Session #3

Fluent Bit による Kubernetes ロギング

アマゾン ウェブ サービス ジャパン 株式会社
ソリューションアーキテクト
浅野 佑貴

#EKSMatsuri

Who are you?

名前：浅野 佑貴（あさの ゆうき）

所属：アマゾンウェブサービスジャパン株式会社
　　ソリューションアーキテクト

ロール：製造業のお客様を中心に、技術的なご支援を担当

経歴：国内SIerにてクラウド関連ビジネスの推進を担当

好きなAWSサービス： Consolidated billing for AWS Organizations

本日のアジェンダ

- 本セッションで取り扱うロギングとは
- Fluent Bit とは
- Flunet Bitを利用したロギング環境の構築

本日のアジェンダ

- 本セッションで取り扱うロギングとは
- Fluent Bit とは
- Flunet Bitを利用したロギング環境の構築

本セッションで取り扱うログとは

アプリケーションログ

主にDev向け

アプリケーションのパフォーマンスに関わるログ

application log: /var/log/myapp.log

Web logs: journalctl -u nginx.service

Transaction logs

Others

インフラ管理用ログ

主にOps向け

システムの安定性・可用性に関わるログ

Syslog and other OS logs

Audit logs

Performance metrics

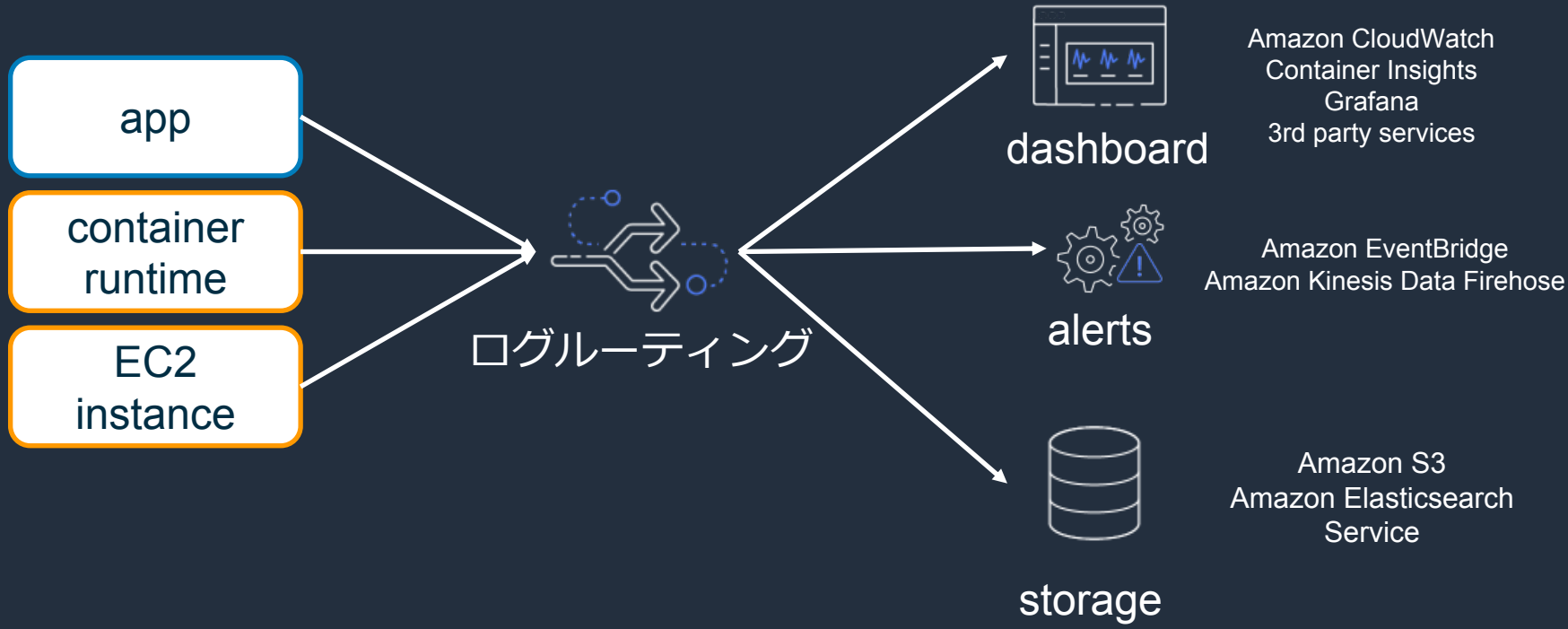
Kubernetes における ロギングアーキテクチャ

- Kubernetes自体は cluster-levelのネイティブなロギングソリューションをサポートしていない。ユーザ側で仕組みを構築する必要がある
    - ロギングには、Woker Node、Pod から独立した個別のストレージとライフサイクルが必要となる

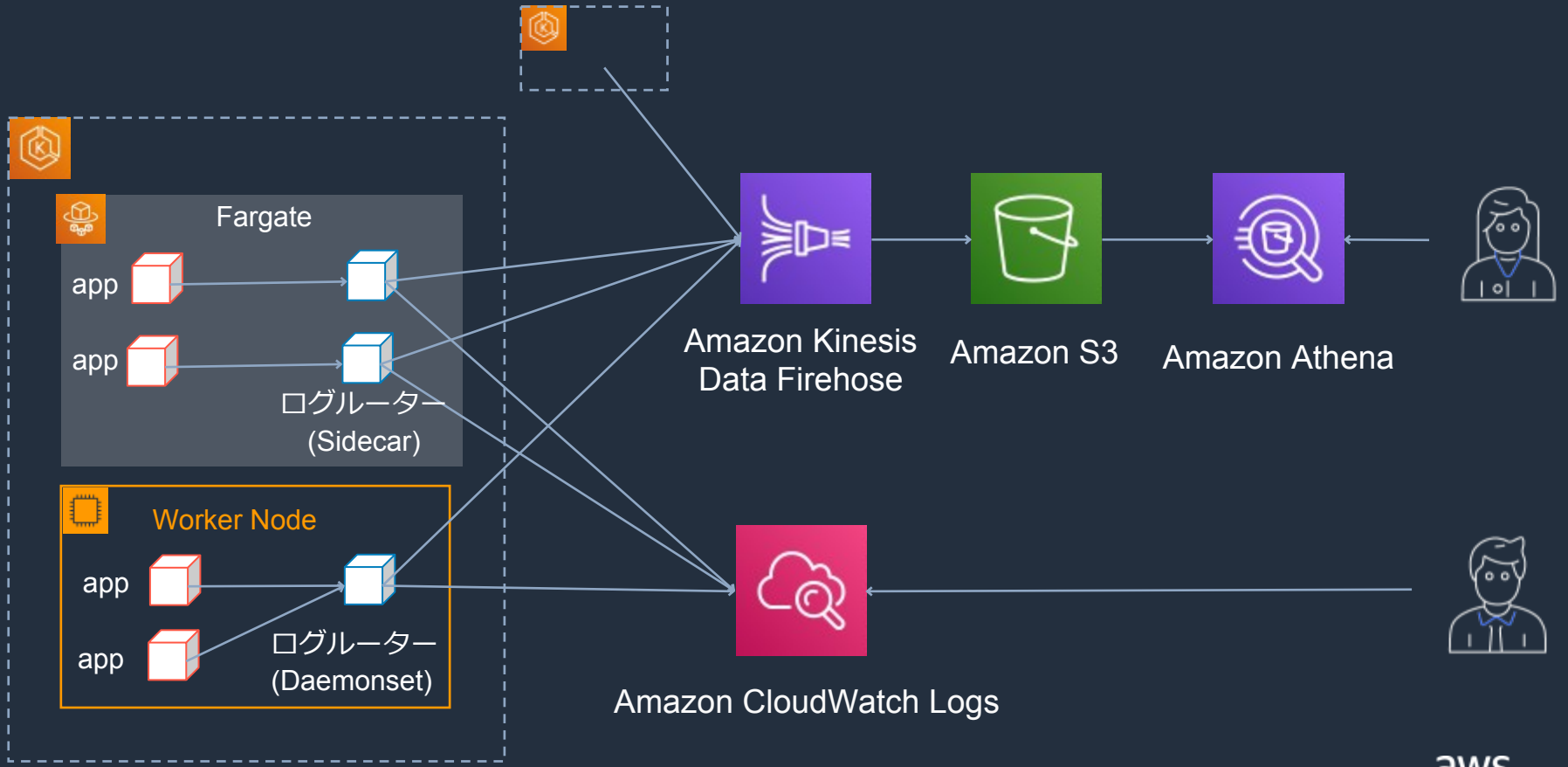
- 3つの典型的なアプローチ方法
    - Worker Node 単位でロギングエージェントを動作させログを集約する
    - アプリケーション Pod に、ロギングを行う Sidecar コンテナを含めてログを集約していく
    - アプリケーションから直接外部にログを送信する

https://kubernetes.io/docs/concepts/cluster-administration/logging/

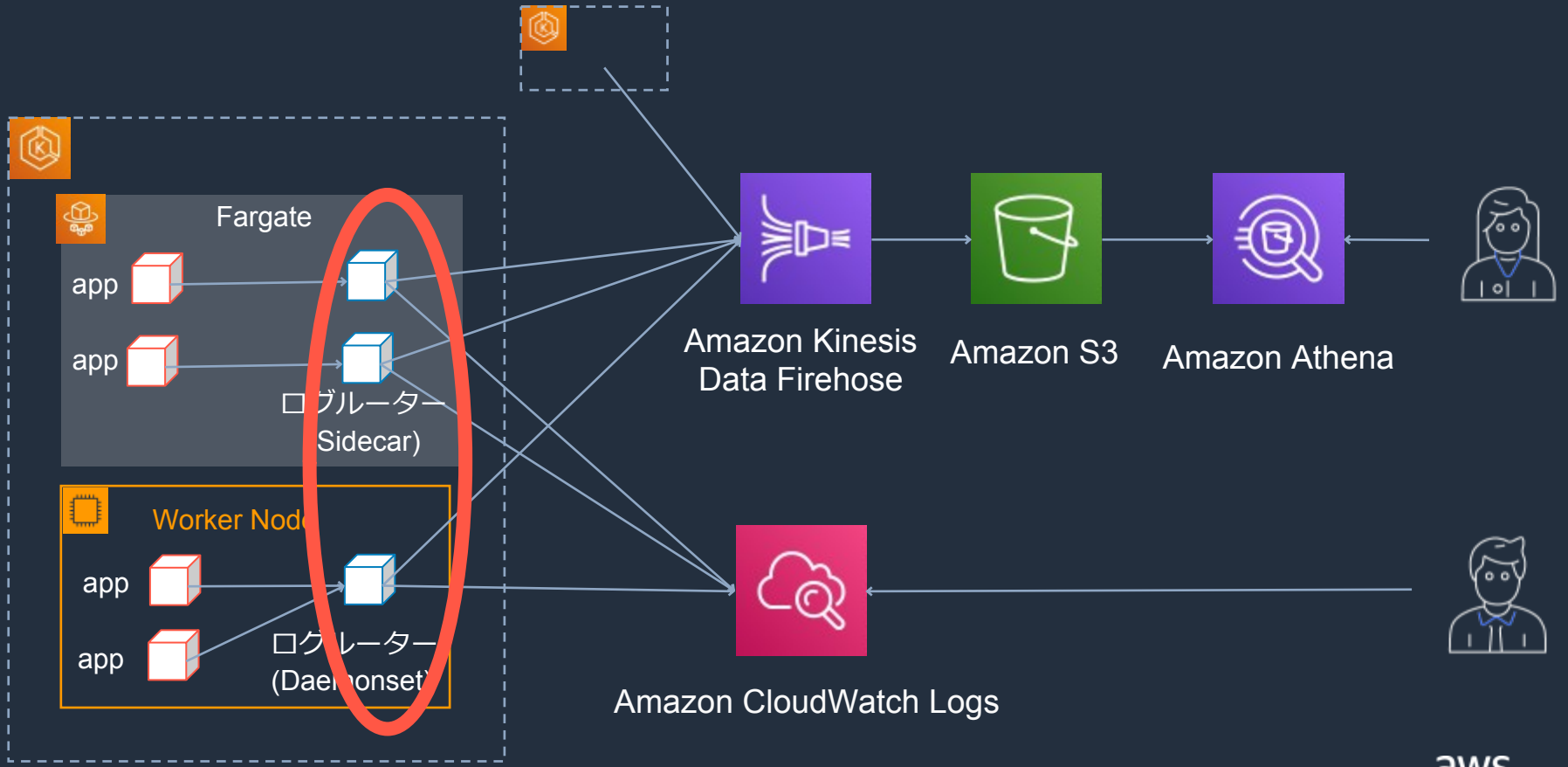
cluster-levelのロギング構成例



Amazon EKS上での構成例



Amazon EKS上での構成例



本日のアジェンダ

- 本セッションで取り扱うロギングとは
- Fluent Bit とは
- Flunet Bitを利用したロギング環境の構築

fluentbit

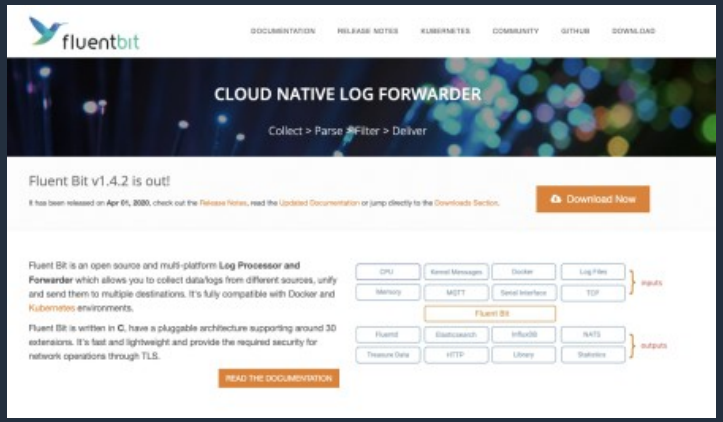
Fluent Bitとは？

- 組み込み Linux/Gateway などの制約のある環境での利用も想定された軽量ログプロセッサー/ フォワーダー

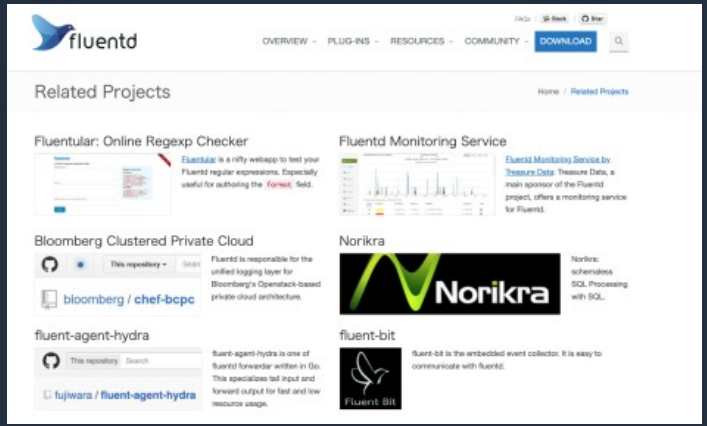
- Fluentd傘下のCNCFサブプロジェクト
  - Treasure Data 社が現在のプライマリースポンサー
  - AWSメンバーもCo-maintainerとして参画

- License: Apache License v2.0

- Latest version: 1.4.2

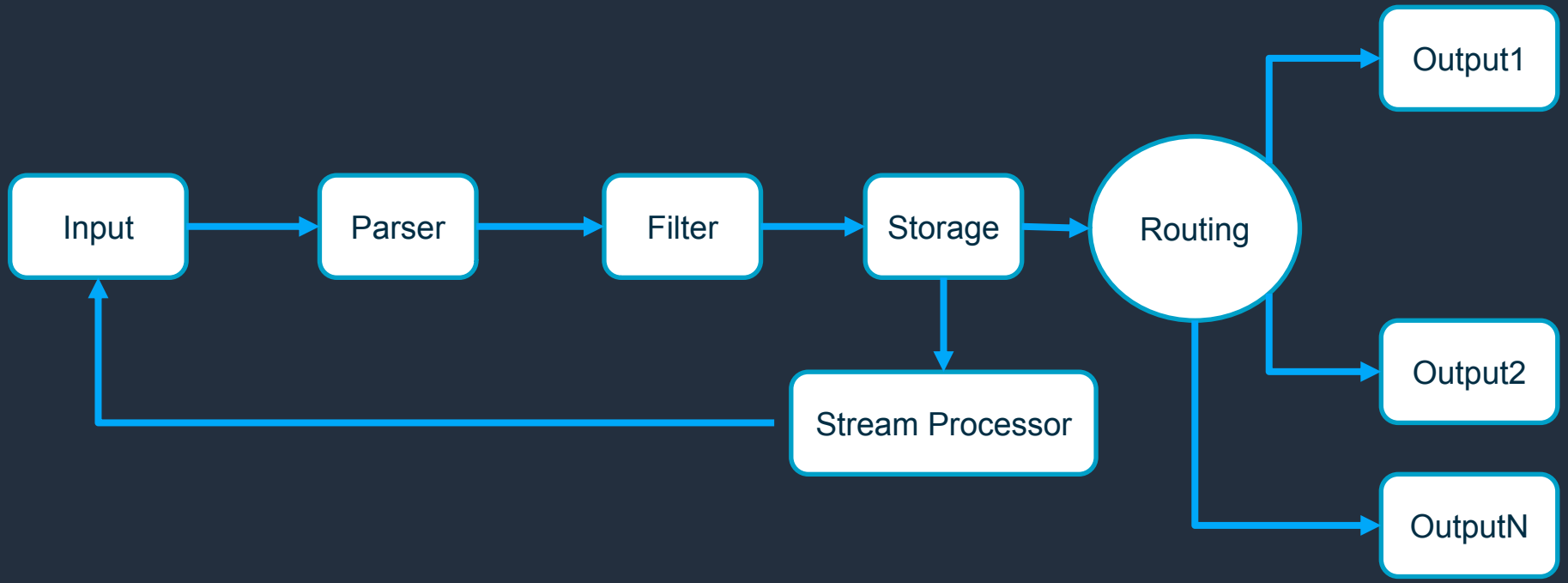


https://fluentbit.io/

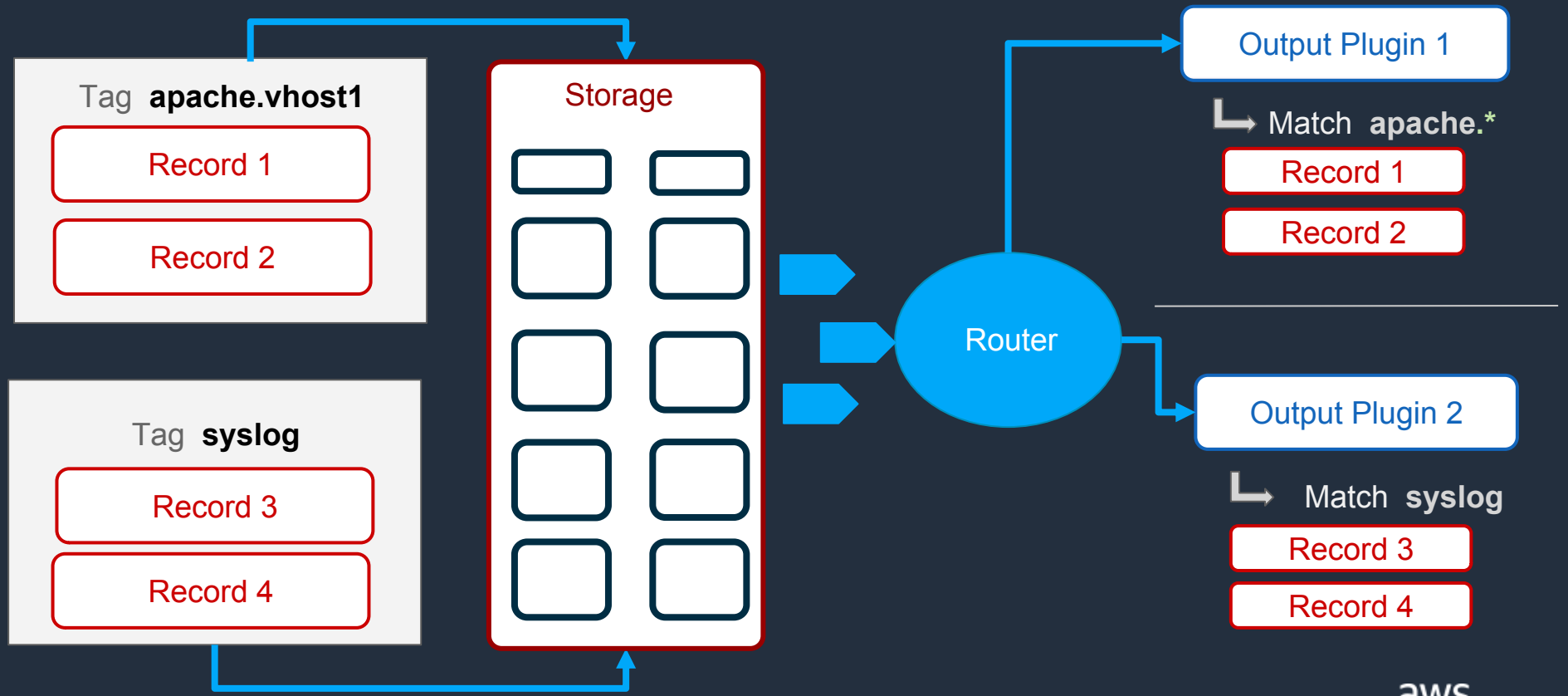


https://www.fluentd.org/related-projects

Fluent Bit ログ処理パイプライン



ロギングとルーティング



ストリームプロセッシング

- SQLクエリを使って新しいストリームを作る事が可能
  - CREATE STREAM users AS SELECT user from STREAM:apache;
  - CREATE STREAM users WITH (tag=‘users’) AS SELECT user from STREAM:apache;

- Tagを付与して再度、ストリームをFluent Bit data pipelineに入れる事ができる
  - ログの中身に応じてOutput先を振り分ける事に使える
    - 例：errorログのみCloudwatchに送信

https://aws.amazon.com/jp/blogs/opensource/splitting-application-logs-multiple-streams-fluent/

Fluentd と Fluent Bit

	Fluentd	Fluent Bit
Scope	Containers / Servers	Embedded Linux / Containers / Servers
Language	C & Ruby	C
Memory	~ 40MB	~ 650KB
Performance	High Performance	High Performance
Dependencies	依存関係あり (Ruby Gemで構成)	依存関係なし (特定のプラグインを除く)
Plugins	1000以上のプラグイン	70程度のプラグイン
License	Apache License v2.0	Apache License v2.0

https://docs.fluentbit.io/manual/about/fluentd-and-fluent-bit

AWSサービスとの連携

- Output プラグインにより連携可能なAWSサービス
    - Amazon CloudWatch Logs
    - Amazon Kinesis Data Firehose
    - Amazon kinesis Data Streams

- 上記プラグインは、Worker NodeのEC2 instance role または EKS IAM Roles for Service Accounts for podsを通じてcredentialを取得可能

- AWS for Fluent Bit Docker Image
    - 上記のプラグインがプリインストールされたDockerイメージ
    - Fluent Bit + プラグインの組み合わせで管理されており、Fluent Bitとは異なる独自のバージョニングされている
    - https://github.com/aws/aws-for-fluent-bit

AWSサービスとの連携

- Amazon Elasticsearch Service
  - Fluent Bit v1.4ではexperimental support
  - credentialを環境変数経由で渡す必要がある
  - v1.5からフルサポートになる計画

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token
```

```
[OUTPUT]
    Name  es
    Match *
    Host  xxx.<region_name>.es.amazonaws.com
    Port  443
    Index my_index
    Type  my_type
    AWS_Auth On
    AWS_Region <region_name>
    tls     On
```

パフォーマンス検証例

CloudWatch プラグイン: Fluentd vs Fluent Bit　利用EC2インスタンス：c5.9xlarge

Log Lines Per second	Data Out	Fluentd CPU	Fluent Bit CPU	Fluentd Memory	Fluent Bit Memory
100	25 KB/s	0.013 vCPU	0.003 vCPU	146 MB	27 MB
1000	250 KB/s	0.103 vCPU	0.03 vCPU	303 MB	44 MB
10000	2.5 MB/s	1.03 vCPU	0.19 vCPU	376 MB	65 MB

Kinesis Firehose プラグイン: Fluentd vs Fluent Bit利用EC2インスタンス：c5.9xlarge

Log Lines Per second	Data Out	Fluentd CPU	Fluent Bit CPU	Fluentd Memory	Fluent Bit Memory
100	25 KB/s	0.006 vCPU	0.003 vCPU	84 MB	27 MB
1000	250 KB/s	0.073 vCPU	0.033 vCPU	102 MB	37 MB
10000	2.5 MB/s	0.86 vCPU	0.13 vCPU	438 MB	55 MB

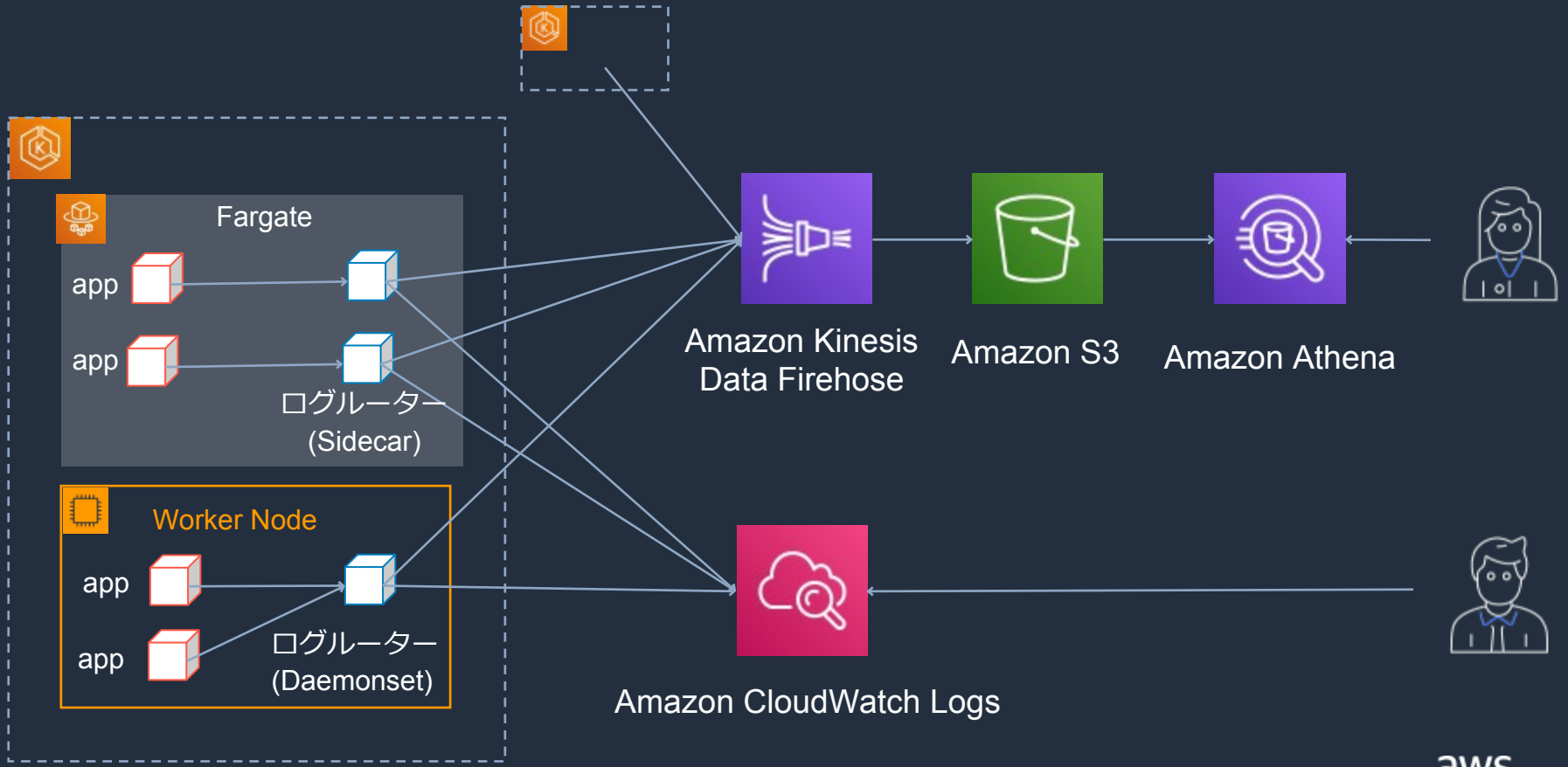
検証例ですので、これらのデータが性能を保証するものではありません。
ご自身のワークロードにおけるパフォーマンスについては、別途検証下さい。

https://aws.amazon.com/jp/blogs/news/centralized-container-logging-fluent-bit/

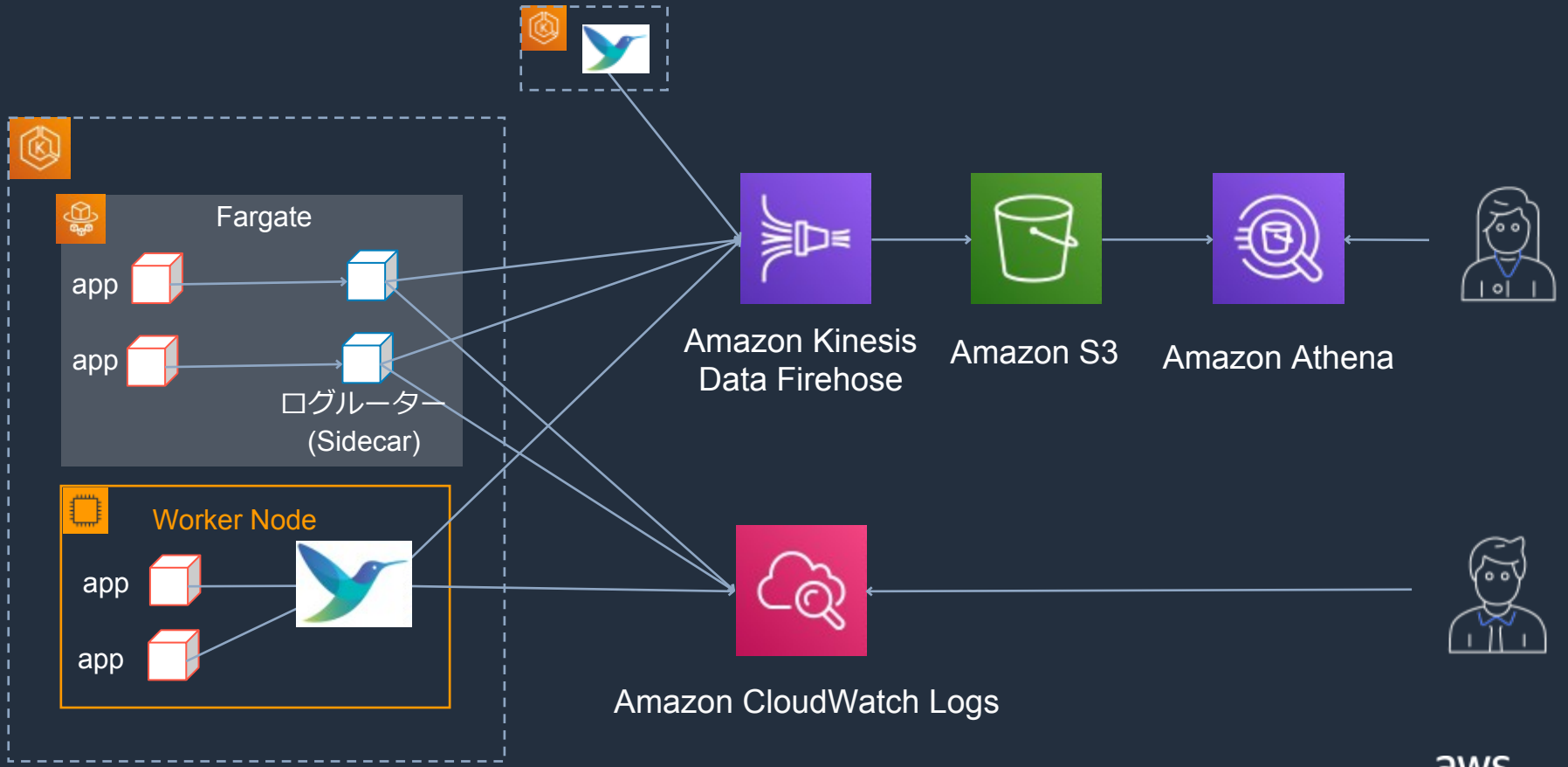
本日のアジェンダ

- 本セッションで取り扱うロギングとは
- Fluent Bit とは
- Flunet Bitを利用したロギング環境の構築

Amazon EKS上での構成例(再掲)



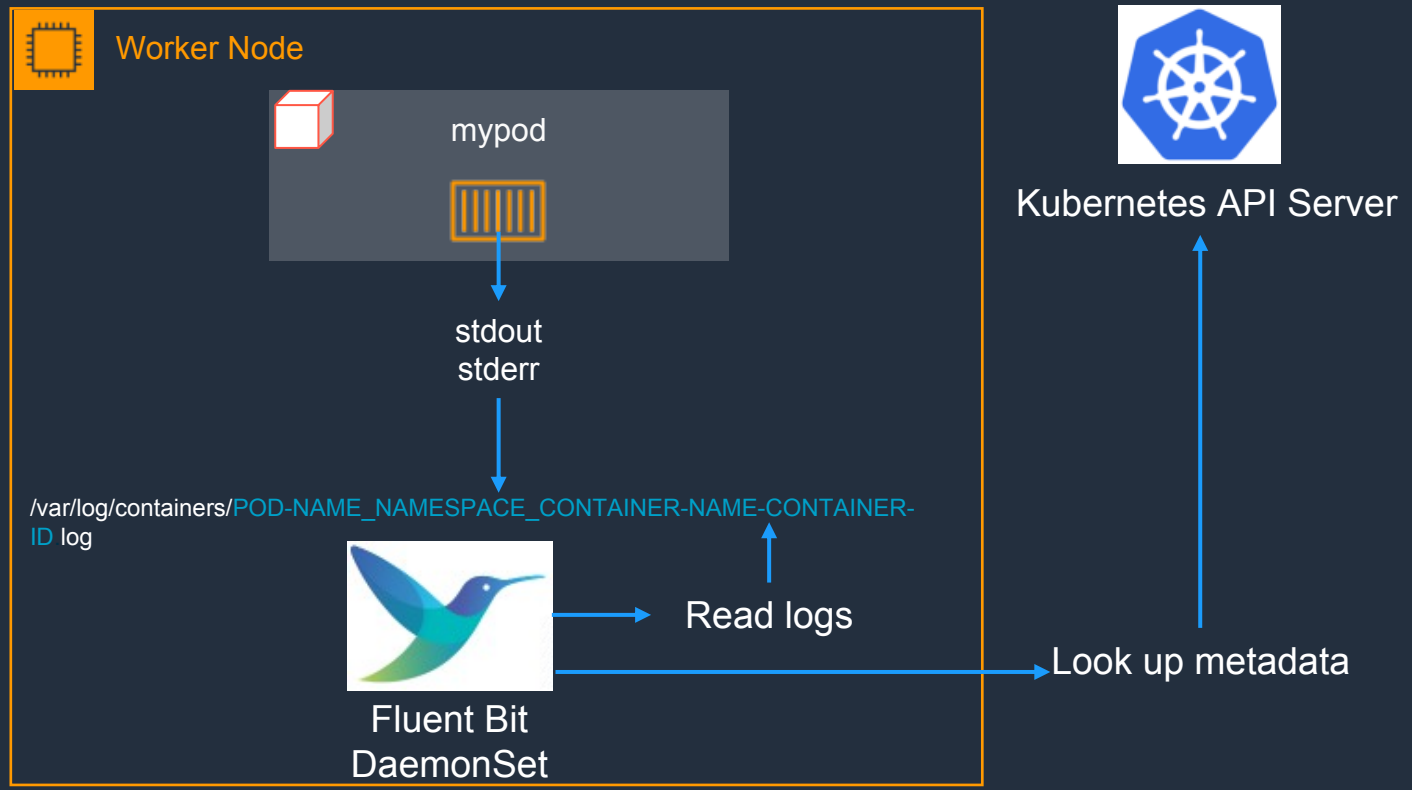
Amazon EKS上での構成例



Fluent BitによるK8s metadataの look up

ログファイル名から取得する metadata
- Pod name
- Namespace
- Container name
- Container ID

API Server から取得する metadata
- Pod ID
- Labels
- Annotations



How to Setup

- 事前準備
    - 関連するAWSサービスのセットアップ

1. EKS Clusterの作成
2. Worker Node にAWSサービスにアクセスする為のIAMポリシーを付与
3. Kubernest RBACの設定
4. ConfigMapを利用したFluent Bitで利用する設定情報の作成
5. Fluent Bit の deploy

EKS Cluster の作成と Worker Node への IAMポリシーの割り当て

手順1・2/5

eksctlを使ったEKS Cluster の作成

```
$ eksctl create cluster --name eksmatsuri-fluentbit --region <region_name>
$ kubectl get nodes
```

Worker Nodeが利用しているIAMロールにFirehose、Cloudwatch関連に対するアクションを許可するIAMポリシーを割り当て

```
$ ROLE_NAME=$(eksctl get nodegroup --cluster eksmatsuri-fluentbit --region <region_name>
 -o json | jq -r '.[].NodeInstanceRoleARN' | cut -f2 -d/)

$ aws iam put-role-policy --role-name $ROLE_NAME ¥
--policy-name eksmatsuri-fluentbit-ds ¥
--policy-document file://eksmatsuri-fluentbit-daemonset-policy.json
```


aws

Kubernest RBACの設定

手順3/5

サービスアカウントの作成する

```
$ kubectl create sa <sa_name>
```

ロールとロールバインディングを作成する

```
$ kubectl apply -f eksmatsuri-fluentbit-daemonset-rbac.yaml
```


aws

Kubernest RBACの設定

手順3/5

```
$ cat eksmatsuri-fluentbit-daemonset-rbac.yaml
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: pod-log-reader
rules:
- apiGroups: [""]
  resources:
  - namespaces
  - pods
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: pod-log-crb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: pod-log-reader
subjects:
- kind: ServiceAccount
  name: <sa_name>
  namespace: default
```

DaemonSetで deployするFluent Bit Podで利用する。作成したServiceAccount名に合わせて修正してください

aws

ConfigMapを利用した Fluent Bitで利用する設定情報の作成

手順4/5

```
$ kubectl apply -f eksmatsuri-fluentbit-configmap.yaml
```

```
$ cat eksmatsuri-fluentbit-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: fluent-bit-config
  labels:
    app.kubernetes.io/name: fluentbit
data:
  fluent-bit.conf: |
    [SERVICE]
        Parsers_File  parsers.conf
    [INPUT]
          <略>
    [FILTER]
          <略>
    [OUTPUT]
          <略>
parsers.conf: |
    [PARSER]
          <略>
```

Fluent Bitの設定は以下の様なセクションに分割して記載する

- SERVICE
  - 主にグローバルなプロパティ関係
  - 追加で読み込む設定ファイル
- INPUT
  - inputプラグインで利用するソースを定義
- FILTER
  - 入力データに対して適用するフィルター定義
- OUTPUT
  - ログの出力先の定義
  - 複数に出力したい場合は複数定義
- PARSER
  - 指定されたデータをパースする為の定義

aws

Fluent Bit の deploy

手順5/5

```
$ aws ssm get-parameters-by-path --path /aws/service/aws-for-fluent-bit/
$ kubectl apply -f eksmatsuri-fluentbit-daemonset.yaml
```

```
 apiVersion: apps/v1
kind: DaemonSet
<省略>
- name: aws-for-fluent-bit
        image: 906394416424.dkr.ecr.us-east-2.amazonaws.com/aws-for-fluent-bit:2.3.0
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
   <省略>
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

- AWS for Fluent Bit Docker Imageは Systems Manager 経由で対象リージョンにあるECRイメージのリストを取得可能
- Worker Nodeのlogディレクトリをマウント

Fluent Bit サンプルコンフィグ

```
[SERVICE]
    Parsers_File  parsers.conf
[INPUT]
    Name              tail
    Tag               kube.*
    Path              /var/log/containers/*.log
    Parser            docker
    DB                /var/log/flb_kube.db
    Mem_Buf_Limit     5MB
    Skip_Long_Lines   On
    Refresh_Interval  10

[FILTER]
    Name kubernetes
    Match kube.*
    Kube_URL https://kubernetes.default.svc.cluster.local:443
    Merge_Log On
    Merge_Log_Key data
    K8S-Logging.Parser On
    K8S-Logging.Exclude On
```

- /var/log/containers配下のログファイルをinput sourceとし、tail プラグインで読み込む
- Kubernetesフィルターを利用してログエントリにmetadataを付加
- incomming messageにlog fieldが存在し、JSONフォーマットの場合にDataにmapする
- 各パラメータの詳細はドキュメントを参照
  - https://docs.fluentbit.io/manual/pipeline/filters/kubernetes

aws

Kubernetes フィルターによる付加される metadata


```
{
    "agent": "curl/7.61.1",
    "code": "200",
    "host": "-",
    "kubernetes": {
        "annotations": {
            "kubernetes.io/psp": "eks.privileged"
        },
        "container_hash": "nginx@sha256:86ae264c3f4acb99b2dee4d0098c40cb8c46dcf9e1148f05d3a51c4df6758c12",
        "container_image": "nginx:1.17",
        "container_name": "nginx",
        "docker_id": "400170d1301d08045f168203876403b4a1242c7bf2af15a349240574c924a6b0",
        "host": "ip-192-168-25-206.us-east-2.compute.internal",
        "labels": {
            "app": "nginx",
            "pod-template-hash": "6c5bfff4d9"
        },
        "namespace_name": "default",
        "pod_id": "403f80a9-a4b6-434e-8939-789cdfa70156",
        "pod_name": "nginx-6c5bfff4d9-87gmr"
    },
    "method": "GET",
    "path": "/",
    "referer": "-",
    "remote": "127.0.0.1",
    "size": "612",
    "user": "-"
}
```

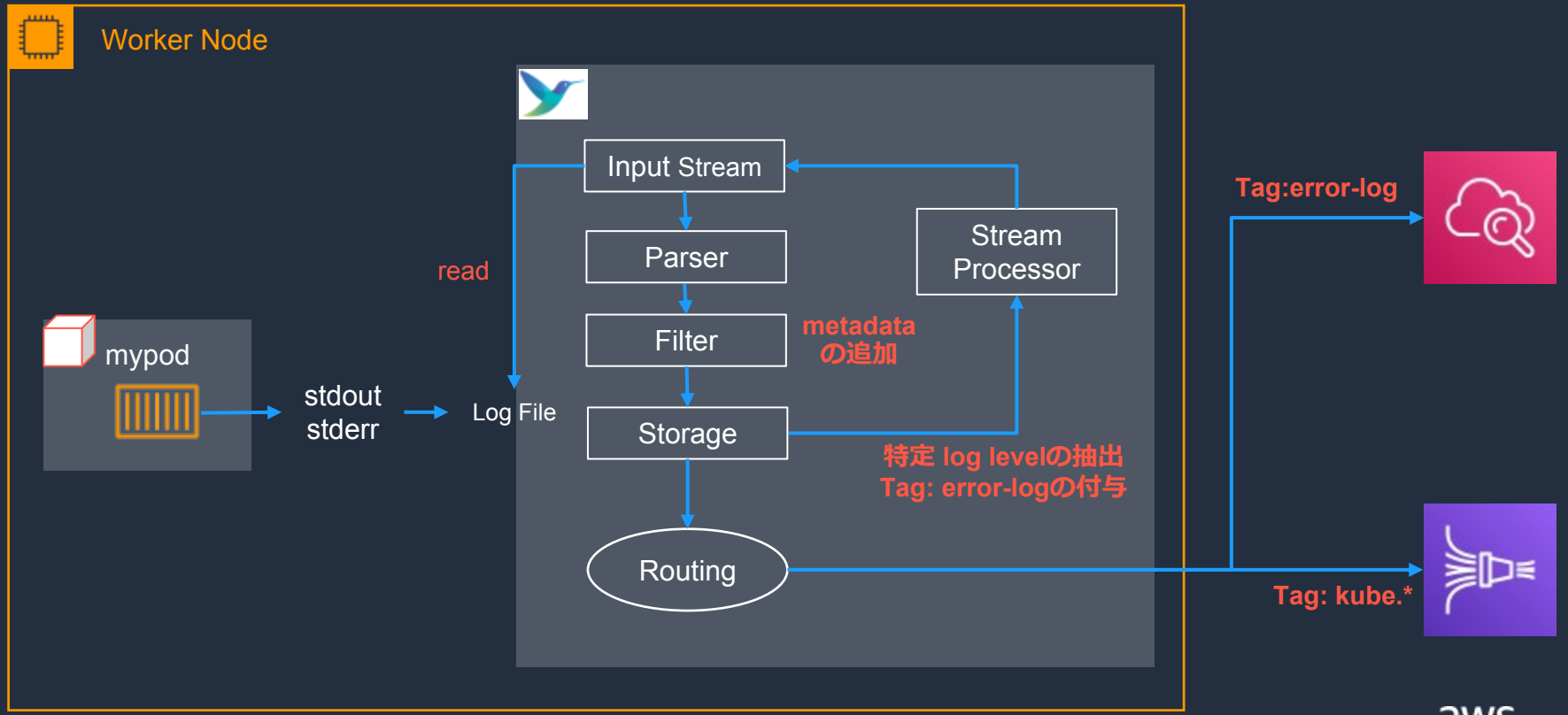

- Cloudwatch Logsへの出力例
- nginx podが出力したアクセスログに対して metadataを付与される

Fluent Bitサンプルコンフィグ - 複数のログ出力先

```
[OUTPUT]
        Name firehose
        Match kube.var.log.containers.nginx*
        delivery_stream <stream_name>
        region <region_name>
[OUTPUT]
        Name cloudwatch
        Match **
        region <region_name>
        log_group_name /eks/eksmatsuri
        log_stream_name fluentbit-kube
        auto_create_group true
```

- 複数の出力先を使い分けたい場合はOUTPUTを複数記載する
- 対象となる log StreamはMatchで記載したTagで指定
  - この場合、nginx podのログはFirehose ・CloudWach Logsに出力される
  - ngix pod以外のログはFirehoseには出力されない

Stream Processorを利用したOutputの分岐例



参考資料

- **Kubernetes Logging powered by AWS for Fluent Bit**

https://aws.amazon.com/jp/blogs/containers/kubernetes-logging-powered-by-aws-for-fluent-bit/

- **Fluent Bit による集中コンテナロギング**

https://aws.amazon.com/jp/blogs/news/centralized-container-logging-fluent-bit/

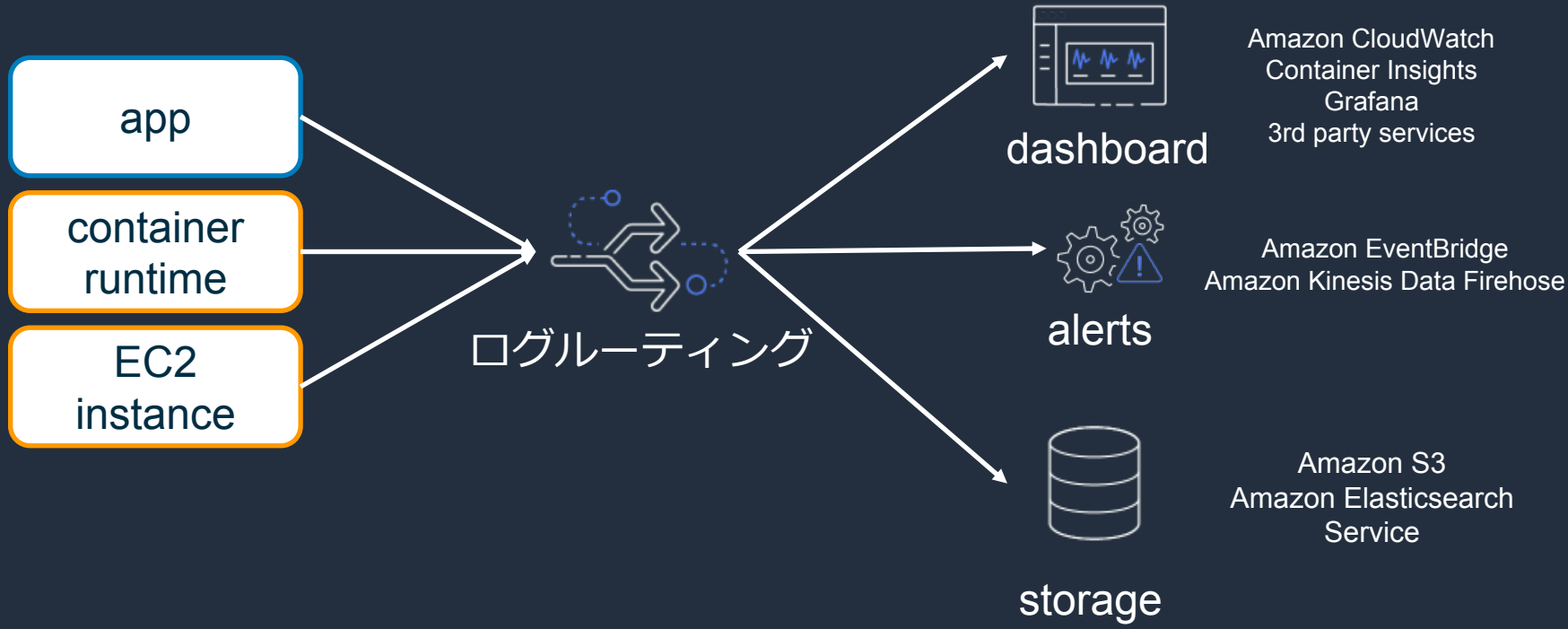
- **Access Logging Made Easy with AWS App Mesh and Fluent Bit**

https://aws.amazon.com/blogs/containers/access-logging-made-easy-with-aws-app-mesh-and-fluent-bit/

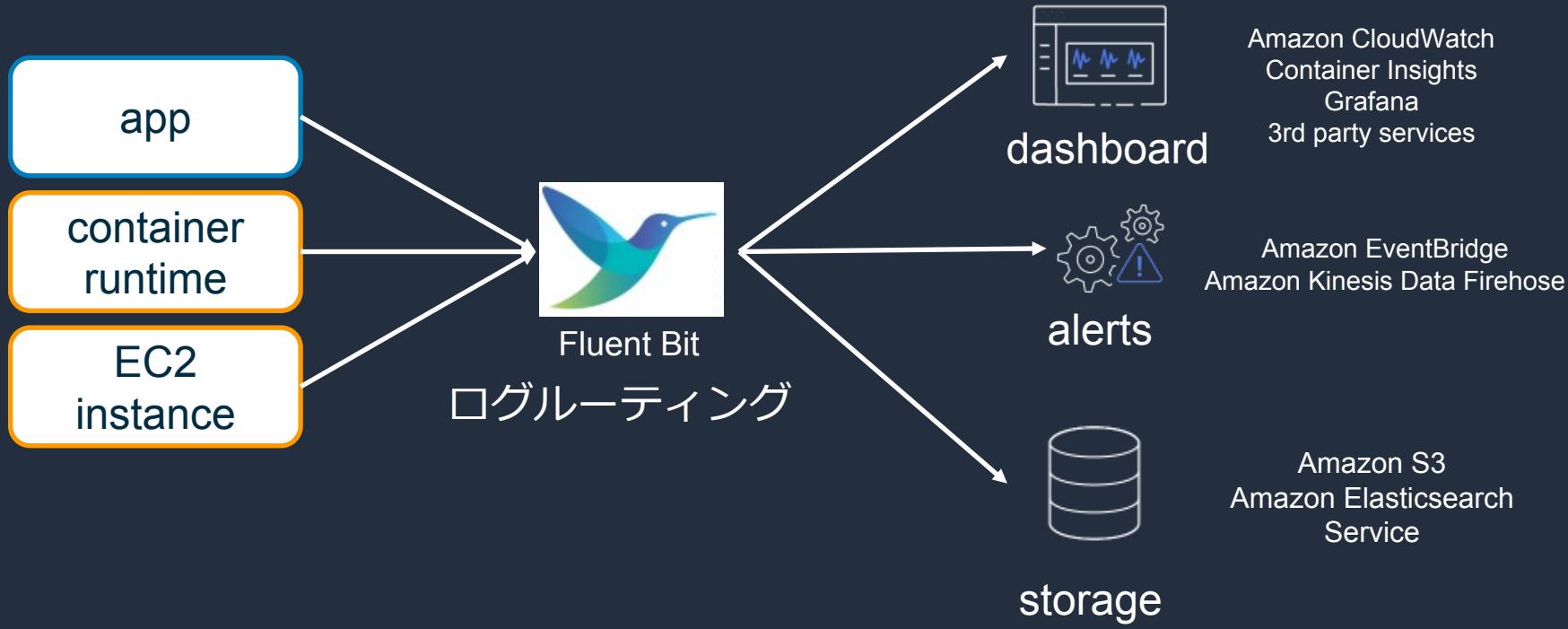
- **Under the hood: FireLens for Amazon ECS Tasks**

https://aws.amazon.com/blogs/containers/under-the-hood-firelens-for-amazon-ecs-tasks/

まとめ



まとめ



Thank you!