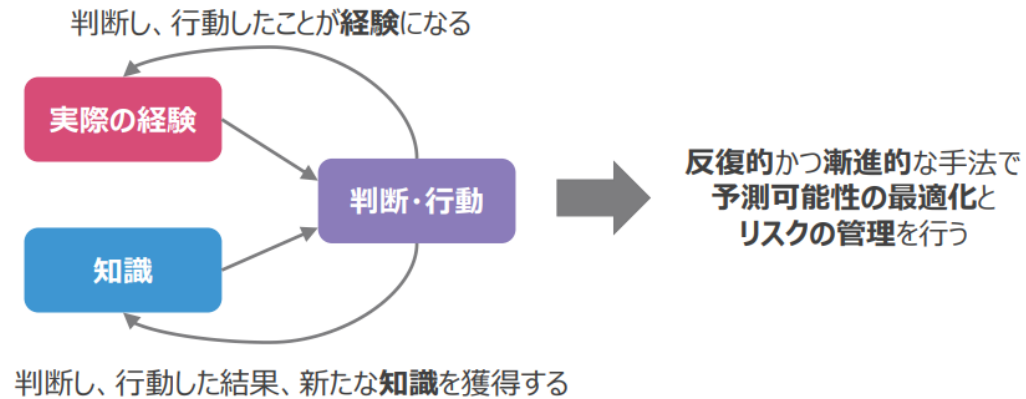
「スクラム」を実践する方法を見ていきます。



# スクラム

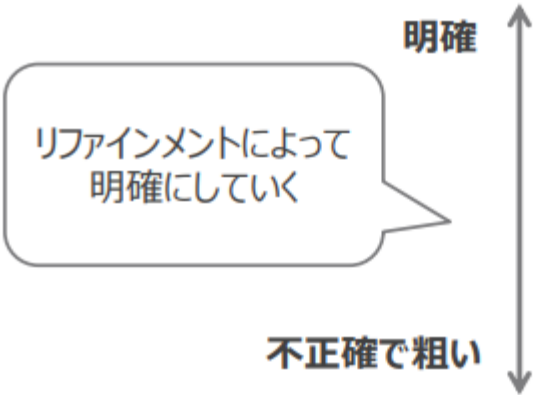
「複雑で変化の激しい問題に対応するためのフレームワークであり、  
可能な限り価値の高いプロダクトを生産的かつ創造的に届けるためのものである。」

スクラムは、経験的プロセス制御の理論(経験主義)を基本にしている。

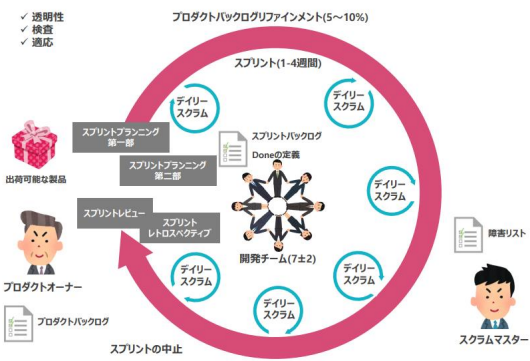


# スプリント全体計画

ストーリーを洗い出し、プロダクトバックログを作ります。  
マイルストーンを決めてスプリントにストーリーを割り当て  
おおよその全体計画を立てます。

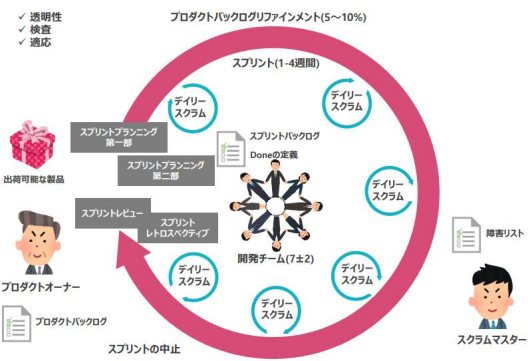


優先順位	ストーリー	見積
1	AとしてXXが出来る。	2
2	BとしてYYを一覧形式で参照できる。	3
3	C処理の性能改善	5
...	...	...
100	Dとしてレポートを作成できる	8



# スプリントプランニング

プロダクトバックログからスプリントで実施するストーリーを決定します。  
各ストーリーのタスクばらしを行い、タスクを決定します。



優先順位	ストーリー	見積
1	AとしてXXが出来る。	2
2	BとしてYYを一覧形式で参照できる。	3
3	C処理の性能改善	5
...	...	...
100	Dとしてレポートを作成できる	8

スプリントで実施するPBIを選択

ストーリー
AとしてXXが出来る。
BとしてYYを一覧形式で参照できる。

具体的なタスクに分解する




タスク	見積
UIのコーディング	3.0h
データモデル設計、変更、Entityの作成	3.0h
Actionのコーディング	2.0h
UIのコーディング	4.0h
Actionのコーディング	2.0h

# 日々の作業

デイリースクラムを開催し、スプリントのゴールを達成するために再計画します。

## デイリースクラム

 タイムボックス: 15分 / 1日

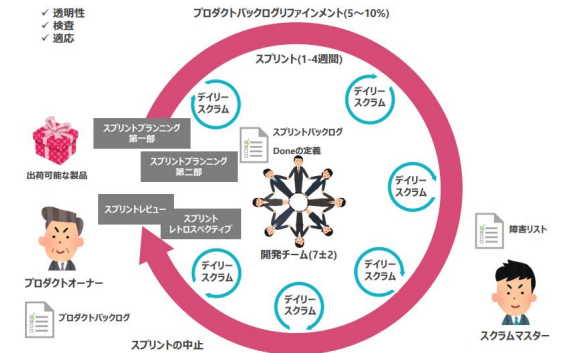
前回のデイリースクラムから行った作業の検査と、次回のデイリースクラムまでに行う作業の計画を立てる。

この場でスプリントバックログの作業進捗を検査する。  
デイリースクラムは毎回同じ時間、場所で開催する。

デイリースクラムでは、開発チームのメンバーが以下のことを説明する。

- ・ 開発チームがスプリントゴールを達成するために、私が昨日やったことは何か？
- ・ 開発チームがスプリントゴールを達成するために、私が今日やることは何か？
- ・ 私や開発チームがスプリントゴールを達成するときの障害物を目撃したか？

デイリースクラムで詳細な内容に踏み込みそうな場合は、デイリースクラム終了後に開発チームまたは一部のチームメンバーで集まり、詳細な議論、適応、再計画を行う。



# ITS/BTSとMRの使い分け

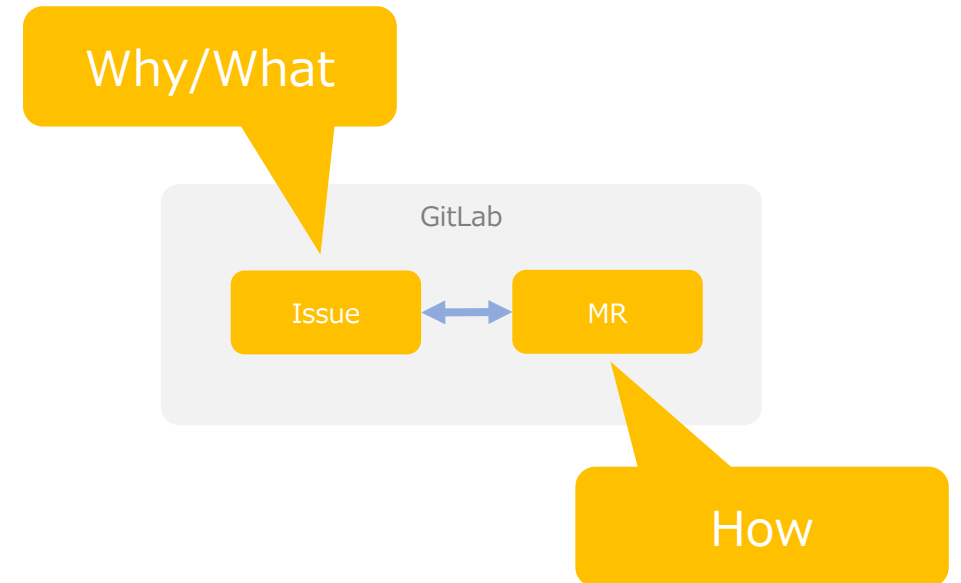
GitLabだとストーリーをIssueで表します。

レビューにはMRを使います。

ストーリー（Issue）には仕様や課題、不具合事象について  
なぜ？何を？を記録します。

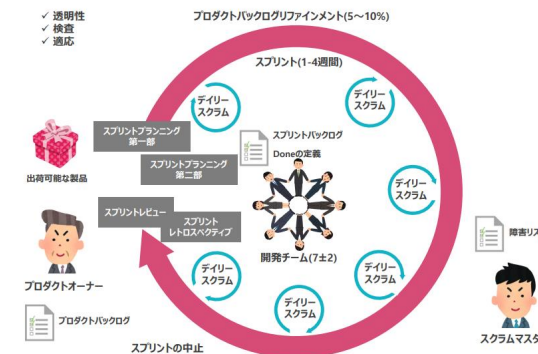
レビュー（MR）には実現方法やテスト方法について  
どのように作られたのか？対応したのか？を記録します。

IssueとMRをリンクできるので、どちらからでも後から経緯を確認できます。



# スプリントレビュー

スプリントレビューを開催し成果物を確認、マイルストーンに向け再計画します。



## スプリントレビュー



**タイムボックス:** 2時間 / 2週間

スプリントの終わりに出荷可能な製品の増分の検査と、必要であればプロダクトバックログの適応を行う。スプリントレビューでは、スクラムチームと関係者がスプリントの成果をレビューする。

進捗確認の場ではなく、フィードバックや更なる協力を引き出すことが目的。

スプリントレビューには以下が含まれる。

- ✓ 参加者はプロダクトオーナーが招待する
- ✓ **プロダクトオーナー**がPBIの完成したものと完成していないものについて説明する
- ✓ 開発チームはスプリントでうまくいったこと、直面した課題、それをどう解決したかを議論する
- ✓ 開発チームは完成したものをデモして、質問に答える
- ✓ プロダクトオーナーは現在のプロダクトバックログを確認し、完了日を予測する
- ✓ グループ全体で次に何をするか議論し、次のスプリントプランニングのインプットにする
- ✓ プロダクトの次のリリースに対するスケジュール、予算、性能、市場をレビューする





# 振り返り

DevOpsに通じる**改善活動の原動力**となるイベントです。  
何を改善していくかチームで認識合わせを行い、  
改善する内容を決めてストーリーとして登録します。

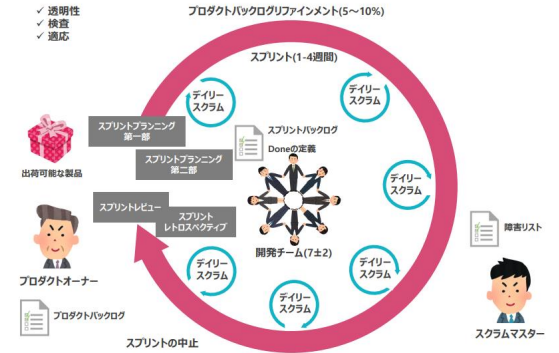
**KPT、YWT、FDL**など色々なやり方があるのでチームに合うものを探して実践します。

YWTという振り返り手法について:KPTとも比較してみた

<https://hisa-magazine.net/blog/manabutikara/ywt/>

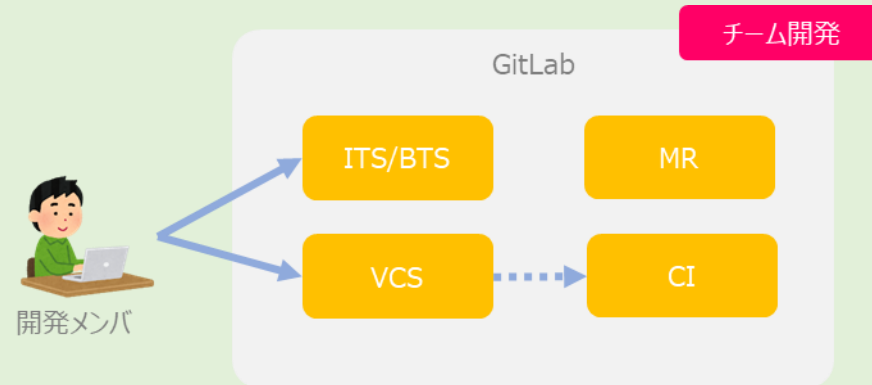
ファン・ダン・ラーン(FDL)ふりかえりボード

<https://qiita.com/yattom/items/90ac533d993d3a2d2d0f>



# タッチ＆トライ

実際に触ってみたい方、体験したい方、操作しましょう！



# スクラムのコンテンツ



<https://fintan.jp/>

FintanというTISが開発ノウハウを公開しているサイトで  
スクラムに関する次のコンテンツを公開していますのでご興味がある方はご覧ください。

[スクラム概論](#) ...スクラムを上司などに説明する資料。個人学習やチームメンバへの教育にも使えます。

[プロダクトオーナーの役割](#) ...あまり情報がないPOに関する資料。POの教育に適した資料です。

[スプリント開始条件チェックリスト](#) ...スクラムの停滞要因を未然に防ぐチェックリスト。始める際は必須です。

[ワーキングアグリーメント](#) ...チームメンバで作るルール。自走するチーム作りのプラクティスです。

[Doneの定義](#) ...リリース条件の定義。スムーズなリリースに向けたプラクティスです。

**CI/CD**

# CI/CD

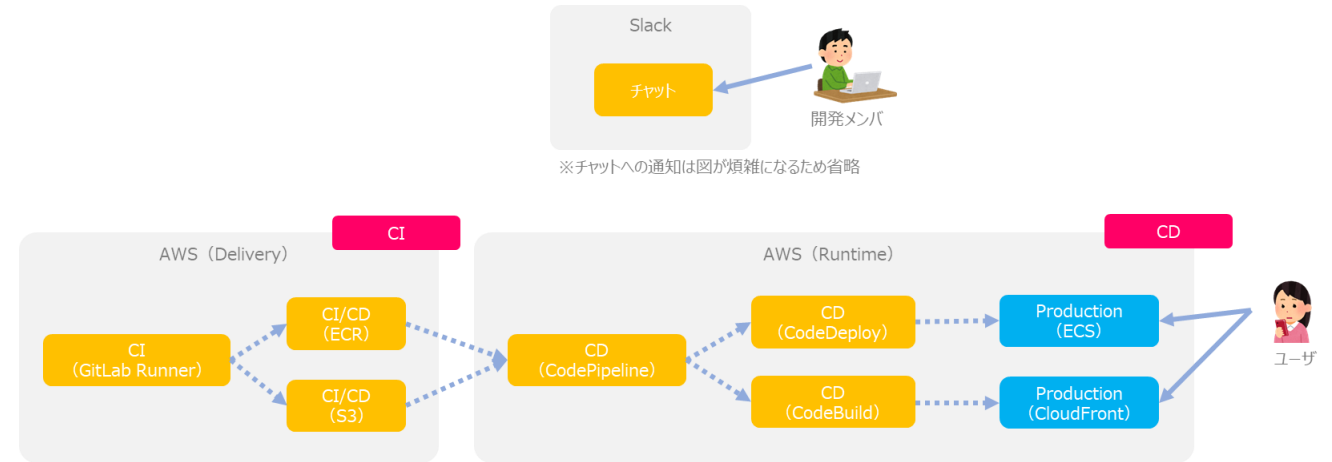
## 継続的インテグレーション

できるだけ早くビルド、テストを行い、  
早期に問題をフィードバック、対応を行い、  
常にテストが通る状態を目指す。

## 継続的デリバリー

できるだけ早くデプロイを行い、  
リリースに伴う問題をフィードバック、対応を行い、  
常にリリースできる状態を目指す。

どちらも問題が発生したら最優先事項としてすべての作業を止めて対応にあたります。



# テスト

テストはアプリが期待通りに動作するか？壊れてないか？をチェックします。

次のようなテストをできるだけ自動化して、早期に問題のフィードバックを得て、健康な状態を保ちます。

## ユニットテスト

ユニット単位（Javaであればクラス）のロジックをテストします。

## E2Eテスト

アプリを立ち上げ、アプリを利用するようにテストします。

全機能テストするとメンテナンスコストが高くなるのでゴールデンプアのみ実施するなど、

目的を満たす最小限のテストに留めます。

他にも、静的解析、セキュリティテスト、性能負荷テストなど、

非機能観点のテストも自動化する場合があります。

# ブランチ

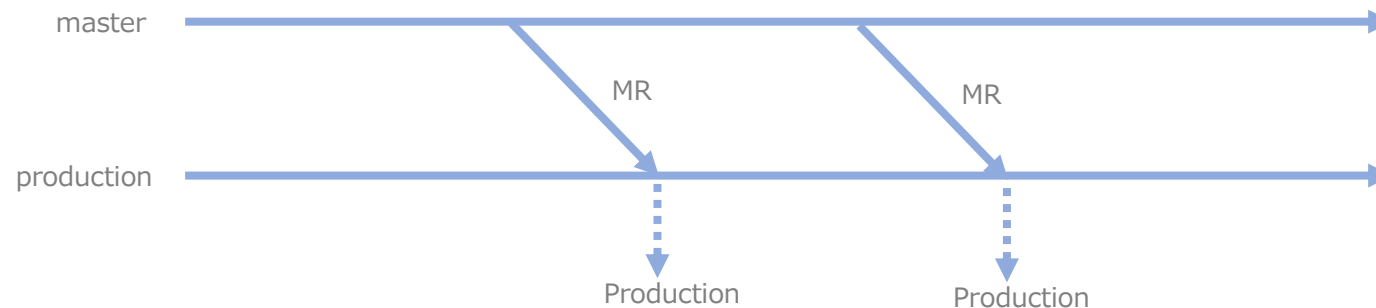
リリースするにあたって**ブランチをどう運用するか**を考える必要があります。

Git-flow GitHub-flow GitLab-flowという開発フローについてまとめる

<https://qiita.com/pandama09396862/items/9f013fa7b60f4d12d1d8>

チャットサービスでは**GitLab Flow**をベースに次のようにしています。

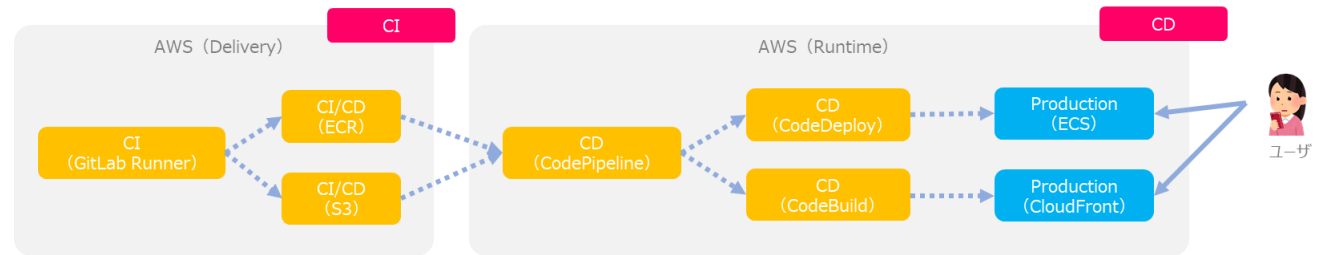
- 開発はmasterから新たにブランチを作成し、masterにMRする。
- リリース時はmasterからproductionにMRする →本番環境へデプロイ



# コンテナ

Dockerを使うことでアプリケーションのコンテナ化ができます。コンテナ化されたアプリはDockerがインストールorサポートされている環境であれば同じように動作します。コンテナ化 = コンテナイメージと呼びレジストリに登録し、いつでも取り出して動かすことができます。

チャットサービスのパイプラインでは、はじめにアプリのコンテナイメージを作り、テストしたイメージを本番環境にデプロイしています。テストと本番環境に同じコンテナイメージを使うので「デプロイしたら動かない」問題が発生しずらいです。



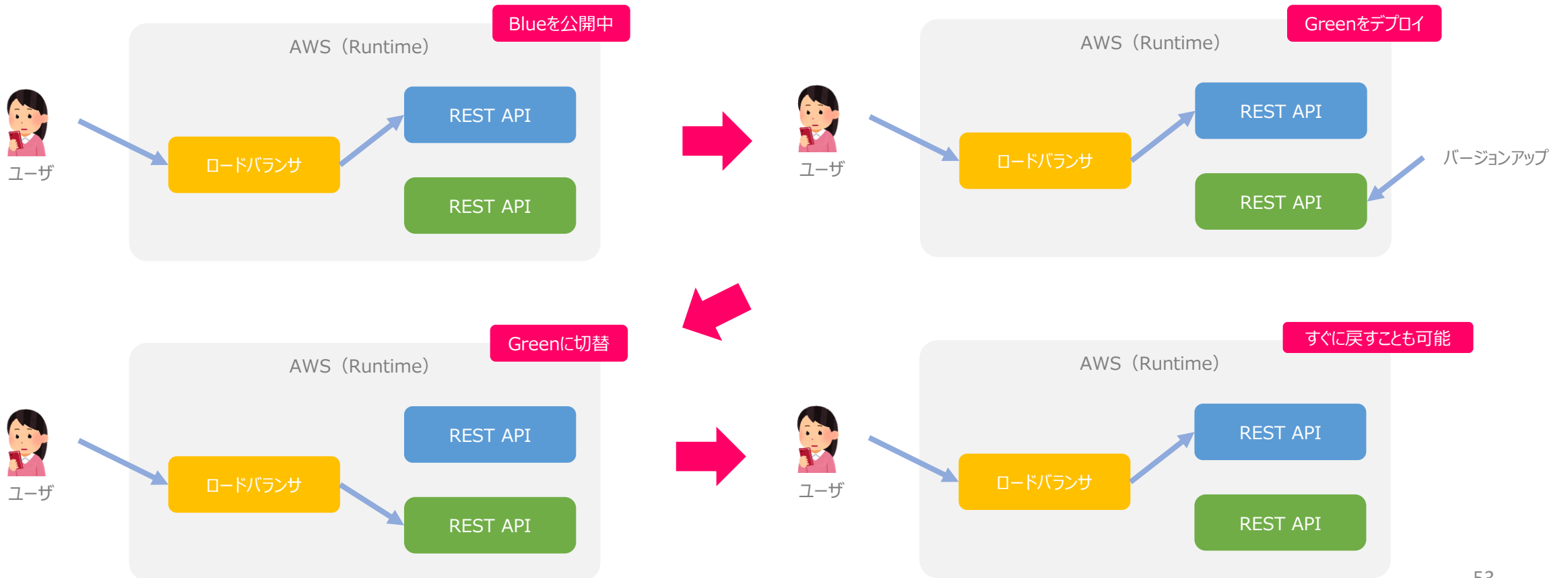
クラウドのコンテナサービスを使用することで、コンテナ化されたアプリのスケールアウトやリソース割り当てを柔軟に管理できます。



# Blue-Greenデプロイメント

「Blue-Green Deployment」とは何か、マーチン・ファウラー氏の解説

[https://www.publickey1.jp/blog/14/blue-green\\_deployment.html](https://www.publickey1.jp/blog/14/blue-green_deployment.html)



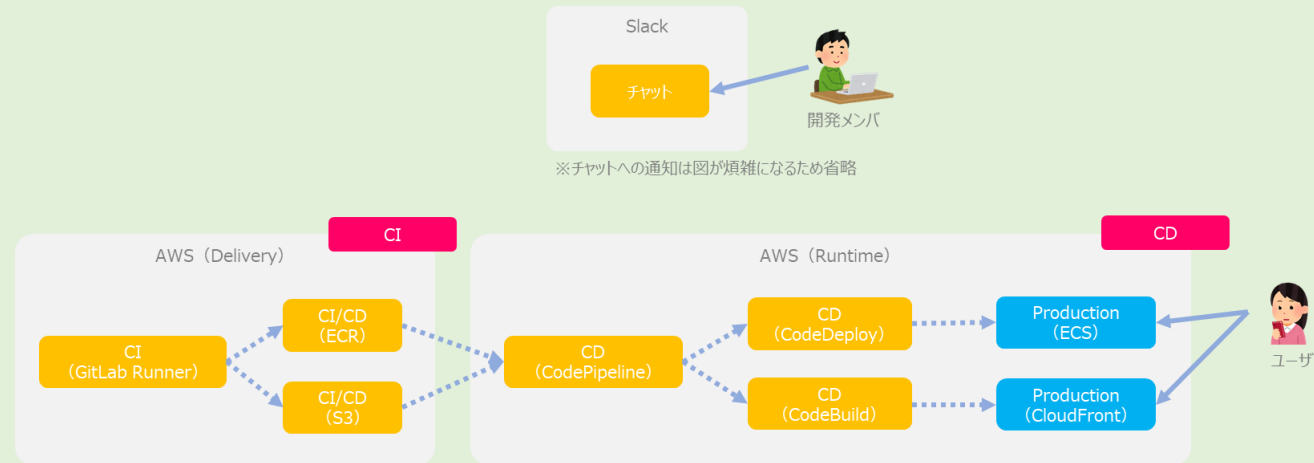
# 管理者によるリリース承認

1日10回デプロイを目指したいのですが、**組織文化や責任の所在**を明らかにするため、管理者が承認しないとリリースできない仕組みを取り入れる場合があります。



# タッチ&トライ

実際に触ってみたい方、体験したい方、操作しましょう！



5分休憩

# モニタリング

# モニタリング



サービスの**安定運用**を実現するためにサービスの稼働状況をモニタリングします。  
モニタリングには[Datadog](#)を使用します。Datadogにより次のことを実施できます。

## 障害検知、予兆検知

アプリやミドルのログに含まれる文字列やリソースの数値（CPU等）を使って障害や予兆を検知できます。  
検知した場合は次に説明するインシデント管理に通知します。

## ダッシュボード

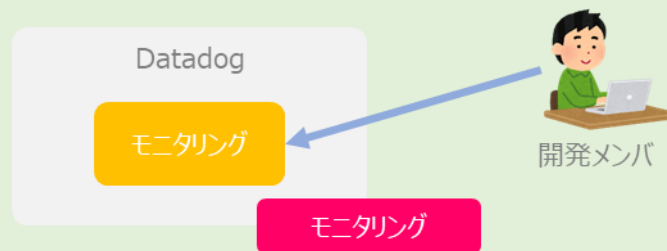
モニタリングしている状況を可視化します。単純に見れるだけでなく、ある時間帯の複数のモニタリング対象の状況を一目で見る、といった問題箇所の特定に役立つような絞り込みができます。

## ログ検索

複数のログを集約し、横断的にログの内容を検索できます。

# タッチ＆トライ

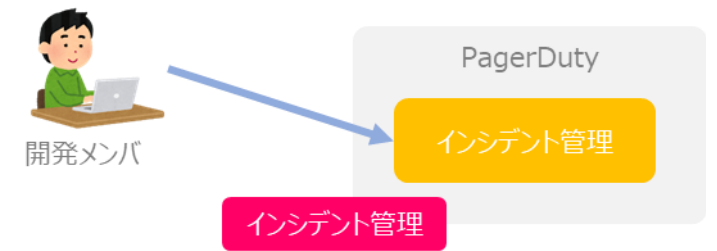
実際に触ってみたい方、体験したい方、操作しましょう！



# インシデント管理



# インシデント管理



モニタリングによりサービスの稼働状況が可視化され、障害や予兆を検知できる状態になります。  
サービスが成長し利用者と機能数が増えることで、モニタリングにより検知されるアラートの数は増加します。  
増加するアラートを効率的かつ確実に対応されるようにインシデント管理を行います。  
インシデント管理にはPagerDutyを使用します。PagerDutyにより次のことを実施できます。

## アラートの通報

連絡網のようなものです。通報ルールを設定し、ルールに基づき連絡がつくまで通報していきます。  
確実に誰かにアラートが通報されます。

## オンコールのスケジューリング

当番表のようなものです。メンバーのスケジューリングができ問題発生時に対応者不在といった状況を未然に防ぐことができます。

# タッチ&トライ

実際に触ってみたい方、体験したい方、操作しましょう！



# Epona

# DevOps環境構築キット(Epona)とは

## サービス用にDevOps用環境をオンデマンドで構築できるツールキット

- サービスを運用できる本番環境
- 本番環境と同様の構成を持つテスト環境/ステージング環境
- 迅速な開発が可能な開発環境

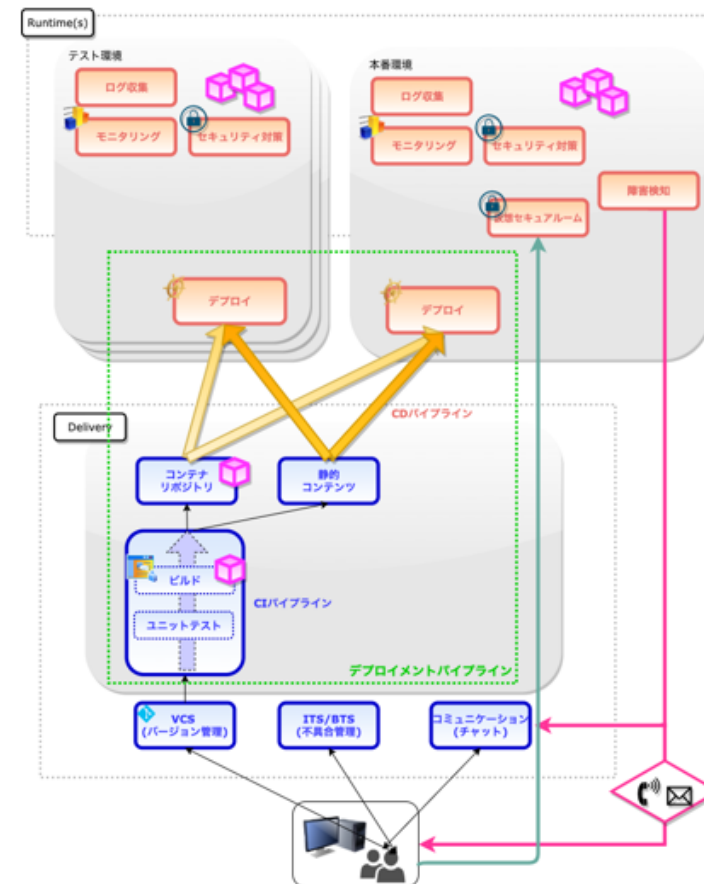


ツールキットの提供する「モジュール」を  
組み合わせることで、様々なサービス用環境を実現できます

### ターゲットとするクラウドサービス

■ 2020年度

- Amazon Web Services (AWS)
- Microsoft Azure (Azure)



# 使い方

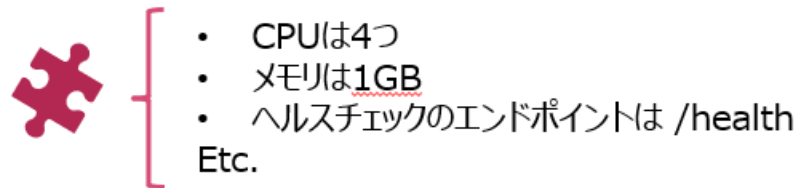
## 1. ユースケースに応じ、使いたいモジュールを選ぶ

- Eponaはユースケース毎のモジュール集です



## 2. モジュールの入力パラメータを設定する

- ガイド上の入力パラメータをご参照ください



## 3. 適用する

```
$ terraform apply  
(snip)
```

**apply complete! Resources: 2 added, 0 changed, 0 destroyed.**



### ガイド

入出力ファレンス

#### Requirements

Name	Version
terraform	>= 0.13.5
aws	>= 3.13.0

#### Inputs

Name	Description	Type	Default
container_cluster_name	コンテナサービスのクラスター名	string	n/a
container_definitions	コンテナサービスのタスクで実行する、コンテナ定義のリスト	string	n/a
container_health_check_path	ロードバランサーの、コンテナに対するヘルスチェックのパス	string	n/a
container_port	ロードバランサーがコンテナに転送するポート	number	n/a

# Eponaによって何が解決できるのか

😞 サービスを動かせる環境の調達や構築に時間がかかる

😞 どのクラウドリソースを使い組み合わせたらよいのかわからない

😞 サービスの調査や環境の立ち上げに時間がかかる

- Eponaの提供するモジュールを適用すれば、「データベース」、「CDパイプライン」、「ログ収集」といったユースケースを実現できます。

提供する  
モジュール群  
(一部)



ネットワーク

VPC

データベース

RDS

コンテナランタイム

ALB+ECS

デプロイメントパイプライン

CodePipeline+CodeDeploy  
CodePipeline+CodeBuild

フロントエンドランタイム

CloudFront+S3

Webアプリケーションファイアウォール

WAF

ログ収集

Datadog+Lambda

仮想セキュアルーム

Workspaces

ユーザ・権限

IAM

# クロージング

# DevOpsハンズオンはいかがでしたか？

DevOpsの作り方を体験いただけましたでしょうか？

今日紹介したEponaというDevOps環境構築ツールキットを  
1月末にFintanで公開予定です。

どなたでもご利用頂けますのでぜひご活用ください。

<https://fintan.jp/>

他にも現場で活用しているコンテンツやノウハウ、  
技術ネタのブログも公開していますのでぜひ覗いてみてください。  
友人に紹介いただいたり、勉強会等でご活用ください。





# サービス開発エンジニア体験

サービス/プロダクトの開発に欠かせない  
アプリ開発とDevOpsを体験してみませんか？

9月 SPAハンズオン

10月 APIハンズオン

11月 モバイルハンズオン

12月 DevOpsハンズオン

本日が最終回です。

ご参加くださった皆さんありがとうございます！

# Aizurage

connpassのグループです。

<https://tidev-aizu.connpass.com/>

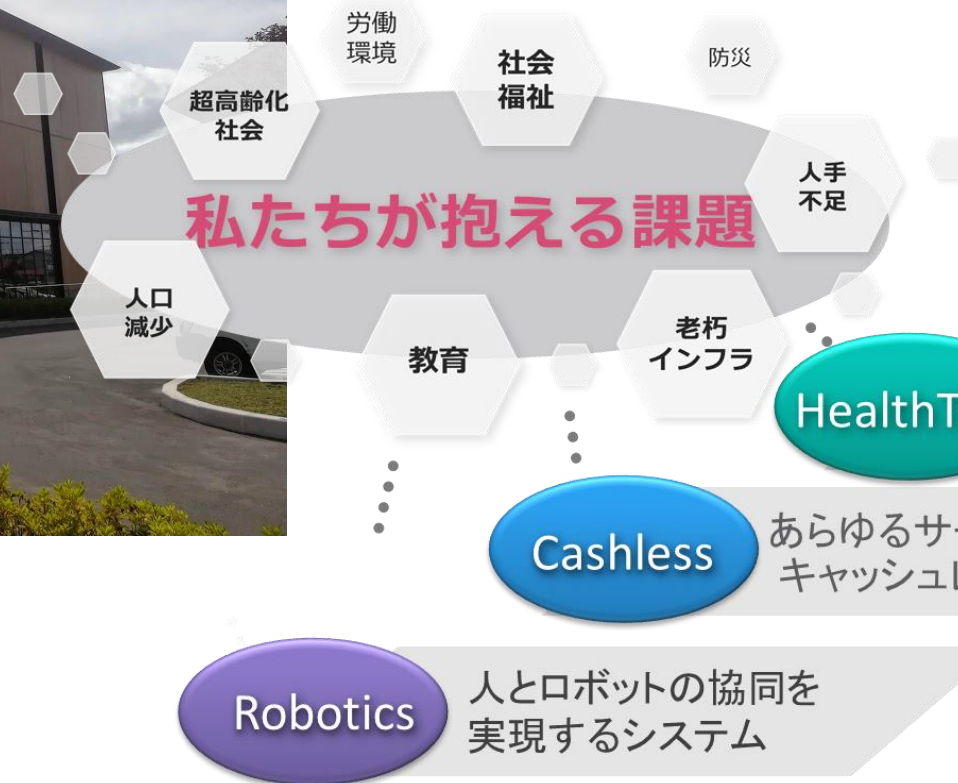
TISの会津拠点のエンジニアが中心になって、  
エンジニア交流を目的にハンズオンや勉強会をやっています。

興味がありましたらグループのメンバーになってください。  
メンバー＝グループのスタッフではないので安心してください。  
グループのメンバーはTwitterのフォロワーのようなイメージです。

「メンバーになると、グループのイベントが作成されると通知がきたり、  
トップページのおすすめイベントに表示されるので、  
興味のあるイベントを見逃すことが少なくなります。」



# TISの会津での取り組み



対処療法から  
予防医療へ

HealthTech

あらゆるサービスを  
キャッシュレス化

Cashless

人とロボットの協同を  
実現するシステム

Robotics

# We're Hiring!

キャリア採用のエントリーページ

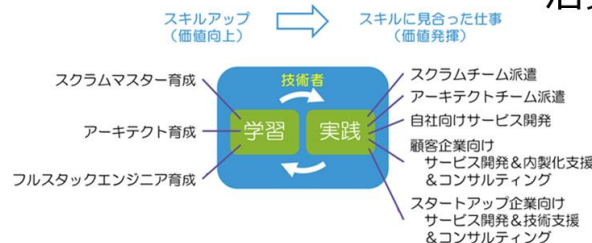
<https://hrmos.co/pages/tissaiyo/jobs/20100400011>

技術力で活躍したいエンジニアを募集しています！  
まずは東京、大阪で経験を積んで、そのままでいいし、  
U/Iターンで会津若松でもいいし、一緒に働きませんか？

テクノロジー＆イノベーション本部は、以下のような**多方面にわたるプロフェッショナルを有した組織**です。

- P J 実践の場で活躍するエンジニア  
( I T アーキテクト、サービス開発エンジニア)
- 数年後に獲得すべき技術テーマについて研究する研究者
- 新規事業の事業化を目指す事業推進者

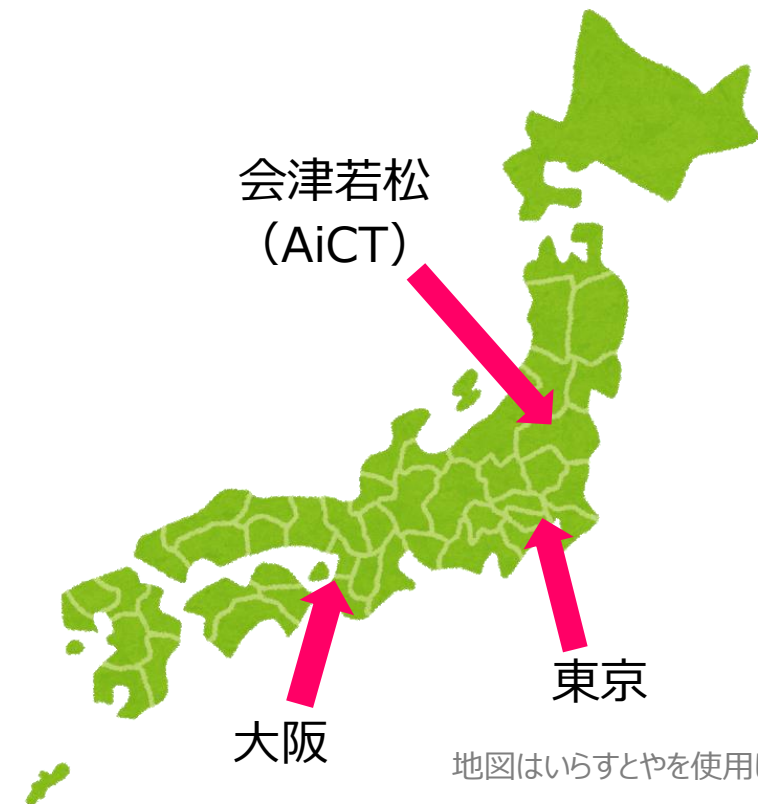
技術者の**育成**に力を入れていて、  
活発で**オープン**な議論ができる風土。



組織内外で各種の育成施策を実施。

- TechBall (技術勉強会)
- アーキテクチャ研修
- 新人研修

組織内コミュニケーションはSlackでオープンに。  
技術系イベント登壇や記事投稿する社員も多数。



地図はいらすとやを使用しています。

<https://www.irasutoya.com/>

# アンケート

今後の改善に活用したいのでアンケートへのご協力をお願いします。

アンケートのURLはZoomのチャットで連絡します。





**Thank you**  
Service Dev Engineer TAIKEN