ノリで造る！
ネットワーク運用基盤開発

合同会社DMM.com
ITインフラ本部 インフラ部ネットワークグループ

NREチーム 大橋幸輝 / 佐々木航平

v1.2

Agenda

- 自己紹介
- 開発した背景
- システム構成と検討構成
- 開発時の課題
- 単体機能の紹介
- Dockerによるマイクロリリース
- EKS移行と苦労
- 今後の拡張と展開
- まとめ
- 討論等

自己紹介

名前(年齢): 佐々木 航平(29)
趣味:

- 珍奇植物
- コーヒー
- 筋トレ
- 競馬
- ミニ四駆

経歴:

- 2016/5 ~ 2017/6 ITベンチャー(SES事業)
- 2017/7 ~ 2020/3 DC事業者(SES事業)
- 2020/4 ~ 合同会社DMM.com(コンテンツ)

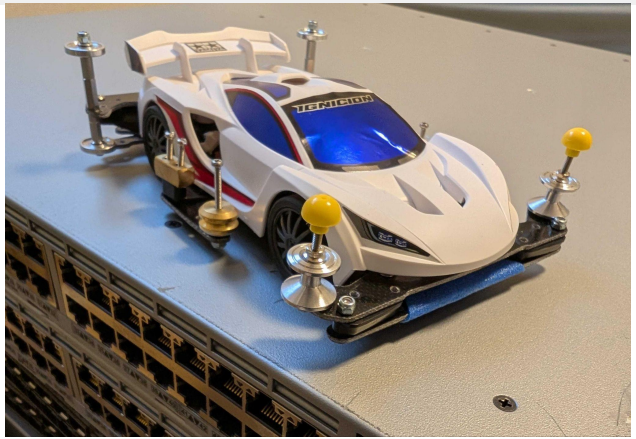
業務範囲:

- NW設計/検証/構築/運用(バックボーン~拠点)
- ツール開発(Ansible/Python/コンテナ)

自己紹介

名前(年齢): 大橋 幸輝(24)
趣味(?):

- 自宅でのサーバ運用とアプリ・システム開発
- 電気工事
- アマチュア無線
- すみっコぐらしグッズ集め
- ミニ四駆



経歴:

- 2022/8 ~　23新卒としてDMM.comに入社(アルバイト期間を含む)

業務範囲:

- NW設計/検証/構築/運用(バックボーン~拠点)
- ツール開発/基盤運用(Ansible/Python/コンテナ/k8s)

開発した背景

ネットワークの日々の運用、どれくらい自動化していますか？

- 定常的な設定作業
  - Vlan追加、経路追加、Peering追加...etc
- 非定常作業
  - ワンオフ構成の収容、イレギュラー対応...etc
- **アラートに対する対応**
  - **コマンドによる状態確認、交換対応　...etc**

今回の話はここに対するアプローチのお話

開発した背景

日々の運用にて

- ネットワーク機器のアラート対応が多い
  - アラートの度に機器にログインしては似たようなコマンドを叩く
- 都度都度VPNはってSSHして…がめんどくさい

→対応自動にしたいよね

ただし

- 社内のZabbixはアラート利用で運用は他部署のため
  Scriptを仕込むにも調整が必要
- 逆に自前でZabbix立てるにもトリガーアクション以外の機能は不要

開発した背景

それならいっそ…

全部造るか！

いけるっしょ、多分

ただし、造る以上は真面目に作り上げる

作成したシステム

ノリで始まった中で作り上げたのが
FANNEL というシステム
FAst： 早い

Network： ネットワーク

Notification： 通知

Engage： 従事

Logging： ログ

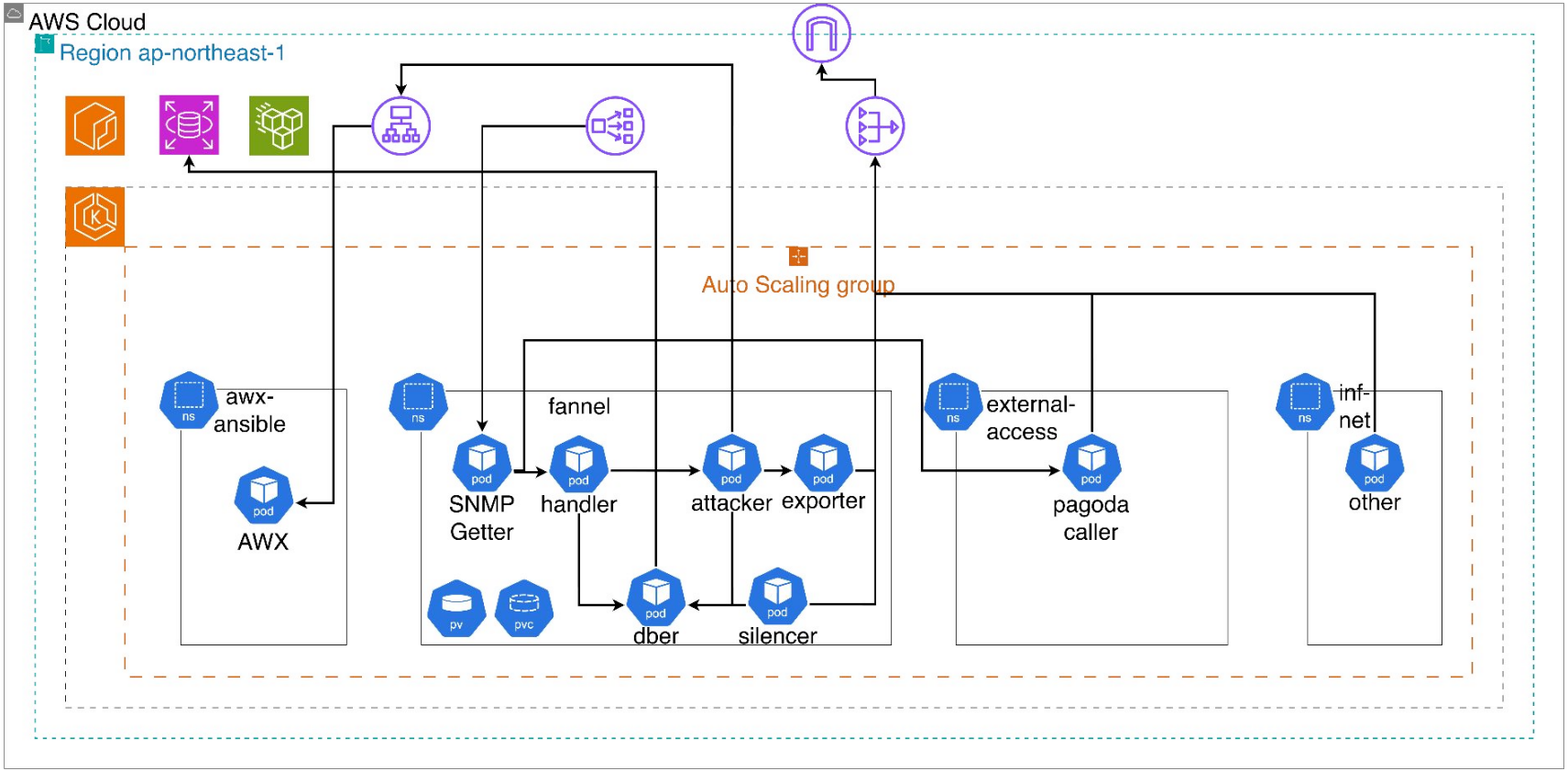
意訳すると、迅速なネットワークアラートの通知とロギング

FANNELとは

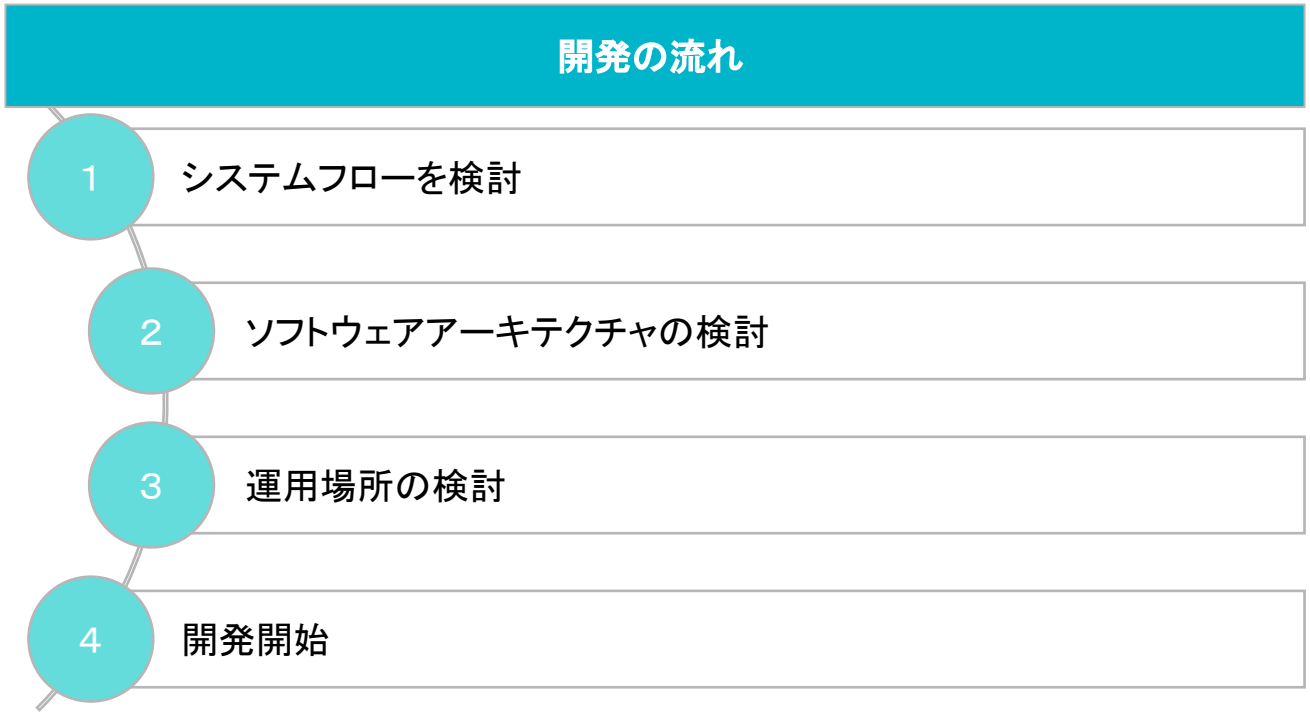
どのようなシステムか？

- SNMP trapやSyslogを受信し、それに対応した運用処理を実行する
  - 現状はtrapのみ、Syslogは今後対応予定
- HostのデータをSSoTツールから参照するため
  Hostデータの二重管理をしない
  - 機材管理台帳のようなものと監視の設定で2重管理になりがちなところを解消している、という意味合い
- 機器構築時に宛先に追加するだけで利用OK

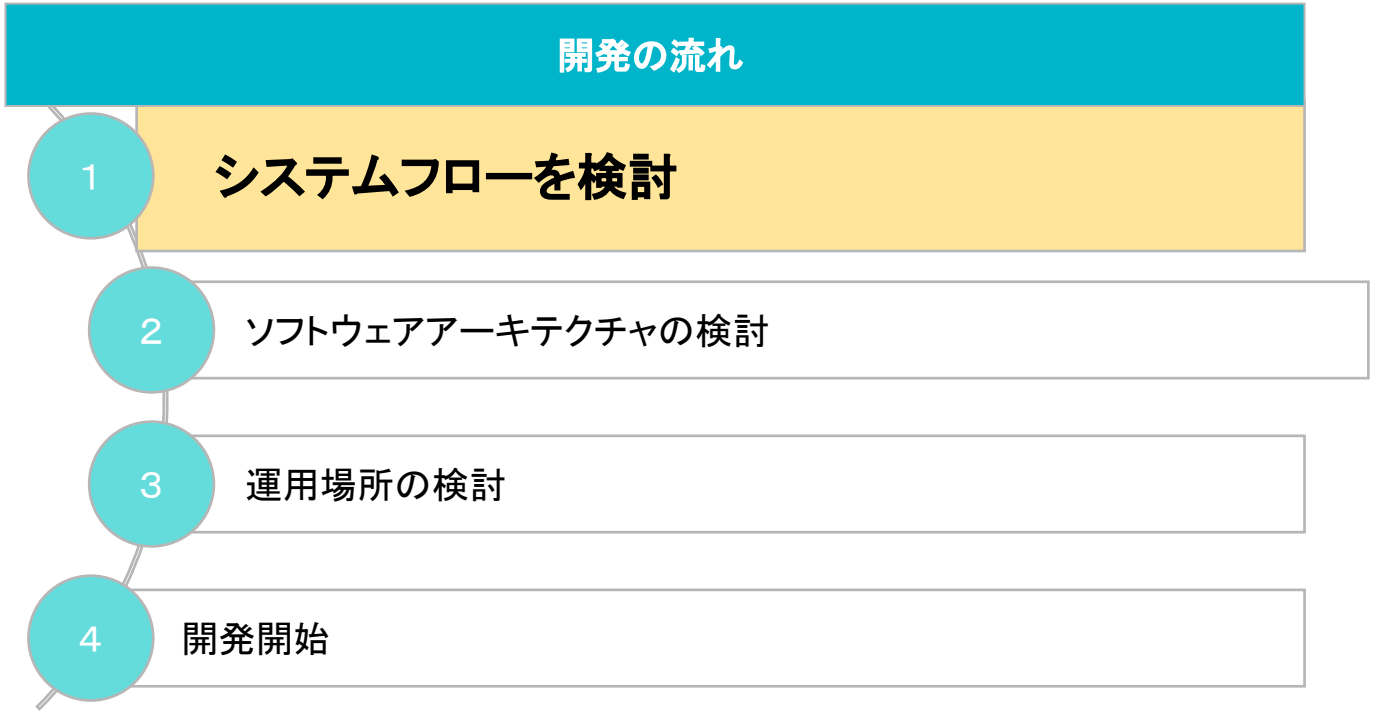
FANNELとは - システム構成図



システム構成と検討構成

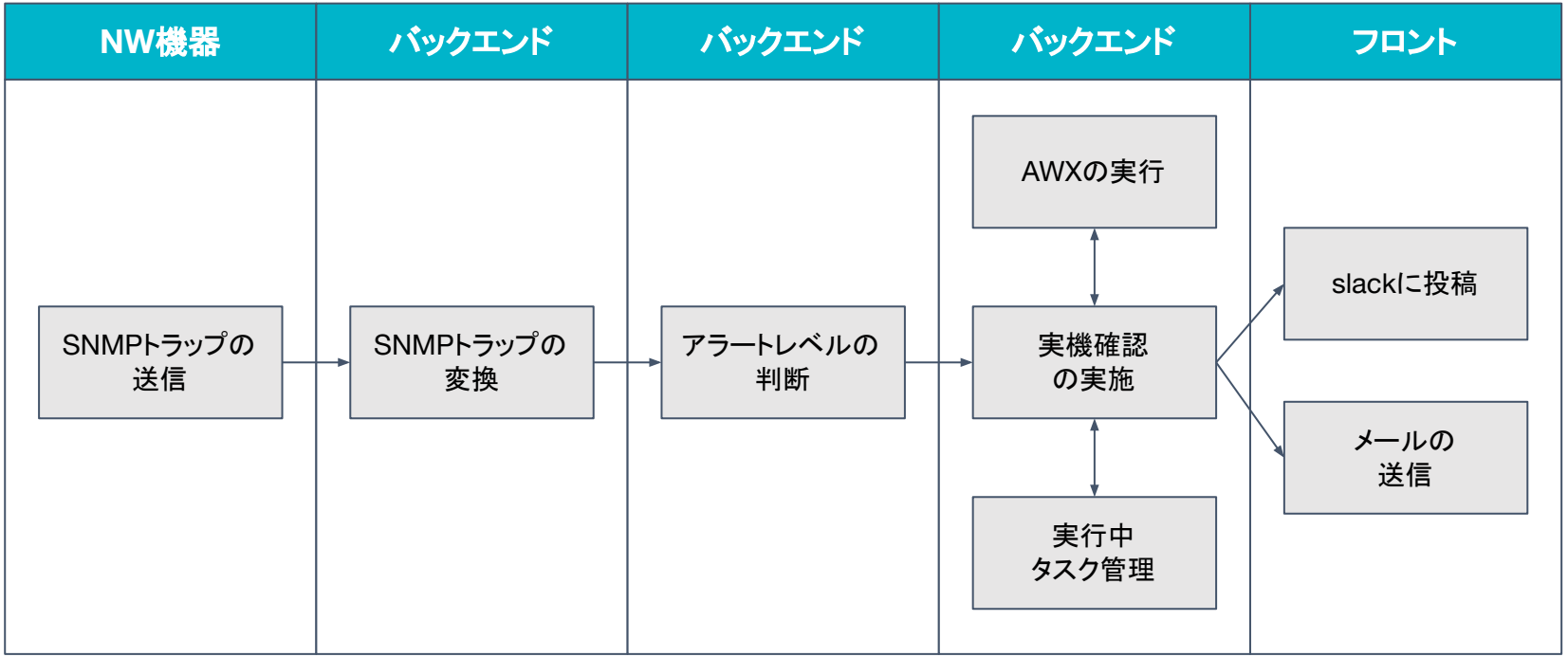


システム構成と検討構成

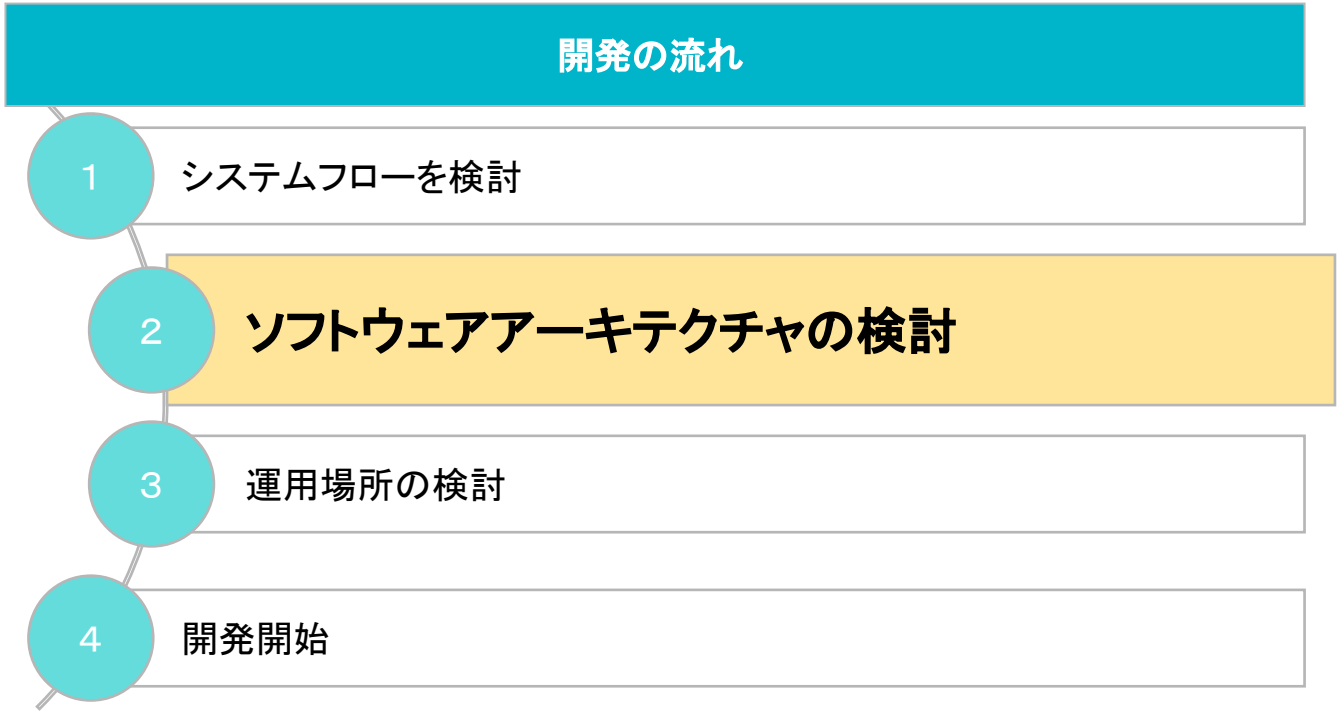


システム構成と検討構成

処理フローと必要な機能



システム構成と検討構成



システム構成と検討構成

モノリシックアーキテクチャ

すべての機能を一つにまとめたアプリケーションで構築されたシステム

メリット	デメリット
● 簡単に開発とデプロイが可能	● 拡張性の低い
● 高パフォーマンス	● メンテナンス性が低い
● インフラがほぼ不要	● スケールが難しい
	● 同時開発が難しい

システム構成と検討構成

モノリシックアーキテクチャ

すべての機能を一つにまとめたアプリケーションで構築されたシステム

従来型のアーキテクチャで
チームでは今までこのアーキテクチャで開
発を行っていた。

各機能はメモリ内でやりとりを行う。

システム構成と検討構成

マイクロサービスアーキテクチャ

サービスや機能などで分割し独立させ、組み合わせて動作させるシステム

メリット	デメリット
● 独立しているので個別に技術選定できる	● 考慮事項が増え、複雑化する
● 機能ごとにスケールできる	● 管理するリソースが増加
● 独立してデプロイが可能	● 分散トランザクションの考慮
● 変更箇所を最小限にできる	● 通信遅延
● **担当範囲を狭めることができる**	

システム構成と検討構成

マイクロサービスアーキテクチャ

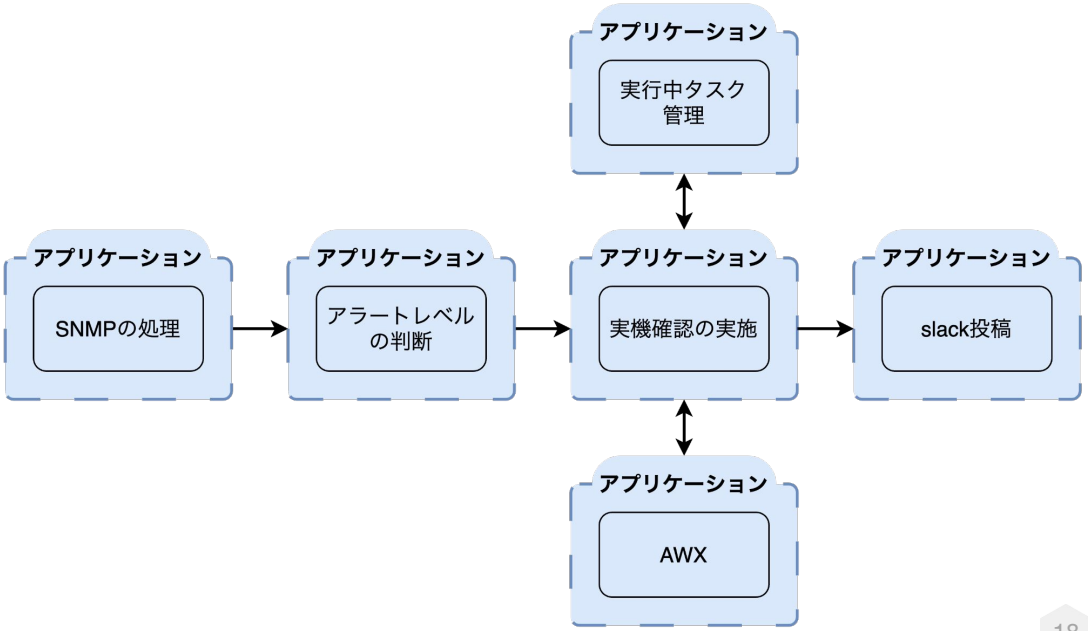
サービスや機能などで分割し独立させ、組み合わせて動作させるシステム

採用したアーキテクチャ

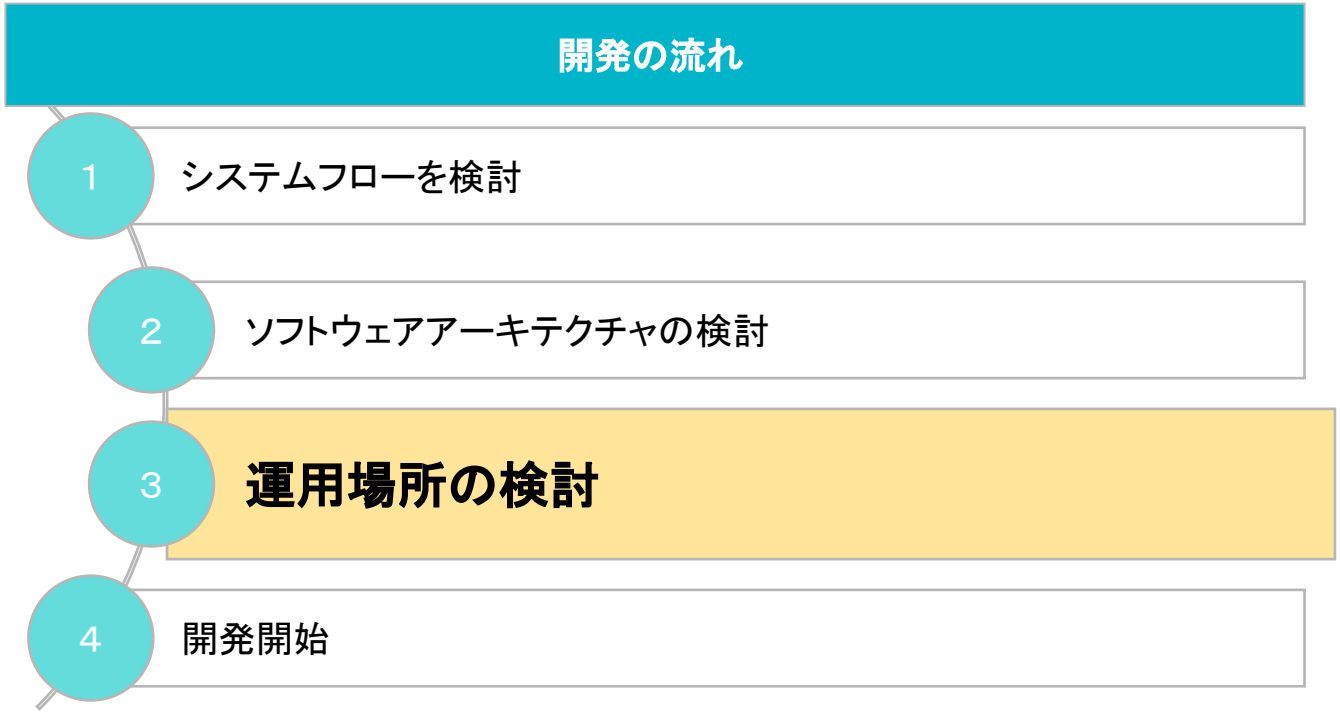
アプリケーション同士の通信は
APIなどを利用

- 技術的にも挑戦したい
- 新卒研修でも勉強したし…
- メリットが大きそう

ということで半分ノリで採用。



システム構成と検討構成



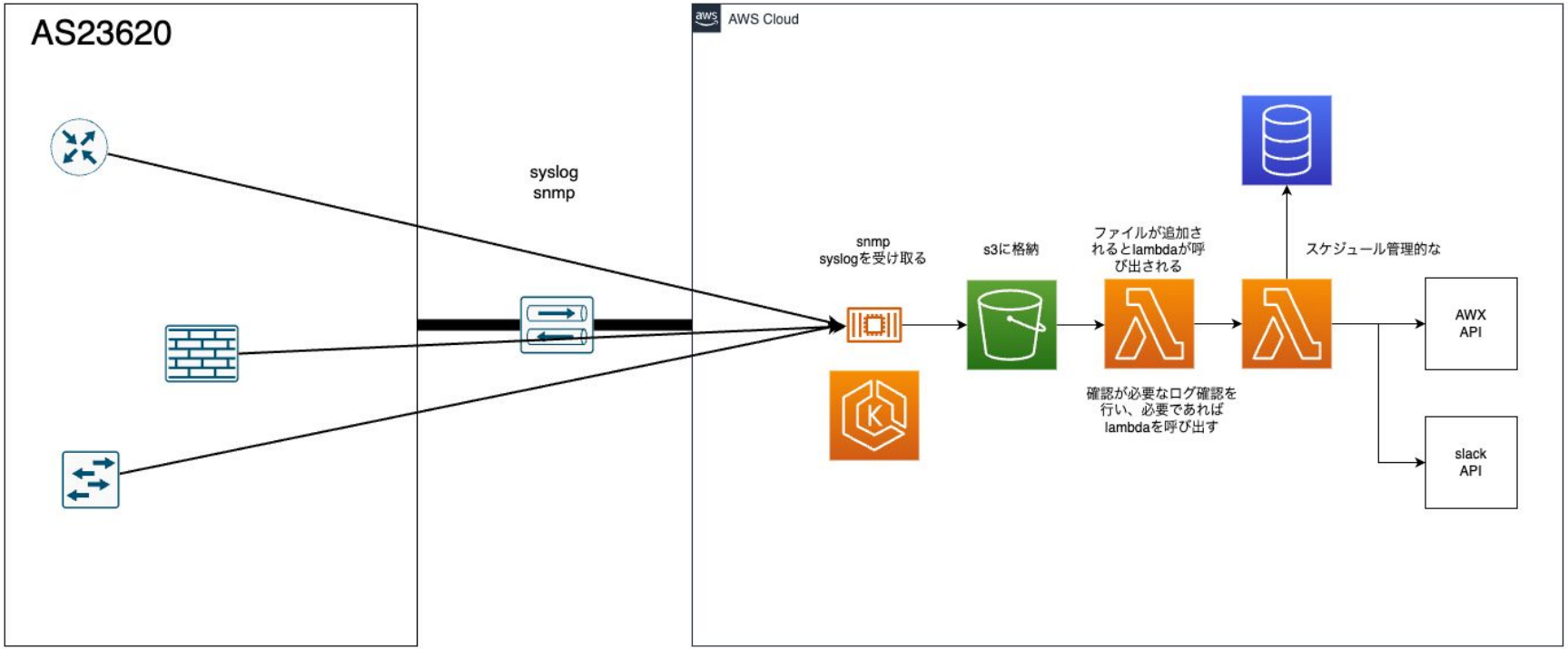
システム構成と検討構成

- アプリケーションエンジニアでは無いため、
  ある程度マネージドで管理の手間が省ける構成としたい
- 自動でスケールする構成にし費用を抑えたい
- 時代はサーバレスだ！

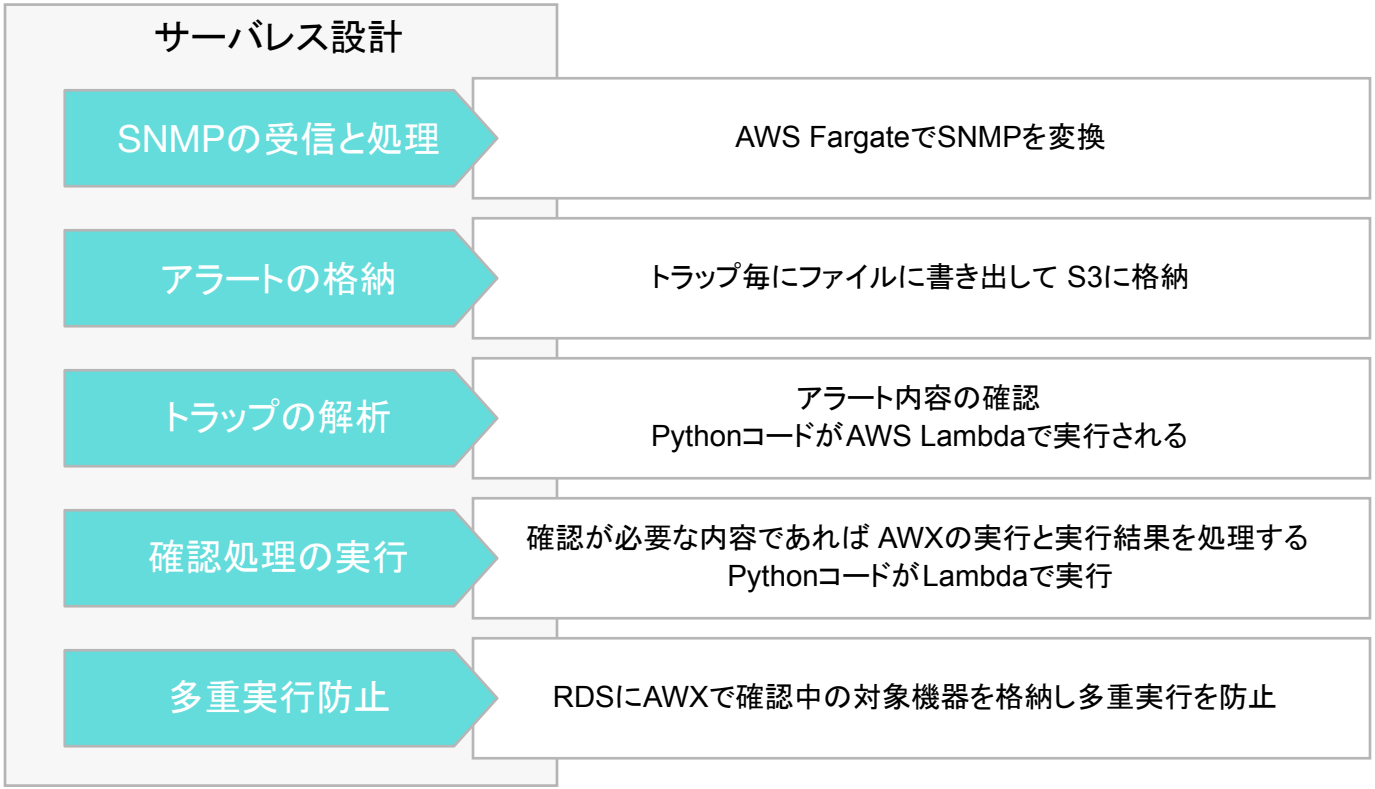
ということでAWSのマネージドサービスで**サーバレス**な設計を開始

システム構成と検討構成

サーバレス設計



システム構成と検討構成



システム構成と検討構成

この構成の問題点

1	AWS S3のコスト	S3はリクエスト数に対しても課金が発生するため リクエストの増加に合わせてコストが増加する
2	AWS Lambdaのコスト	アラート数によってリクエストが増加しコストも増加する
3	コストの予測が難しい	今後の機能追加や機器追加によってアラート数が変わってくる その月にどれだけアラートが出るかわからない
4	ECS Fargateのコスト	常にデータが来るので常時 1コンテナ以上が待機 処理速度によっては更に増える可能性も

システム構成と検討構成

この構成の問題点

5 管理できるのか

普段はネットワークの運用を行っているネットワークエンジニアであり、専属SREではない
マネージドサービス側のアップデートに直ぐに対応できるとは限らない
デバッグ自体が難しく方法がわからないことも

システム構成と検討構成

この構成の問題点

リクエスト数がどの程度になるのか計算してみると...

トラップ数とsyslog数の合計　：　秒間50程度

31日 × 24時間 × 60分 × 60秒 × 50 = 133,920,000

LambdaとS3のコストが**年間1万ドル**を超えそう

売上のないシステムで
このコストは無理

システム構成と検討構成

考えた結果…

- マネージドなサービスは便利だが使った分コストがかかる
- クラウド特有の考えるべき部分なども出てきて
  ネットワークエンジニアには大変そう
- AWXを実行するためにEKSの運用を既に行っている

サーバレスは断念

システム構成と検討構成

要件を整理し再検討

- SNMPを処理するコンテナは常駐、処理数によってスケール
- コスト計算が行いやすい構成
- 自前で必要な機能は実装する

既にEKSのクラスタの運用を行っておりそこではAWXしか動いていない

ノードのリソース不足となってもノードを追加するだけでいいので
コストがわかりやすい

となり、**EKSクラスタ上に実装** する形に決定

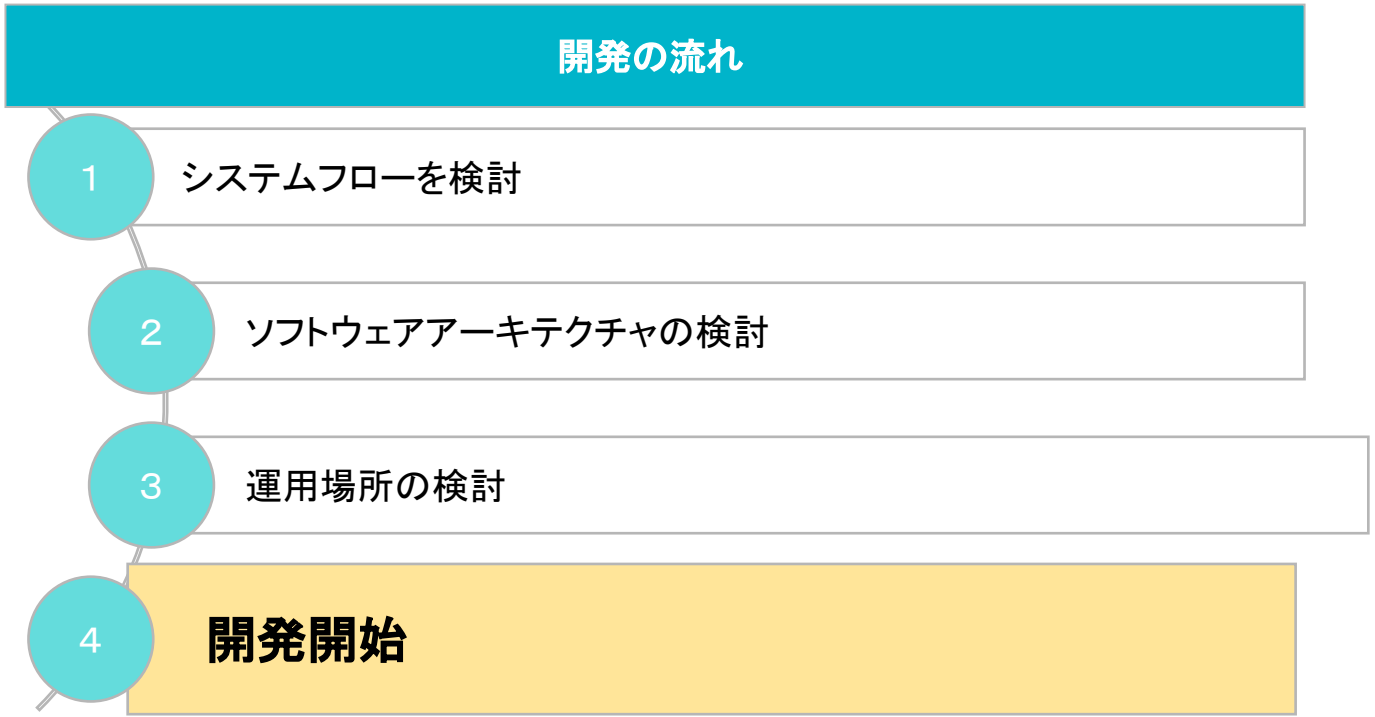
システム構成と検討構成

決まったことをまとめると

- マイクロサービスアーキテクチャ
- EKSにデプロイ
- コンテナで開発
- データのやり取りはAPI

システム構成と検討構成

開発の流れ



システム構成と検討構成

今回はサービスというより機能ごとに分割した

大きく分けて、

- 変換
- 判断
- 制御
- 問い合わせ処理
- 出力

の形で機能分割を行った

また、機能単位で担当者を決めて開発を実施した

システム構成と検討構成

アプリケーションを作ると

- DBへのアクセス
- 外部APIへのアクセス
- データの変換

など複数の動作をさせなければならないことが多い

しかし、DBへのアクセスはDberのみから実施
外部APIからの情報取得はpagodaCallerから実施など
極力**抽象化**できるように分離を行った

開発時の課題

開発当時は自動化はプロジェクト的タスクのうち一つ

DMM.comのサービスへの直接的な影響がないため、複数の問題が発生

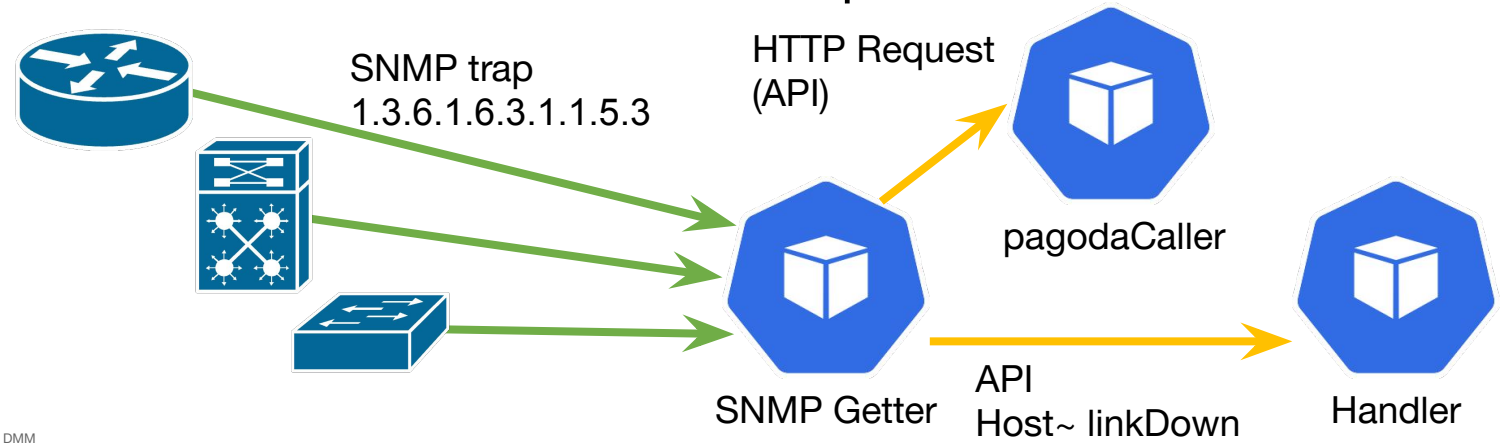
- ネットワークの設計運用などが優先で開発は優先度が低い
- 専任メンバーが不在
  - ネットワークの設計から開発など、業務が切り替わる度に
    開発者の脳のコンテキストスイッチが発生
  - 日勤帯では問い合わせなどもあるため
    定時後に開発時間をまとめて取得

これにより、当初の想定より開発期間が延長

単体機能の紹介

SNMP Getter

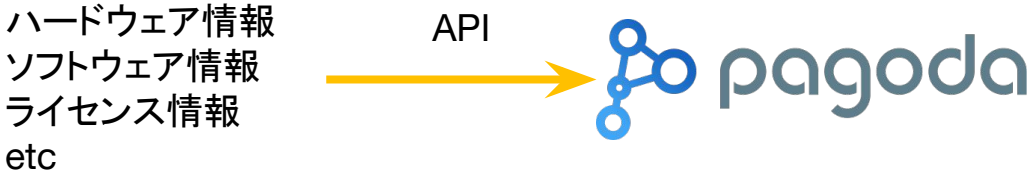
- 文字通りネットワーク機器からのSNMP trapを受信するPod
- 後続処理のためにSourceIPでpagodaCallerにHost名を問い合わせ
- SNMP trapのoidからMIBに変換
- 上記の情報を集約してHandlerにRequestを送信



単体機能の紹介

pagoda

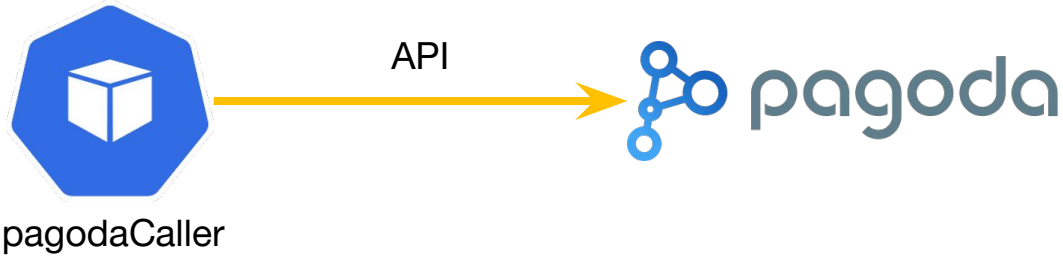
- DMM.comで内製開発しているSSoTツール(OSS)
- 現在外部向けにデモサイトも公開中(リンク)
- ここにデータが集約されているため、
  各種構成情報の取得に活用中



単体機能の紹介

pagodaCaller

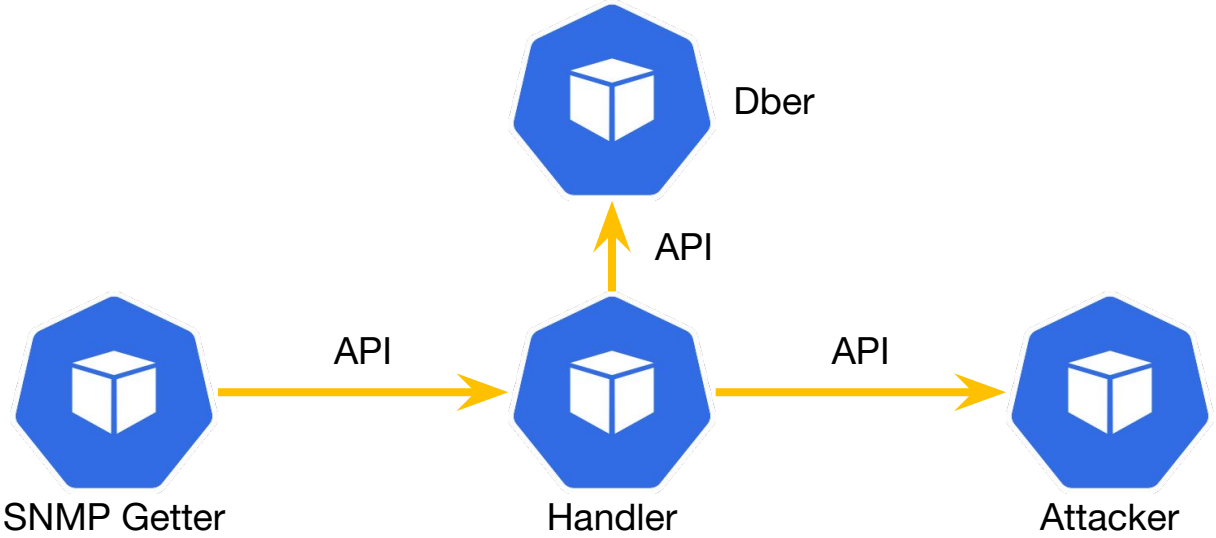
- pagodaにHTTP Requestを行うPod
- EKS環境内からpagodaに情報取得を行う場合はここが一度受ける
- URLとBodyからpagodaに検索Requestを投げ、
  特定の情報に絞り込んで応答
- pagodaに変更があった際の影響範囲をこのPodだけに
  絞り込んでいる



単体機能の紹介

Handler

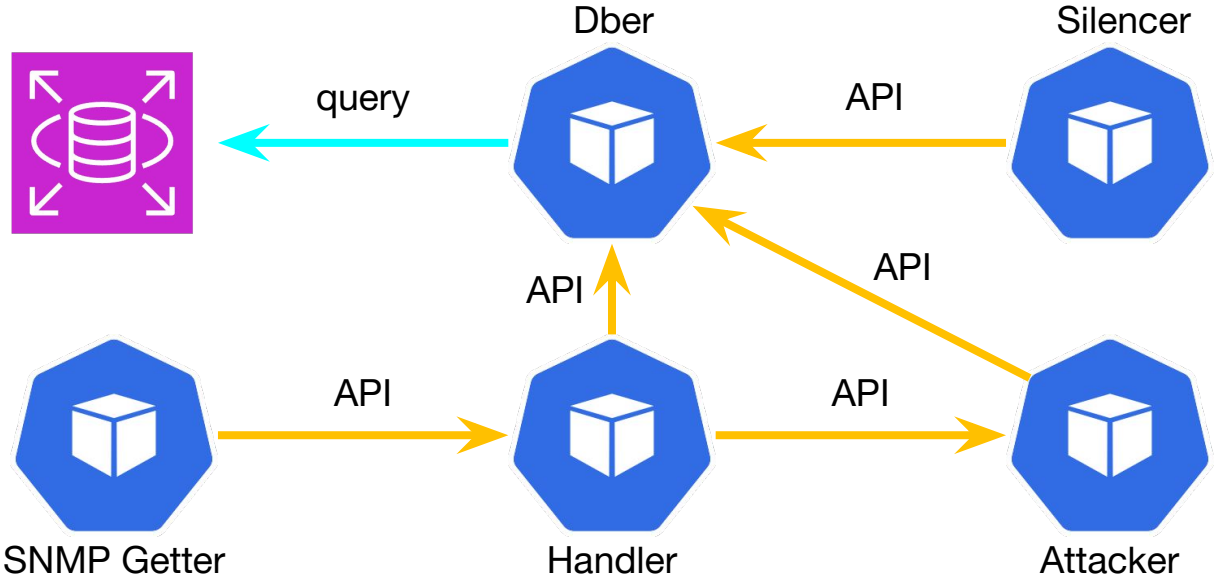
- SNMP GetterからAPIでアラートデータを取得
- SNMPのoidとosの情報を使ってDberへ確認が必要なアラートか確認
- アラート確認が必要であればAttackerのAPIにデータを渡す



単体機能の紹介

Dber

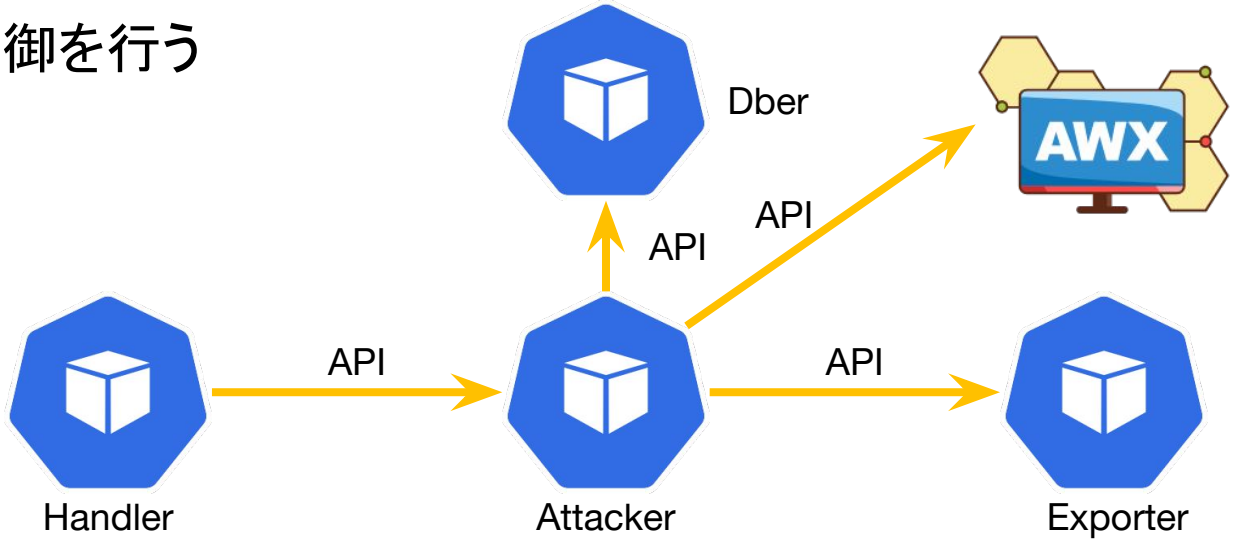
- FANNELで使用してるDBへアクセスを行う
- 静観対象の機器をDBに登録する



単体機能の紹介

Attacker

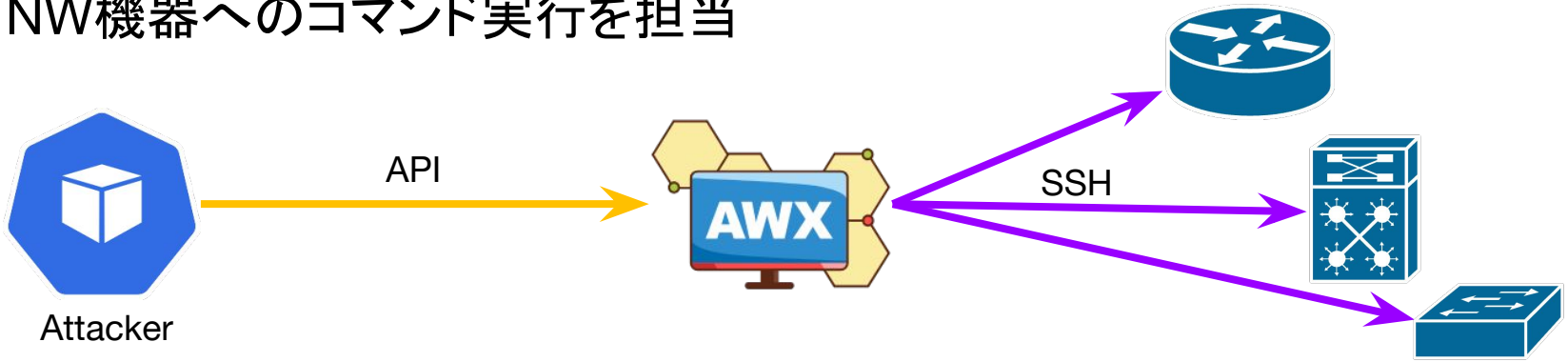
- アラートを発報した機器の確認を行う
- 確認結果をExporterにAPIで渡す
- 同じインターフェイスに同時にステータス確認を実行しないように同時実行制御を行う



単体機能の紹介

AWX

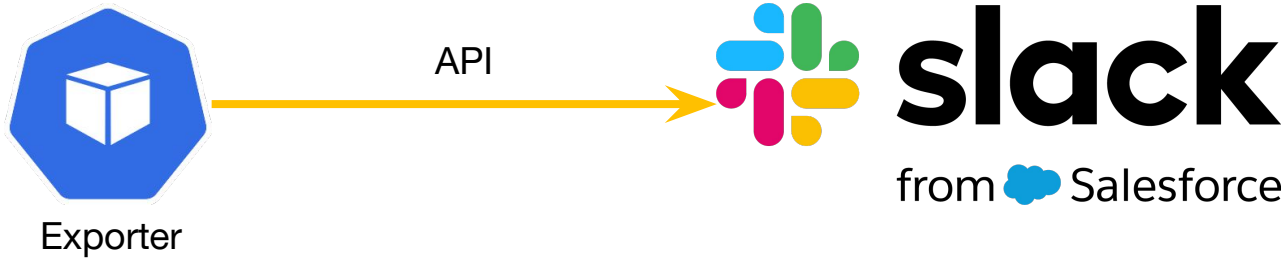
- 皆さんご存知AnsibleのGUIのOSSアプリ
- Secret管理やGUIからの実行、承認など色々あって運用上便利
- ネットワークの自動化はAnsibleで全部やるつもりだった
- API経由で実行もできるので、FANNELに組み込む形で
  NW機器へのコマンド実行を担当



単体機能の紹介

Exporter

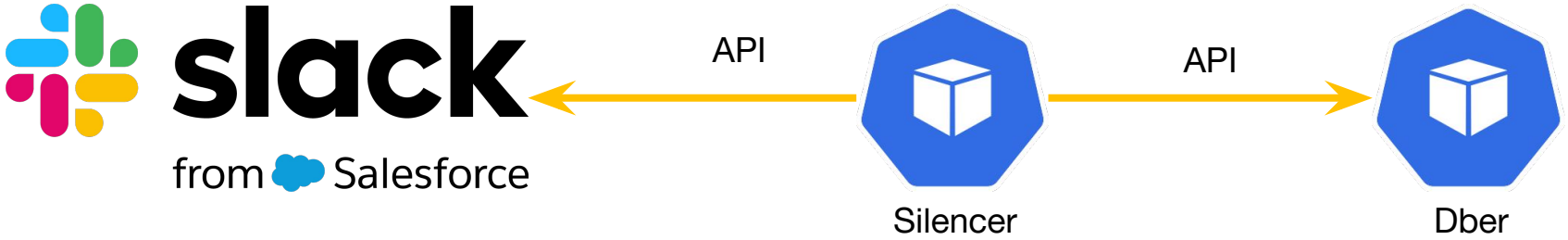
- 受け取ったデータを外部に通知するためのコンテナ
- 現状はSlackに対しての通知のみ実施
- 一定期間内に同一Hostからアラートが発生している場合(flapなど)
  既存スレッドに追加する形で投稿



単体機能の紹介

Silencer

- 静観情報を取得してDberに登録するコンテナ
- 既存の運用で架電抑制のためにSlack投稿をしているので
  その情報を取得してFANNELの動作を停止/再開させる
- 作業でも動作するのを止めたくなって後から作成



Dockerによるマイクロリリース

EKSの為のmanifestの準備と並行して、挙動の確認を実施するため
1VMで全てのコンテナを起動する形でマイクロリリース

すると

- 連携自体は問題なく、全体的な動作はOK
- overlay2が肥大化してVMごとDown
- 負荷の高い前段側コンテナが時々落ちる

などなどの問題

やっぱり早くEKS載せよう

EKS移行と苦労

1. 状態を持ったコンテナ問題
2. NLBのマニフェスト管理
3. すべてのPodが異常判定
4. OOM Killerの頻発

EKS移行と苦労

1 状態を持ったコンテナ問題

Dberは内部にDBの構造を保持していたため、
すべてのPodに同じ構造データなければクエリの生成ができない

解決策

EFSを各Podの特定のディレクトリにマウントさせることにより
同じデータを参照できるように変更

スケール時にも問題なくクエリを実行できる

EKS移行と苦労

2 NLBのマニフェスト管理

VMで動作させるためにNLBを作成、付与されたIPに対してSNMPを流していたが、NLBを作り直してしまうとIPが変わってしまう
可能性があるため再作成せずマニフェスト管理する必要があった

解決策

マニフェストにNLBの状態を書き込むだけでは管理できなかった

AWS Load Balancer Controllerで作成されたリソースには特定のタグが付与されるため既存NLBにも付与

正常にPodが認識されるようになった

EKS移行と苦労

3 すべてのPodが異常判定

SNMP GetterはUDP(SNMP trap)で
パケットを受信する仕組みのため、応答を返さない仕様

このため、

- SNMP trapを処理するプロセス(PID1)
- ヘルスチェック用プロセス

の2つのプロセスがコンテナ内で稼働しているが、
OOM Killerの発動によりヘルスチェック用プロセスが停止

しかし、SNMPを処理するプロセス自体には問題がないため、
コンテナの再起動は発生しない

EKS移行と苦労

その結果、NLBからはヘルスチェックがNGと判定されるものの、コンテナの再起動が行われないため、最終的にすべてのPodがNGとなり、パケットを処理するPodが存在しない状態に陥った

解決策

kubeletからヘルスチェックをしてくれるProbe機能を使用

Liveness Probeの設定を行うことで異常判定されたPodを再起動するようにした

EKS移行と苦労

4 OOM Killerの頻発

Dockerではメモリ使用量は増加しなかったが、
EKSに載せた際メモリが増加し続ける現象が発生

ターミナル出力によってメモリ使用量が増加、
毎日何かしらのPodが再起動している

未解決

対策検証中

今後の拡張と展開

ここまで凄そうに見せておいてアレですが

- 現状対応しているのはlinkDownだけ
- PCなら見れるけどスマホじゃ確認しづらい
- 構想時に後回しにした機能が未実装のまま
  - コマンド結果の判定機能をAttackerとExporterの間に
- 構想時にはなかった機能の追加開発中
  - 当初のアーキテクチャで実装したことで見えてきた課題とも言える

というのが実情

今後の拡張と展開

今後の拡張としては以下などを進めたい気持ち

- 対応trapの追加
- 確認コマンドログの送付しかしていないので
  判定処理を中間に入れたい
- Host単位での静観しかないので、条件付き静観したい
  Hostのこのインターフェースの静観のようなイメージ
- コマンド結果に応じた動的対応をしたい
- FANNEL自体のログを現状捨ててるのをやめたい
- CI/CDでよしなになるようにしたい

ちなみに

DMM.comでは、一緒に働くメンバーを募集しています

もし興味のある方はメンバーにお声がけください！

まとめ

- Zabbix を使わないためにZabbix の一部機能みたいなものを作ったことになった
- 使用者と開発者が同じだからこそ見えるものもある
- 作って実際に使おうとすることで見えてくる課題もある
- 拡張性を持たせた設計にしておけば後々の変更が簡単
- 設計構築運用と開発を並列でこなすのはきつい
- 考えたことを形にするのは楽しい
- MicroService やり出すとContainer の便利さに気づける
- Container 化してDeploy することや、EKS の管理は慣れれば楽
  Managed Service のようにCost 計算も難しくないため合っていた

討論等

例として以下のような点で議論できれば

勿論記載ない点でもOK

- ネットワーク運用自動化ツールは、
  自分たちで造るものなのか / それ以外の方法を取るか
- 造る場合はネットワークの検証や構築、運用と
  開発の両立はどうするべきか
- 過去の設計や技術的な制約(技術負債)にどのように対応するべきか
- コード規約の策定、コード自体の保守についてどうすべきか