

レガシーなEメールサービスを Kubernetes化しました！ (+ホームページサービスもKube化)



2023/01/26

IIJ: 衣笠 茂浩

30th
Anniversary

Ongoing Innovation

私は長年 ISP(Eメール_{+HP他}) システムに携わっています。

このたび、Kubernetesを弊社Eメールサービスで全面採用し、HPサービスもKube化を進めています。

本日はKubernetes導入の「考え方」・「はまりどころ」を伝え、導入障壁が下がれば良いと考えています

お断り： 会社としての見解では無く個人の見解です

- **話者: IJ 衣笠 茂浩**

- ISP担当歴21年

- システム移行: 2桁超

- のべ移行アカウント: 1100万超

- **仕事:**

- メール・動画配信

- アーキテクト

- 管理職 (部長)



2011年 ～ ASP・オンプレ

- 2021年-現在：
メール・HP ASP 導入・
キャッシュDNS ASP開発
- 2020年： 数十万規模メールASP 導入
- 2019年： 数百万規模メールASP 導入
- 2018年：
 - 数十万規模メールASP導入
 - 新メールASP立ち上げ
- 2017年： 数十万規模 導入
- 2015-16年： 数百万規模 導入
- 2013-14年：
 - 数百万規模導入・数十万規模導入
- 2011年： メールASP立ち上げ

2003年 ～ 2010年 オンプレ

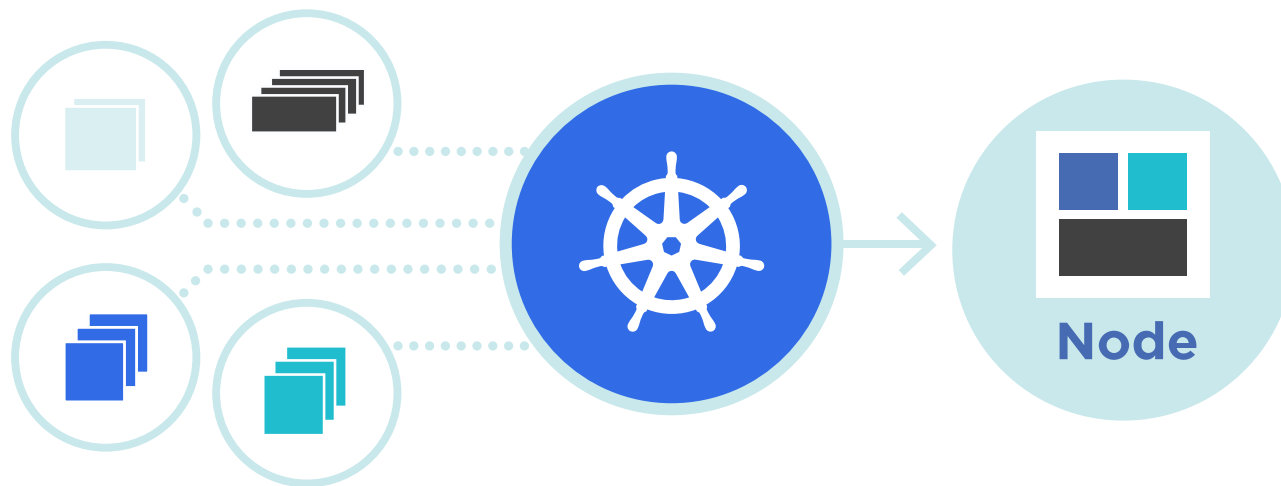
- 2009年-2011年： 数十万規模導入
- 2007年-2008年： 数十万規模導入
- 2006年： 数万規模導入
- 2005年：
 - 数十万規模導入
 - 数万規模導入
- 2004年： 数十万規模導入
- 2003年： 数十万規模導入

ISPのASP化 が加速

Kubernetesとは

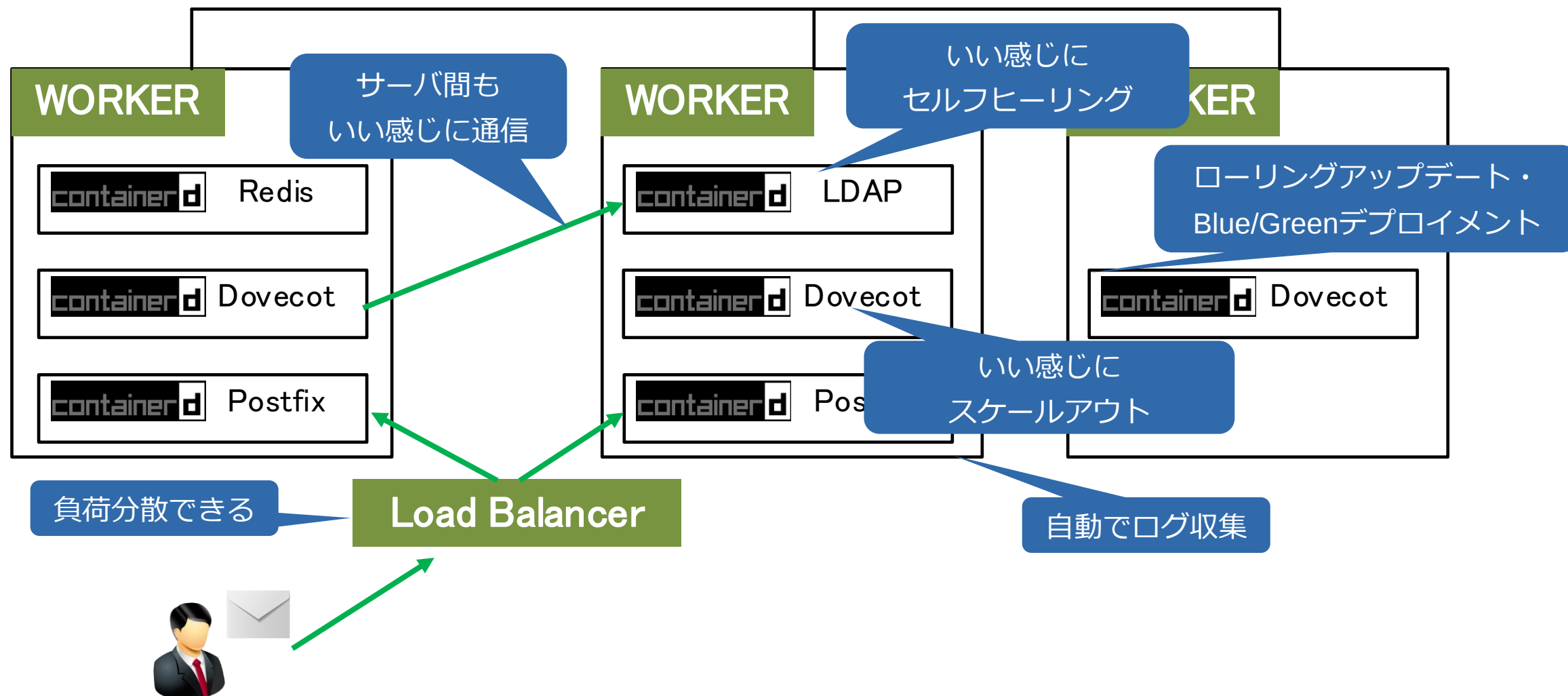
- デプロイやスケーリングを自動化
- コンテナ化されたアプリケーションを管理
- **オープンソース**

自分で調査可能（すごく大切）



コンテナフレームワークです

いい感じなEメールシステム



頑張ることいっぱいでした

1. クラスタ作りこみ

1. クラスタ作成
2. 外部連携 (LB・ストレージ)

2. アプリ作りこみ

1. コンテナ作成
2. デプロイ
3. コントローラ自作

3. モニタリング・ログ

1. モニタリング
2. 監視
3. ログ収集

4. 工夫

1. 性能面
2. 可用性

トピックとはまりどころを伝えます

お断り: できるだけリアルな情報にしていますが、アプリ周りはわかりやすいようにOSS中心に紹介します

Kubernetes クラスタ作りこみ

(オンプレです)

クラスターベース(今回利用した構成の場合)

	プロダクト名	説明
	RockyLinux	Linux OS
	Kubeadm	Kubernetesクラスター環境を作成・管理するツール
	Kubelet	Podやコンテナの起動
	Kubectl	クラスターにアクセスするためのコマンドラインツール
	etcd	クラスターの構成情報・状態を保存するKVS
	Calico	コンテナにネットワークとセキュリティポリシー機能を提供
	F5 Container Ingress Services	F5製連携ソフト。BIG-IP連携で利用。
	Astra Trident	NetApp製連携ソフト。NetApp連携(NFS/iSCSI)で利用。

基本OSS。コード調査可能

基本:

手順を見れば大体できます

- **Kubeadm + calicoでOK**

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

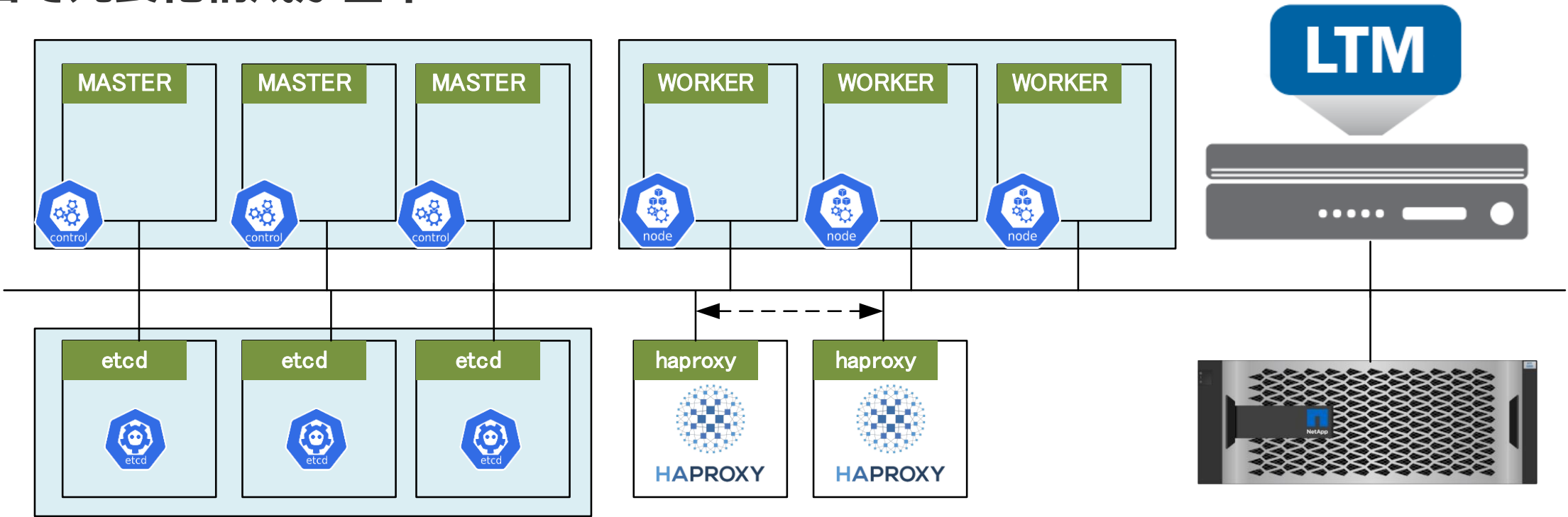
応用:

- **可用性考慮で構成変化有り**

<https://kubernetes.io/docs/setup/#production-environment>

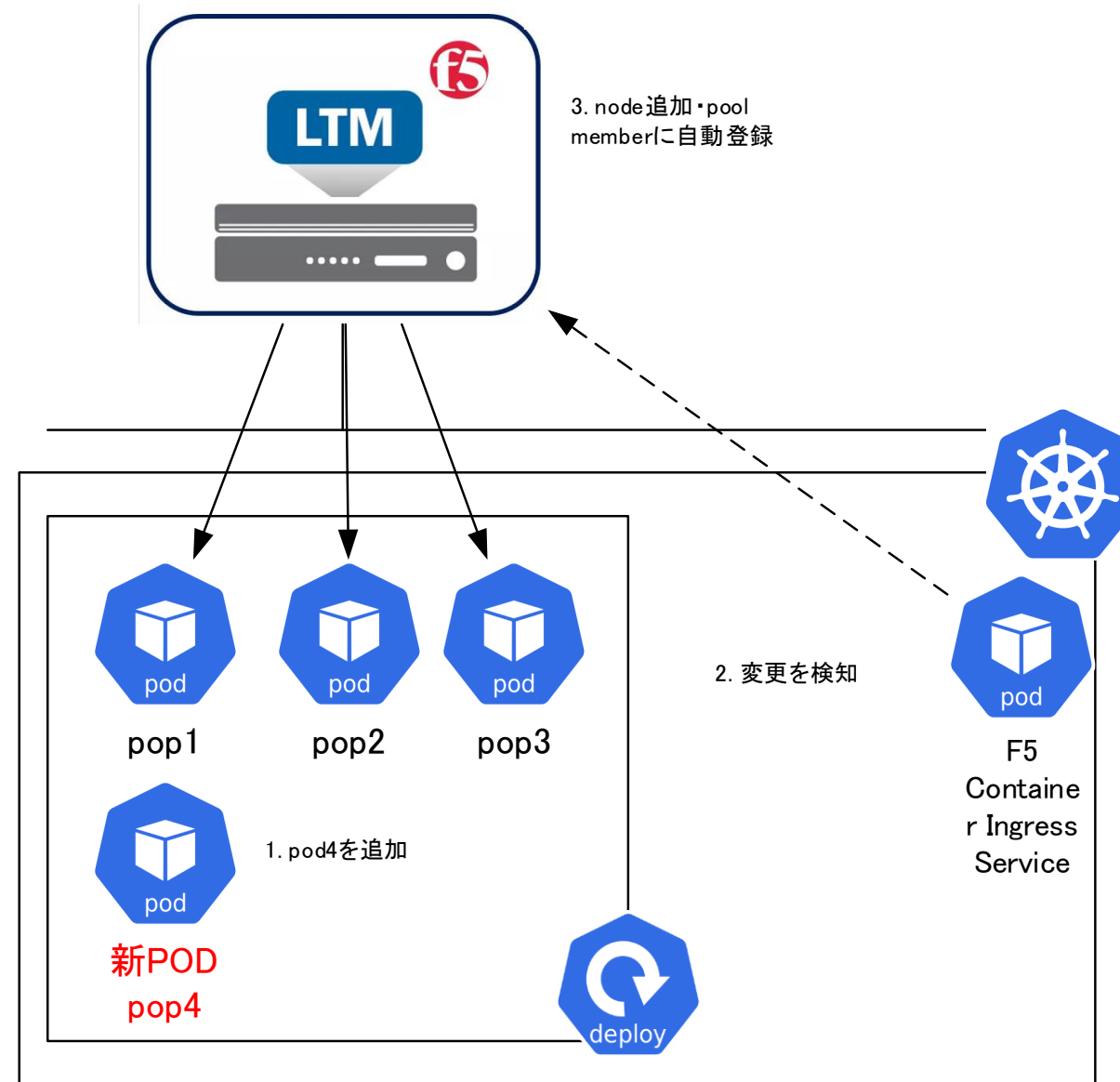
- **Container Network Interface(CNI) ・ Container Storage Interface (CSI) は状況に応じて選択を。**

3台で冗長化構成が基本



名称	説明
MASTER	コントロールプレーン。スケジューリング・イベント検出および応答等を行う。
WORKER	ワーカーノード。POD実行。
etcd	キーバリューストア。クラスタ情報保存に利用。多数決モデル利用。
haproxy	クラスタ外部からMASTER APIへのアクセス冗長化に利用

スケールアウトでLB配下に動的組み込み。IPv6にも対応。



スケールアウトでLB配下に動的組み込み。IPv6にも対応。

3台しかいない

☒	Status	Name	Description	Application	Address	FQDN	Ephemeral	Partition / Path
☐		10.244.139.33			10.244.139.33		No	mail-test
☐		10.244.52.127			10.244.52.127		No	mail-test
☐		10.244.52.67			10.244.52.67		No	mail-test
☐		fd00:1:0:3459:57f7:b97b:403:244f			fd00:1:0:3459:57f7:b97b:403:244f		No	mail-test
☐		fd00:1:0:3459:57f7:b97b:403:247c			fd00:1:0:3459:57f7:b97b:403:247c		No	mail-test
☐		fd00:1:0:6258:3e51:ea64:3455:e2d6			fd00:1:0:6258:3e51:ea64:3455:e2d6		No	mail-test

Enable Disable Force Offline Delete...



```
# kubectl scale statefulset pop --replicas=4
```



4台構成に！
Dual Stackにも対応！

☒	Status	Name	Description	Application	Address	FQDN	Ephemeral	Partition / Path
☐		10.244.139.33			10.244.139.33		No	mail-test
☐		10.244.139.60			10.244.139.60		No	mail-test
☐		10.244.52.127			10.244.52.127		No	mail-test
☐		10.244.52.67			10.244.52.67		No	mail-test
☐		fd00:1:0:3459:57f7:b97b:403:244f			fd00:1:0:3459:57f7:b97b:403:244f		No	mail-test
☐		fd00:1:0:3459:57f7:b97b:403:247c			fd00:1:0:3459:57f7:b97b:403:247c		No	mail-test
☐		fd00:1:0:6258:3e51:ea64:3455:e2c0			fd00:1:0:6258:3e51:ea64:3455:e2c0		No	mail-test
☐		fd00:1:0:6258:3e51:ea64:3455:e2d6			fd00:1:0:6258:3e51:ea64:3455:e2d6		No	mail-test

Enable Disable Force Offline Delete...

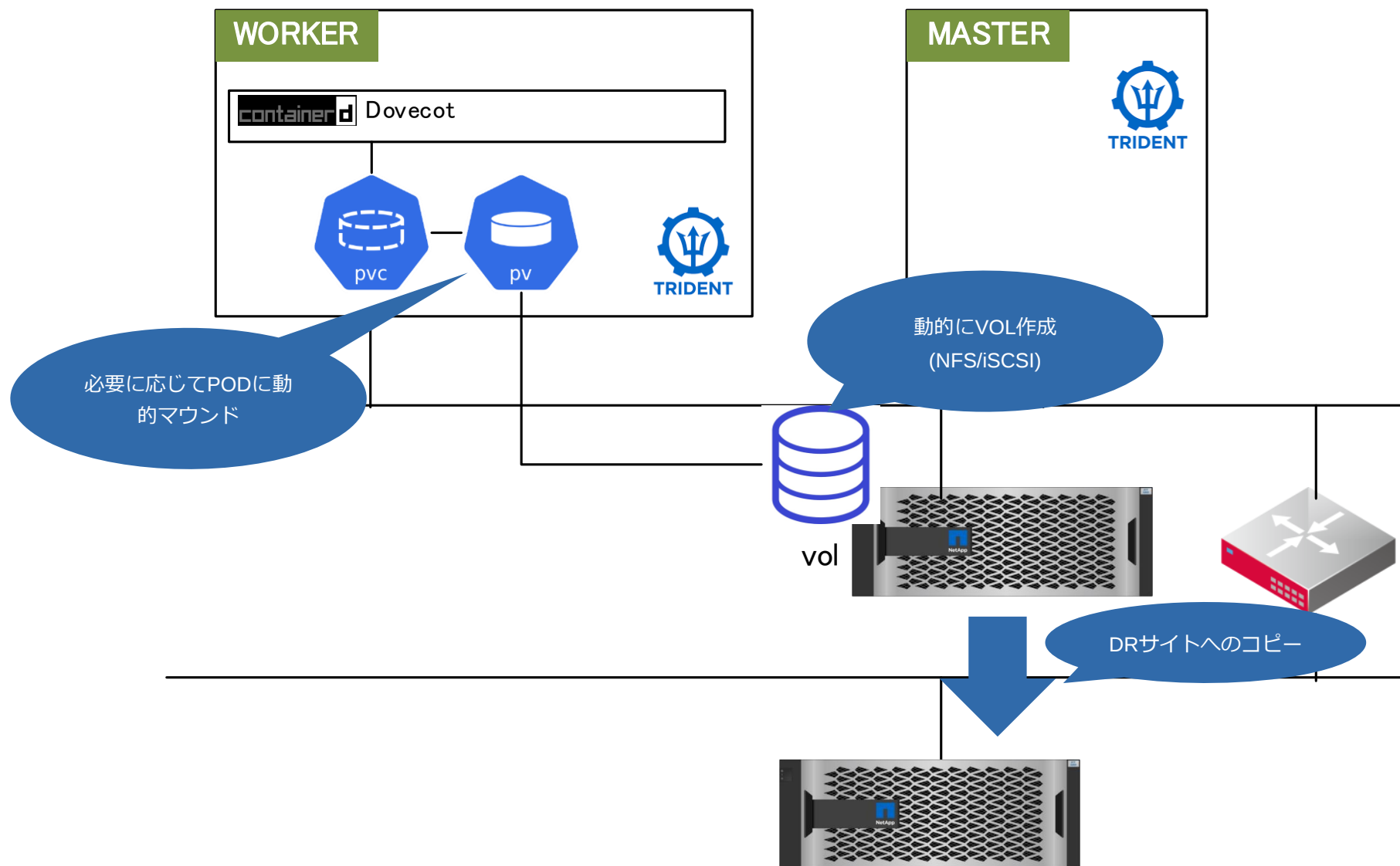
Scale outも簡単(自動でnode/poolに追加)

注意点・はまりどころ

1. BIGIP本体でREST・BGP有効化・AS3導入が必要
2. (重要) 構成変更 (POD増減等)反映まで60秒程度かかる
 - ・パーティションが複数存在する場合はさらに時間がかかる場合あり
3. (重要) BIGIP restjavadが安定しないことがある
 - ・BIGIP本体の最新化・Java HEAP最適化等必要
<https://cdn.f5.com/product/bugtracker/ID769581.html>
<https://support.f5.com/csp/article/K12410040>

上記を踏まないチューニングが必須

ボリューム（NFS・iSCSI）を必要に応じて自動作成。DRにも対応。



注意点・はまりどころはあまり無い（安定してる）。

1. メールキュー等ロック・性能が必要な部分はiSCSIを利用

- オンライン領域解放(discard option)は使わないこと！（プチフリします）
- https://docs.netapp.com/ja-jp/trident-2204/pdfs/fullsite-sidebar/Astra_Trident_22_04_%E3%81%AE%E3%83%89%E3%82%AD%E3%83%A5%E3%83%A1%E3%83%B3%E3%83%88.pdf

2. iSCSI利用時にiSCSIマウントできない(Linux Kernel 不具合)

- - scsi: iscsi: iscsi_tcp: Avoid holding spinlock while calling getpeername() (Chris Leech) [2011470]
- <https://cbs.centos.org/koji/buildinfo?buildID=36146>

回避策: Kernelを最新にあげて回避

Trident連携は安定していました

アプリの作り込み

備考: 皆様にわかりやすいようにOSS系を中心に紹介します

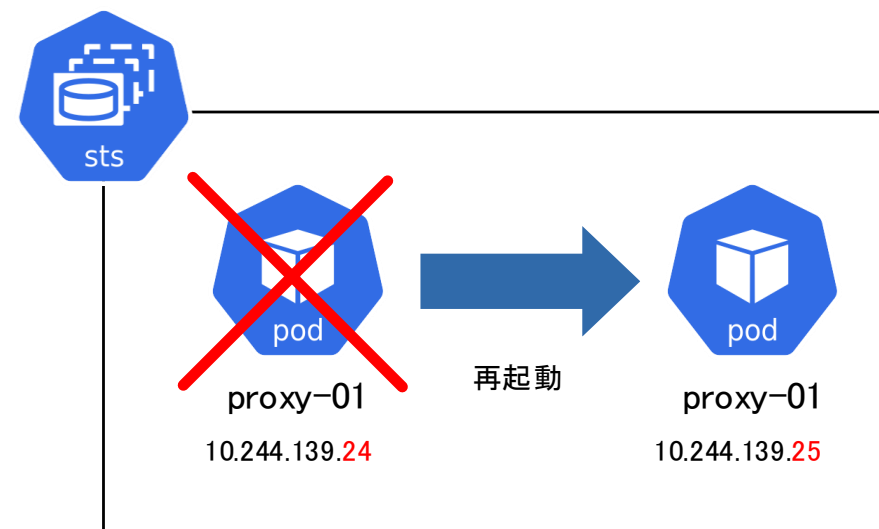
主な特徴

- 全部コンテナにする
 - ミニマムアプリにする

(sample)Postfixでのコンテナプロセス例

```
[root@postfix-6db444b6b4-ccd29 /]# ps axwwuf
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1098  0.1  0.0  22804  3484 pts/0    Ss   10:31   0:00 bash
root      1111  0.0  0.0  55440  3568 pts/0    R+   10:31   0:00  _ ps axwwuf
root         1  0.0  0.0 105232  1996 ?        Ss   Oct24   0:03 /usr/libexec/postfix/master -i
postfix    77  0.0  0.0 121364  1836 ?        S    Oct24   0:00 qmgr -l -t unix -u
postfix    82  0.0  0.0 125524  1140 ?        S    Oct24   0:00 tlsmgr -l -t unix -u
postfix   1093  0.0  0.0 121244  8792 ?        S    10:22   0:00 pickup -l -t unix -u
[root@postfix-6db444b6b4-ccd29 /]#
```

- (最重要)通信は名前解決のみ
 - IPアドレスは変わるもの
 - Kubernetesが提供しない機能は作りこむ
 - クラスタリング(Dovecot Director)
 - MariaDB (Galera Cluster)
- 他



POD再起動でIPは変わる！

Dockerfile(sample: postfix)

```
FROM rockylinux/rockylinux:8.6
RUN dnf -y install postfix
RUN echo "maillog_file = /dev/stdout" >> /etc/postfix/main.cf
CMD ["/usr/sbin/postfix","start-fg"]
```

ポイント

- ログ: 標準出力
- 起動: フォアグラウンド
- 開発・デバッグツール: ご自由に

メールアプリコンテナ

作成コンテナ（これ以外にも多数あり）

#	アプリケーション	ログ出力	フォアグラウンド起動
1	postfix	○	○
2	Dovecot	○	○
3	Halon MTA	○	○
4	Cloudmark Authority Server	○	○
5	TwoFive Safe25	○	○
6	Vade MTA	○	○
7	Zimbra	○	○
8	Haproxy	○	○
9	Fluentd	○	○
10	Fluent-bit	○	○
11	apache	○	○
12	Appsuite	○	○
13	Keycloak	○	○
14	Nginx	○	○
15	Gunicorn	○	○
16	Openldap	○	○
17	Mailman3	○	○
18	Openresty	○	○
19	ProFTPD	○	○

一つ一つは大きくないが、全体で見ると作業量が多い

- **docker hub等のサンプルを改造**
- **下調べと十分な検証が必要**
 - 1. ログ出力
 - stdout/stderr or syslog
 - 2. モニタリング対応(後述)
 - exporter必要有無
 - 3. 同一コンテナで複数プロセス動作要否
 - rsyslog・外部exporter同居時はsupervisordが必要等

モニタリング連携まで揃えると作り込みに時間がかかる

sample: Dovecot Directorのコンテナ起動例

```
# ps axwwuf
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      3906217 0.0  0.0  23112  3880 pts/0    Ss   10:39   0:00 bash
root      3906804 0.0  0.0  55472  3672 pts/0    R+   10:43   0:00 ¥_ ps axwwuf
root       1  0.1  0.0 109548 17216 ?        Ss   Aug08 146:31 /usr/bin/python3.6 /usr/bin/supervisord -c
/etc/supervisord.conf
root       9  0.0  0.0  16020   852 ?        S    Aug08   0:00 /bin/bash /boot-dovecot.sh
root      20  0.0  0.0  52944  1612 ?        S    Aug08 18:15 ¥_ /usr/sbin/dovecot -F
dovenull  23  0.0  0.0  32892  7924 ?        S    Aug08   3:18 ¥_ dovecot/pop3-login
dovenull  24  0.0  0.0  29472  3344 ?        S    Aug08   8:20 ¥_ dovecot/imap-login
dovecot   25  0.0  0.0  10384   764 ?        S    Aug08   1:18 ¥_ dovecot/anvil [0]
root      26  0.0  0.0  11156  1340 ?        S    Aug08 10:43 ¥_ dovecot/log
dovenull  27  0.0  0.0  35136  8816 ?        S    Aug08   4:26 ¥_ dovecot/pop3-login
dovenull  28  0.0  0.0  33872  7944 ?        S    Aug08   3:44 ¥_ dovecot/pop3-login
dovenull  29  0.0  0.0  37912 10968 ?        S    Aug08   5:33 ¥_ dovecot/pop3-login
dovenull  30  0.0  0.0  30636  3624 ?        S    Aug08   8:48 ¥_ dovecot/imap-login
dovenull  31  0.0  0.0  30808  4232 ?        S    Aug08   9:23 ¥_ dovecot/imap-login
dovenull  32  0.0  0.0  31184  5724 ?        S    Aug08 10:20 ¥_ dovecot/imap-login
root      33  0.0  0.0  29556  3868 ?        S    Aug08 59:32 ¥_ dovecot/config
dovecot   34  0.0  0.0  15872  2872 ?        S    Aug08 35:44 ¥_ dovecot/stats [11 connections]
dovecot   36  0.0  0.0  18064  2248 ?        S    Aug08 35:50 ¥_ dovecot/director [121 users, 0+0 req/s, 0+0
kB/s]
dovecot   250  0.0  0.0  10200   520 ?        S    Aug08 251:20 ¥_ dovecot/ipc [9 connections]
root      10  0.1  0.0  530276 10200 ?        S    Aug08 251:20 /usr/bin/python3.6 /usr/sbin/healthcheck.py
root      11  0.0  0.0  199228  1536 ?        Sl   Aug08 46:34 /usr/sbin/rsyslogd -n
```

dovecot・ヘルスチェック・rsyslog起
動のためにsupervisord利用

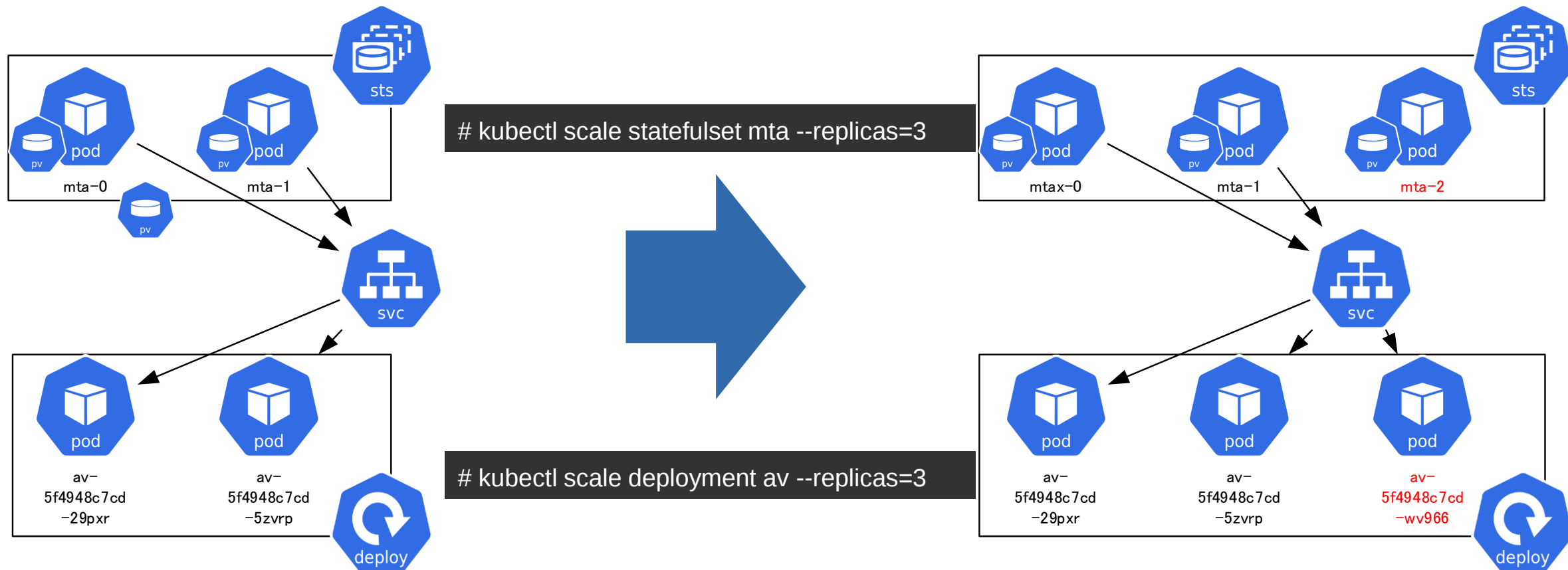
Dovecot フォアグラウンド起動(-F)
ログは/dev/stdoutに出力

rsyslogは/dev/stdoutにログを出力

ステートレス・ステートフルでデプロイプランを検討

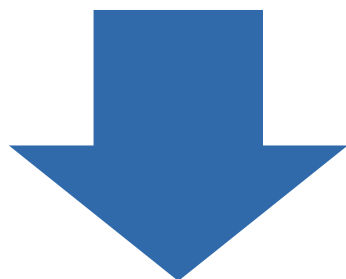
- **ステートレス: Deployment**
 - ステートレスアプリケーションの管理に利用
 - プロキシ・Webサーバ・アンチウイルス・アンチスパム等にご利用
- **ステートフル: Statefulset**
 - ステートフルなアプリケーションの管理に利用
 - メールボックス・MTA・LDAP等ストレージを利用する物にご利用

必要に応じてスケールアウト



動的に増減できるのがKubernetesの醍醐味！

大量メール着信に備えて2倍にスケールアウト！



2倍きっちり打ち込まれる・・・

教訓：大量着信には流量制御で勝負すべし

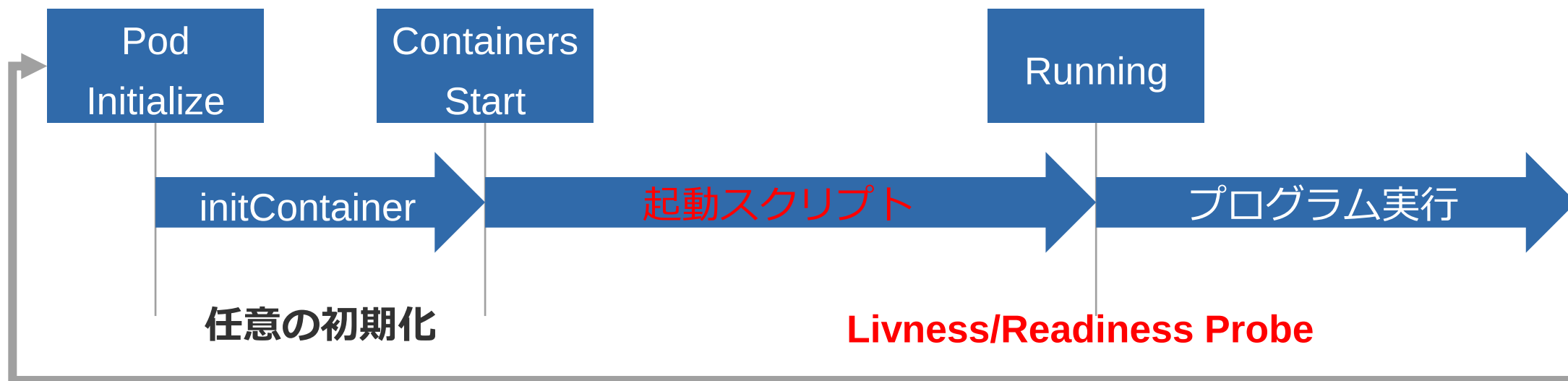
Kubernetes化には以下強化が必須

- **起動・停止スクリプト**
 - VMの起動・停止とは前提が異なる
 - gracefulな停止が必要
- **適切なヘルスチェックが肝**
 - POD生死判定: LivenessProbe
 - POD RUNNING判定: ReadinessProbe

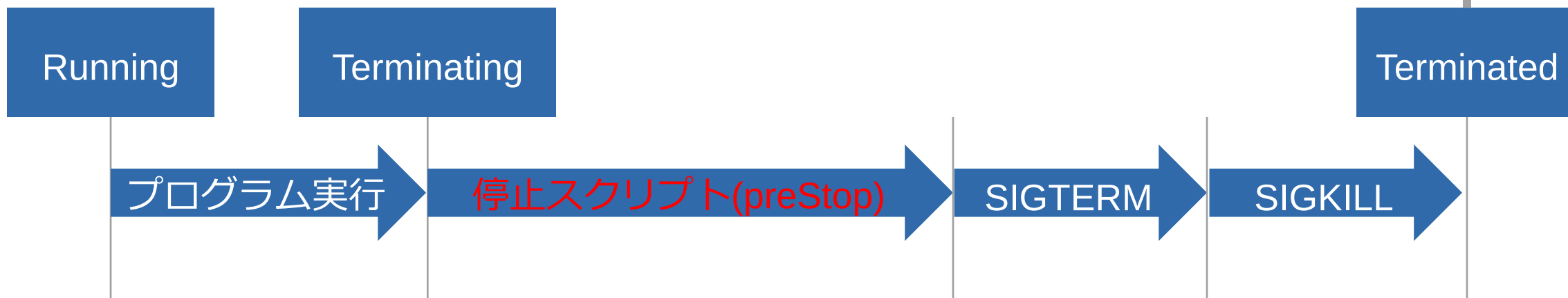
ソフトウェアによっては
かなりの作り込むになるケースも

セルフヒーリングに最も重要なポイント！

POD起動時のライフサイクル



POD終了時のライフサイクル



POD初期化含めた起動スクリプトが必須

- 例: メールキューディレクトリ作成

```
for D in /var/spool/mailq /var/spool/mailq/queue /var/spool/mailq/tmp
do
    mkdir -p $D
    chown smtpd:smtpd $D
    chmod 750 $D
done
```

- VMでは既に終わってる前提だった初期化を敢えて意識した実装が必要

独自起動スクリプトが必要なケース多し

停止スクリプト(preStop)は以下で実行される

- POD Terminating時
- LivenessProbe 失敗

停止スクリプト例

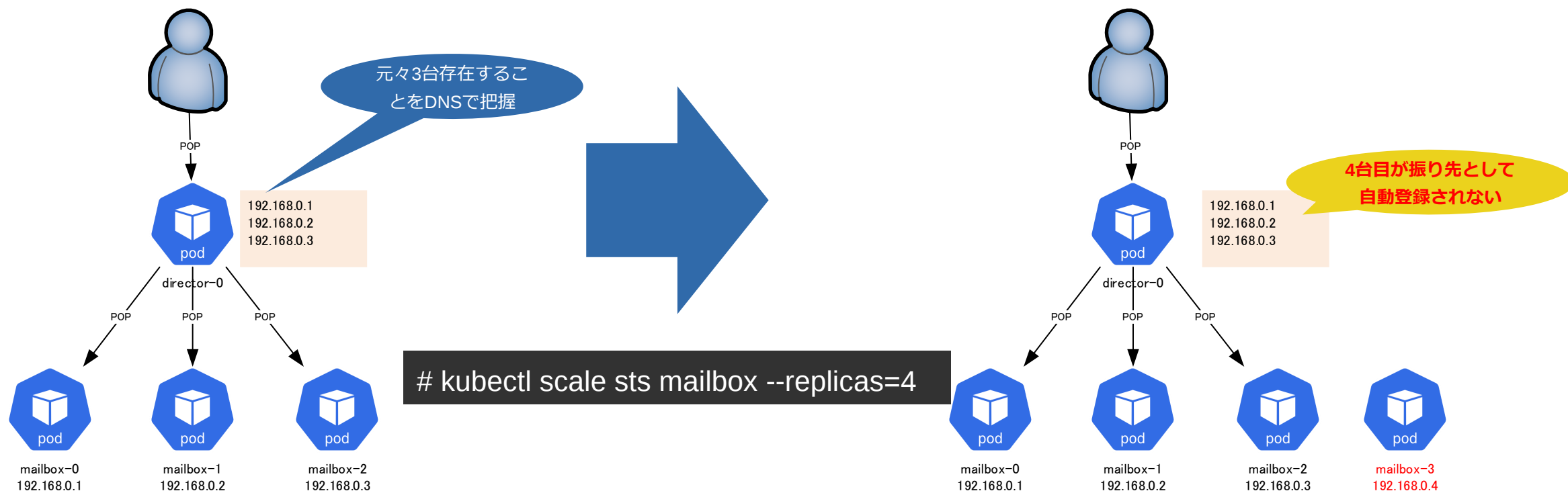
- メールキュー強制吐き出し
 - ユーザコネクション切断待ち & 切断
 - クラスタリングからの自分自身の削除
- 他

クラスタリングの作り込みが
複雑になるケースあり

メンテナンス・セルフヒーリングには
graceful停止が必須

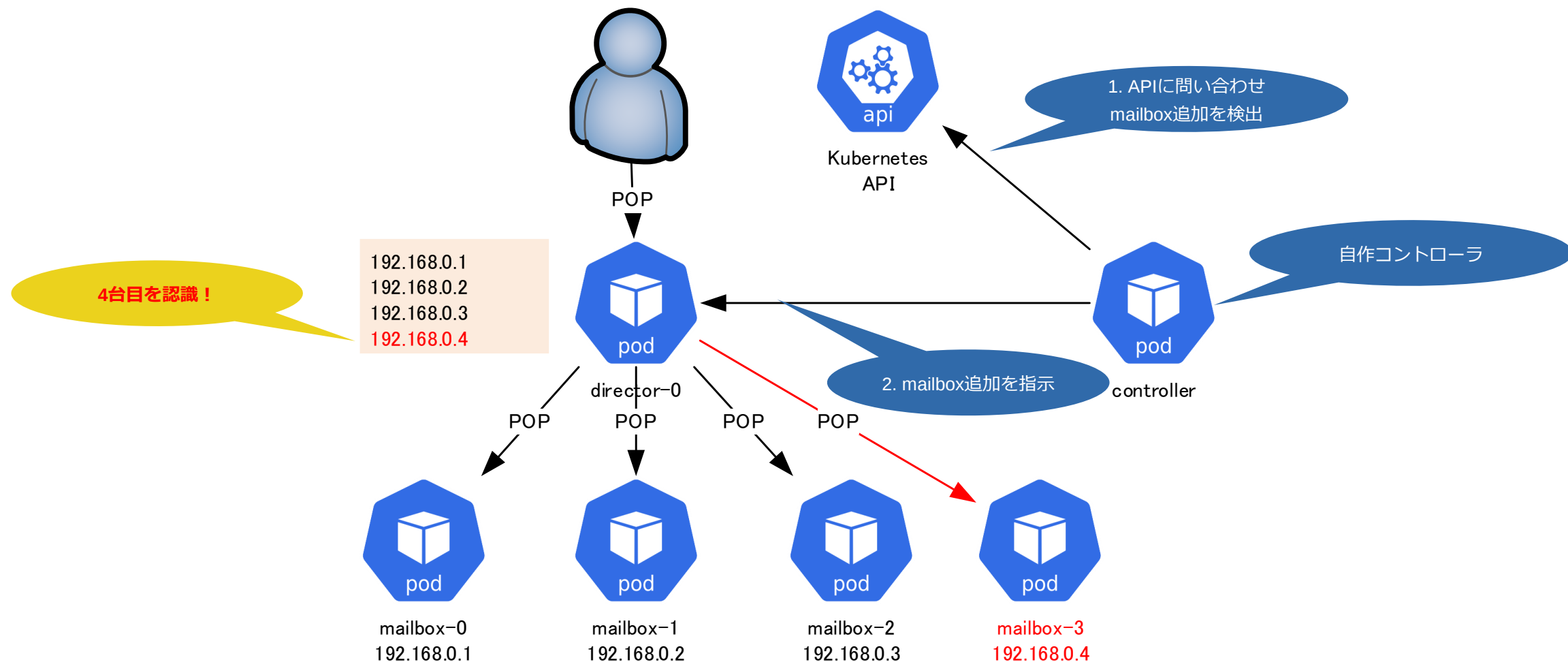
アプリケーションクラスタリングは自作

sample: Dovecot Directorの場合



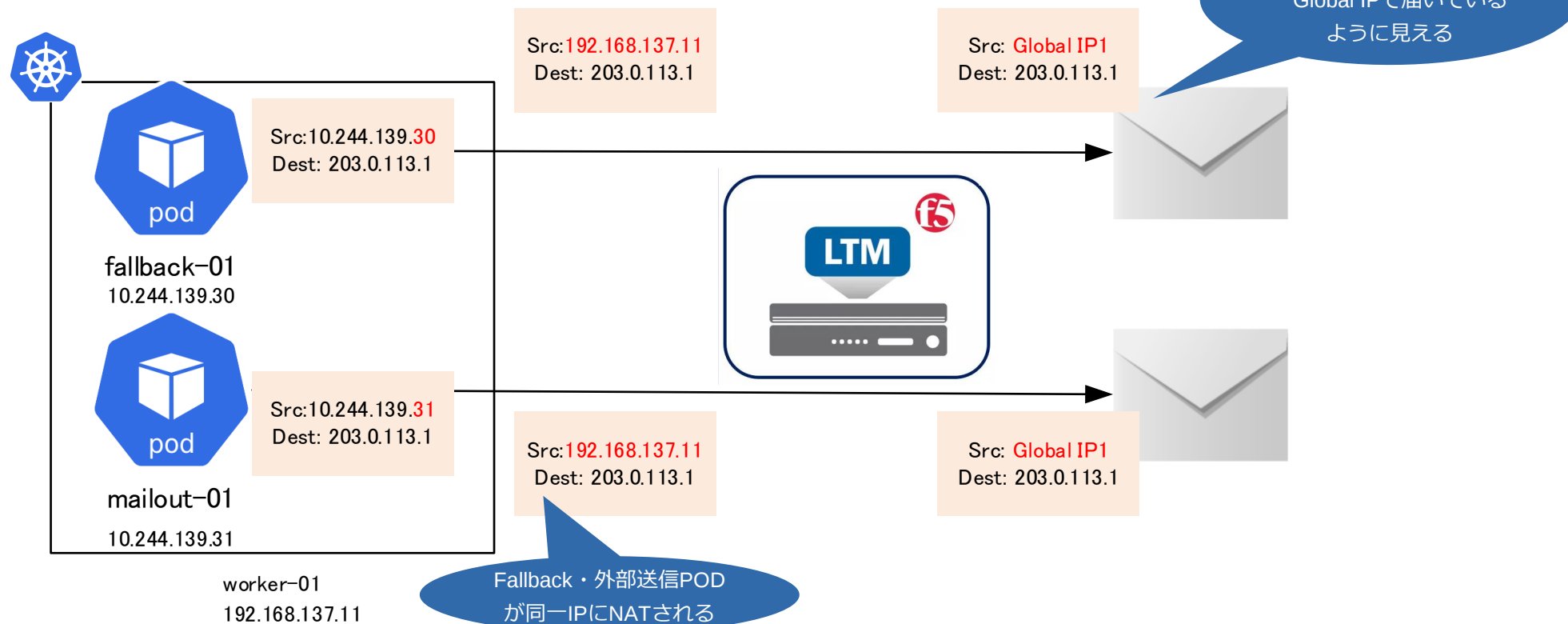
Directorは起動時のみ名前解決するため起動後のノード増減を把握できない

POD増減に応じて動作するコントローラが必要



似たような物はこちら（未検証です）
<https://github.com/yartu/yartu-dovemon/>

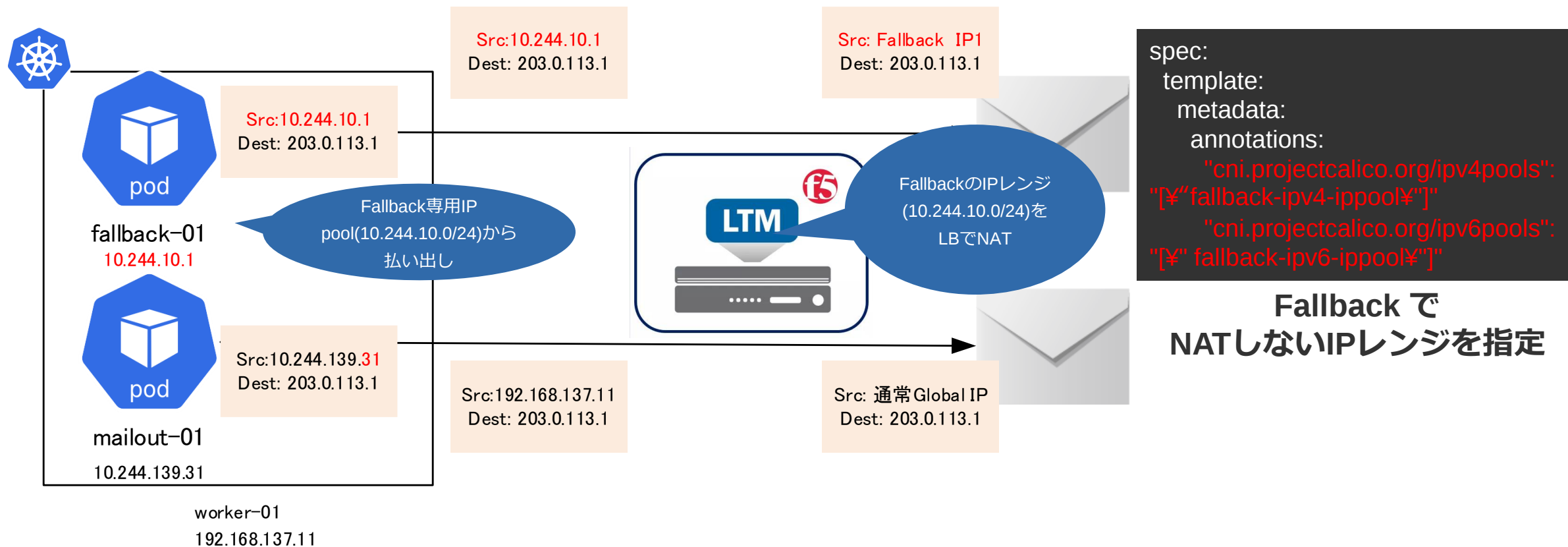
Kubernetesではノード外通信は基本ノード IPでNATされる



例: fallback・通常配送で同じsrc IP(192.168.137.11)にNATされるため、出口IP使い分けできないケース

Fallback・通常送信が同一グローバルIPとなり、IPの送信評価が低下するリスク

ノード出口でNATせず、ノード外(LB・FW)でNATすることで出口振り分け実施

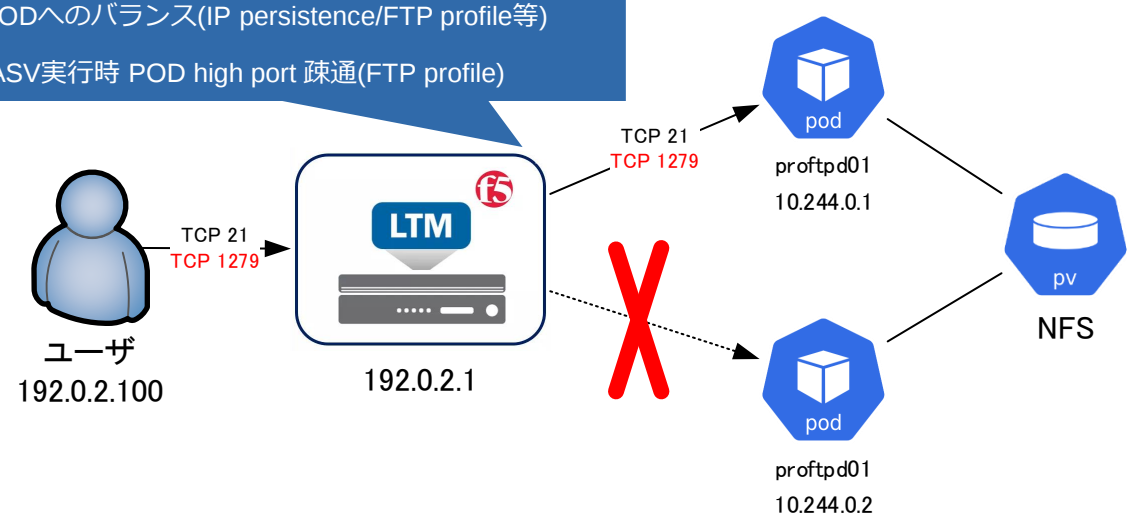


IPv4・IPv6DualStack可能です

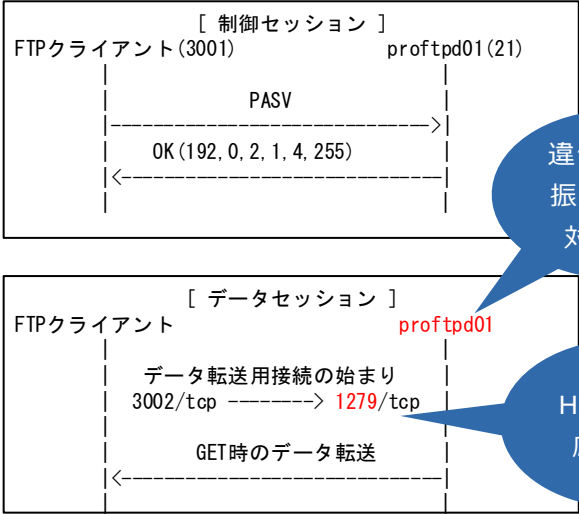
<https://projectcalico.docs.tigera.io/networking/workloads-outside-cluster>

負荷分散装置でのIP Persistence/FTP用バランシングが必要

- 1. 同一PODへのバランス(IP persistence/FTP profile等)
- 2. PASV実行時 POD high port 疎通(FTP profile)



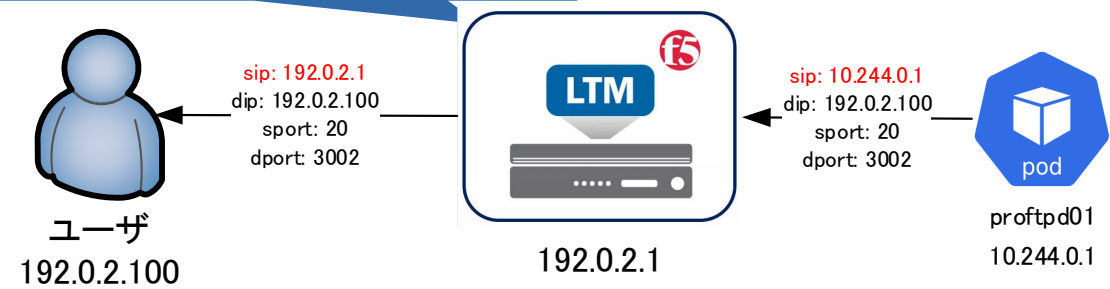
PASVコマンド



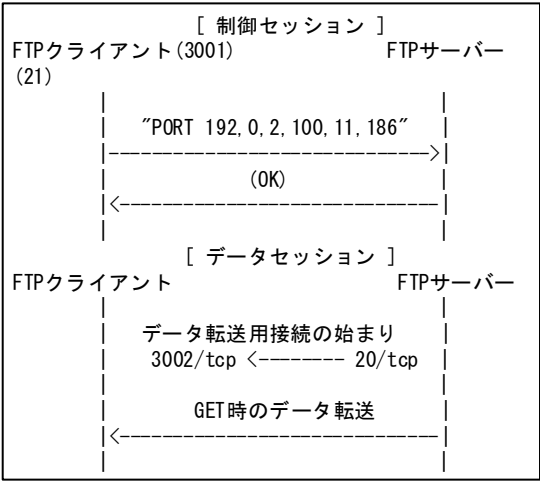
違うPODに振られない対応必須

High port対応も必要

- 1. PORT実行時 VIP(192.0.2.1)へのNAT(FTP profile)



PORTコマンド



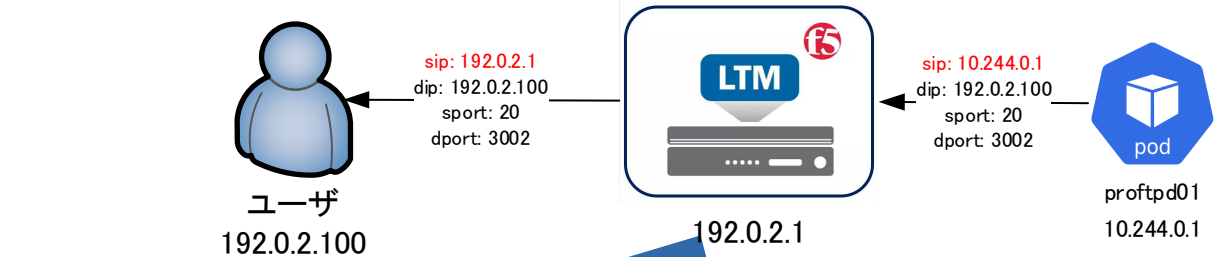
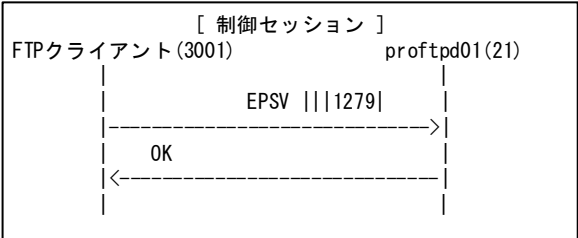
経路暗号化により、EPSV(high port対応)・EPRTコマンド NAT考慮が必要

(ftp profileは利用できないため、自前で対応する必要あり)



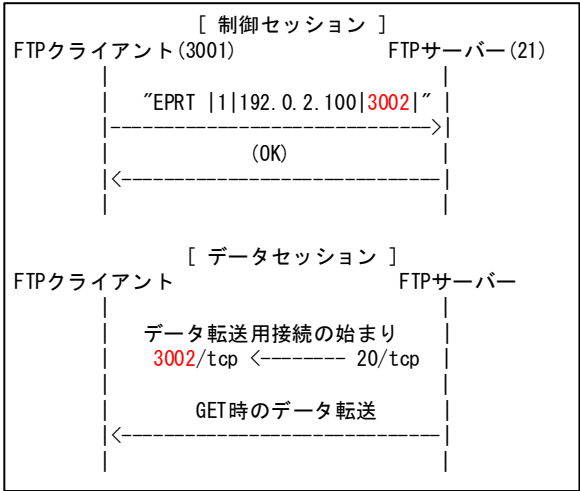
- 1. 同一PODへのバランス(IP persistence)
- 2. PASV実行時 POD high port 疎通(wildcard VIP)

EPSVコマンド



- 1. 同一PODへのバランス(IP persistence)
- 2. EPRT実行時 VIP(192.0.2.1)でのNAT
(NATを明確に指定するため前述のPOD IPレンジ固定が必要)

EPRTコマンド



モニタリング・ログ収集

主な特徴

- **PULL型監視**
 - 定期的にHTTP経由でmetricsを取得し評価
- **監視対象（POD）が動的に増減**
 - Service discover必須
- **ノード・アプリ監視 それぞれ必要**
 - ノード監視: 標準でほぼデータ取得可能
 - アプリ監視: HTTP経由でOpenMetricsに対応していればOK
 - メールアプリは基本非対応(Dovecot・Halonぐらいか?)

Prometheus互換。安定稼働中。



- Highest Ingestion Rate
- Fastest Query Performance
- **Smallest Disk Storage Size**
- Lowest Memory Usage
- Long-term Storage for Metrics
- Highly Scalable
- Cloud Readiness
- Simple Set-up & Operation

<https://victoriametrics.com/>
<https://docs.victoriametrics.com/>

Clients

vmselect fully supports PromQL and can be used as Prometheus datasource in Grafana

Stateless

vmselect fetches and merges data from vmstorage during queries

Stateful

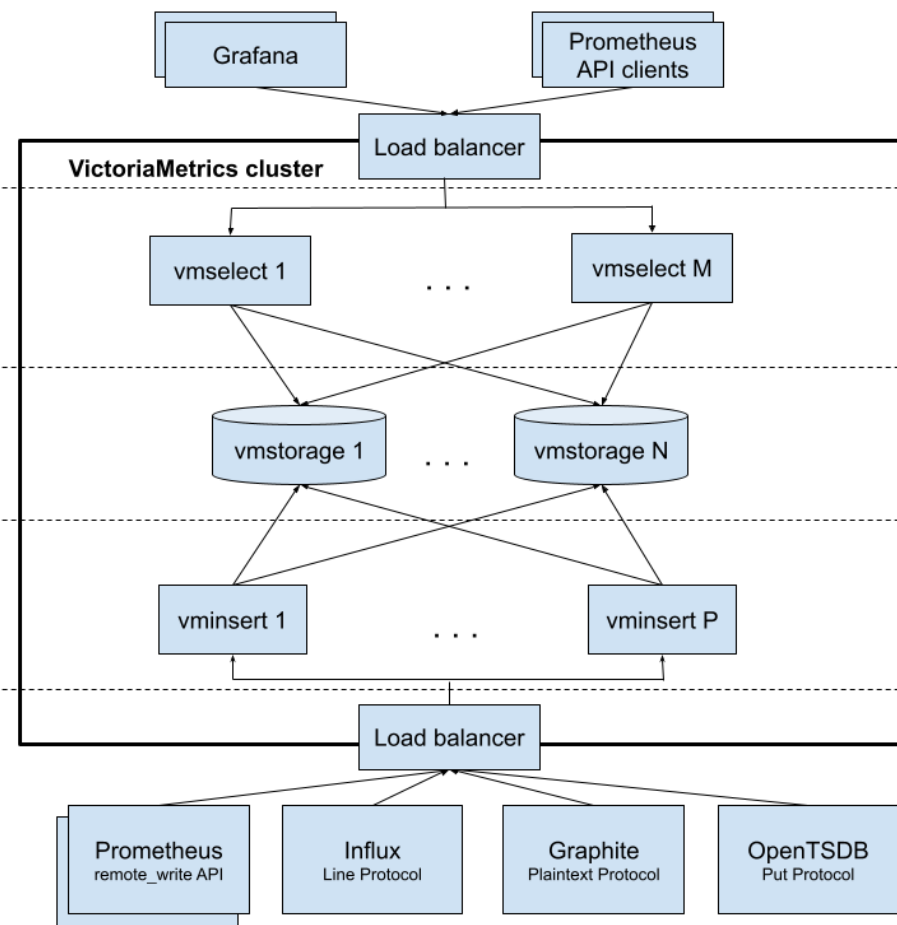
vmstorage stores time series data

Stateless

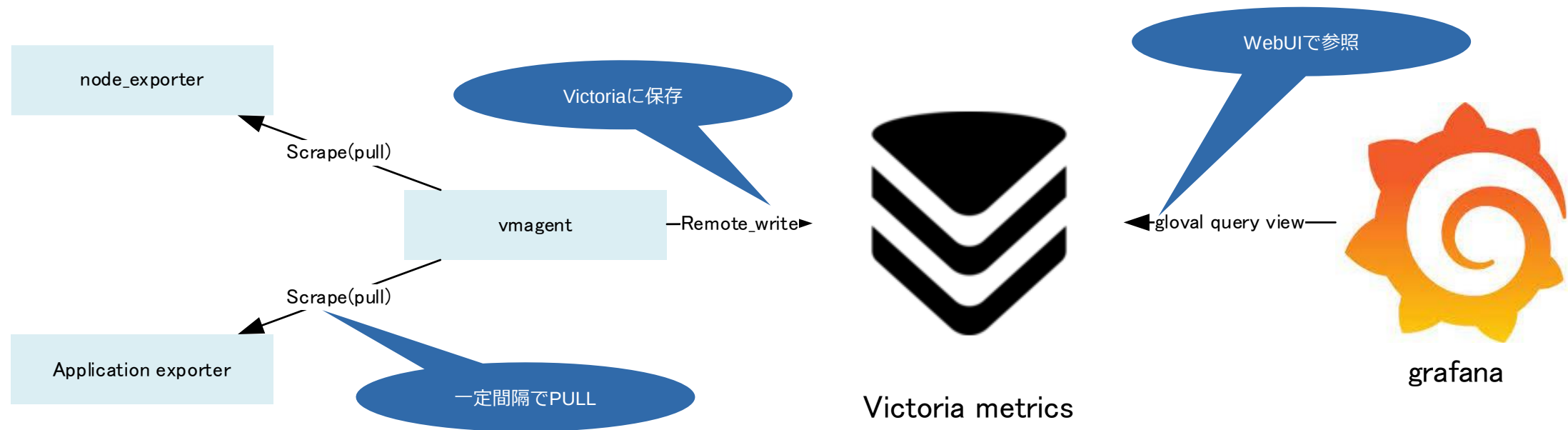
vminsert spreads time series across available vmstorage nodes

Writers

*Multiple Prometheus instances may write data to VictoriaMetrics cluster
There is support for other ingestion protocols*



VictoriaMetrics + GrafanaでPULL型監視



一定間隔でHTTPでmetricsデータ取得
VictoriaMetricsにデータ投入する

アプリ監視対応状況

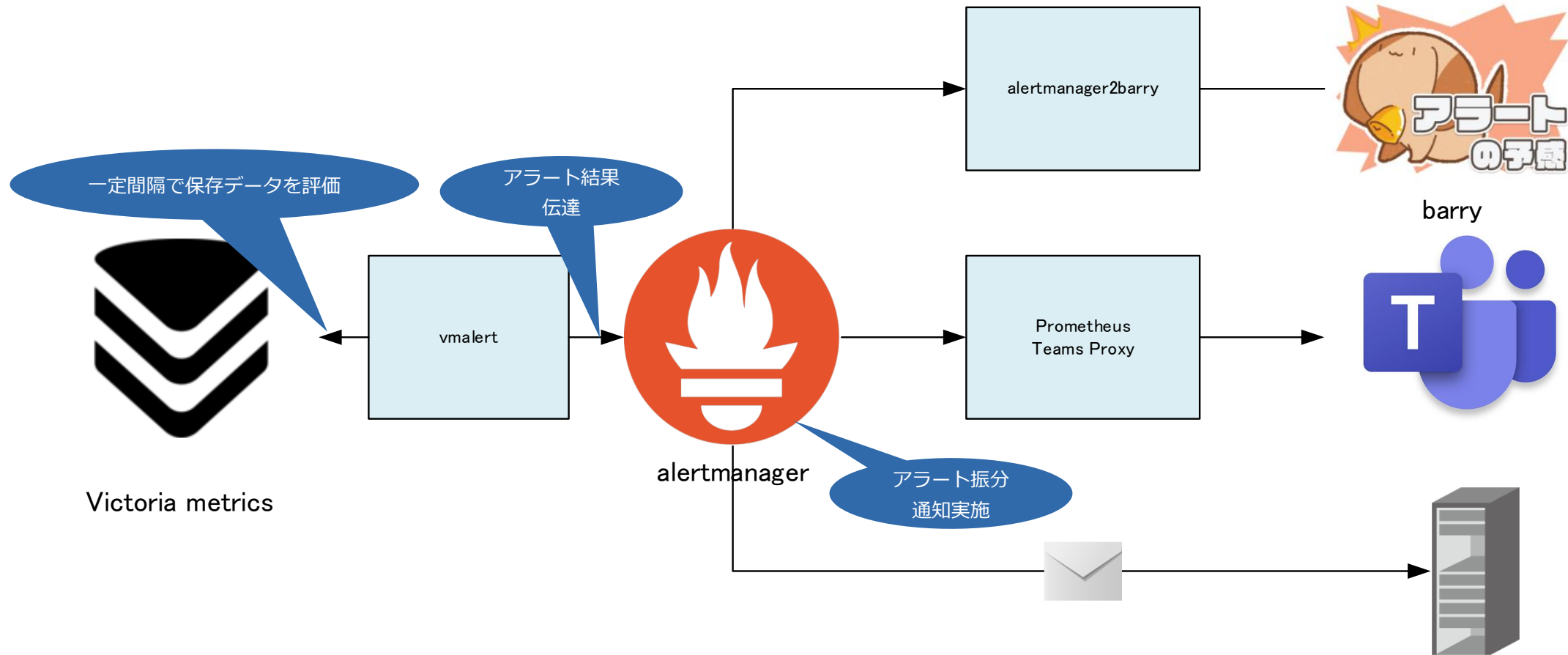
検証結果			
#	アプリケーション	監視ポイント	Grafanaダッシュボード
1	postfix	○(外部ツール)	○
2	Dovecot	◎(標準提供)	●(自作)
3	Halon MTA	◎(標準提供)	●(自作)
4	Cloudmark Authority Server	●(自作)	●(自作)
5	TwoFive Safe25	●(自作)	●(自作)
6	Zimbra	◎(標準提供) * 最新	●(自作)
7	Haproxy	◎(標準提供)	○
8	Fluentd	◎(標準提供)	○
9	Fluent-bit	◎(標準提供)	○
10	apache	○(外部ツール)	○
11	Appsuite	◎(標準提供)	○
12	Openresty	○(lua拡張)	○
13	ProFTPD	○(外部ツール)	○

望む監視のために自作するケースも多々あり
メールアプリのダッシュボードは基本自作

Grafana dashboardで可視化



監視はAlertmanager + barry(IIJ 内製 インシデント管理アプリ) + 他



Barryとは:

<https://www.iij.ad.jp/dev/report/iir/051/03.html>

<https://eng-blog.iij.ad.jp/archives/4488>

監視テンプレート有り

#	アプリケーション	監視テンプレート
1	Kubernetes	https://awesome-prometheus-alerts.grep.to/rules.html
2	ホスト	https://awesome-prometheus-alerts.grep.to/rules.html
3	MySQL	https://awesome-prometheus-alerts.grep.to/rules.html
4	Redis	https://awesome-prometheus-alerts.grep.to/rules.html

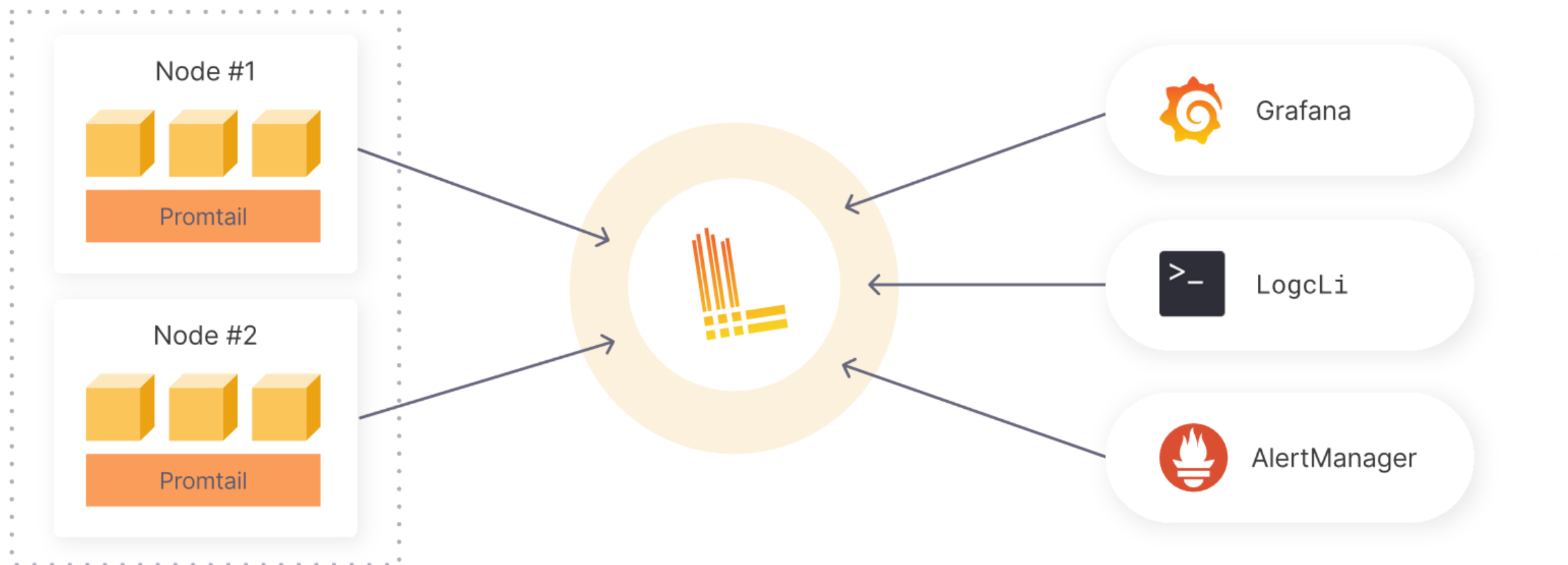
これで事足りる

Eメールアプリは監視テンプレート無し

#	アプリケーション	その他
1	postfix	ポート監視 自作SMTP・POP・IMAP・HTTPトランザクション等 キュー監視 リソース監視
2	Dovecot	
3	Halon MTA	
4	Cloudmark Authority Server	
5	TwoFive Safe25	
6	Zimbra	

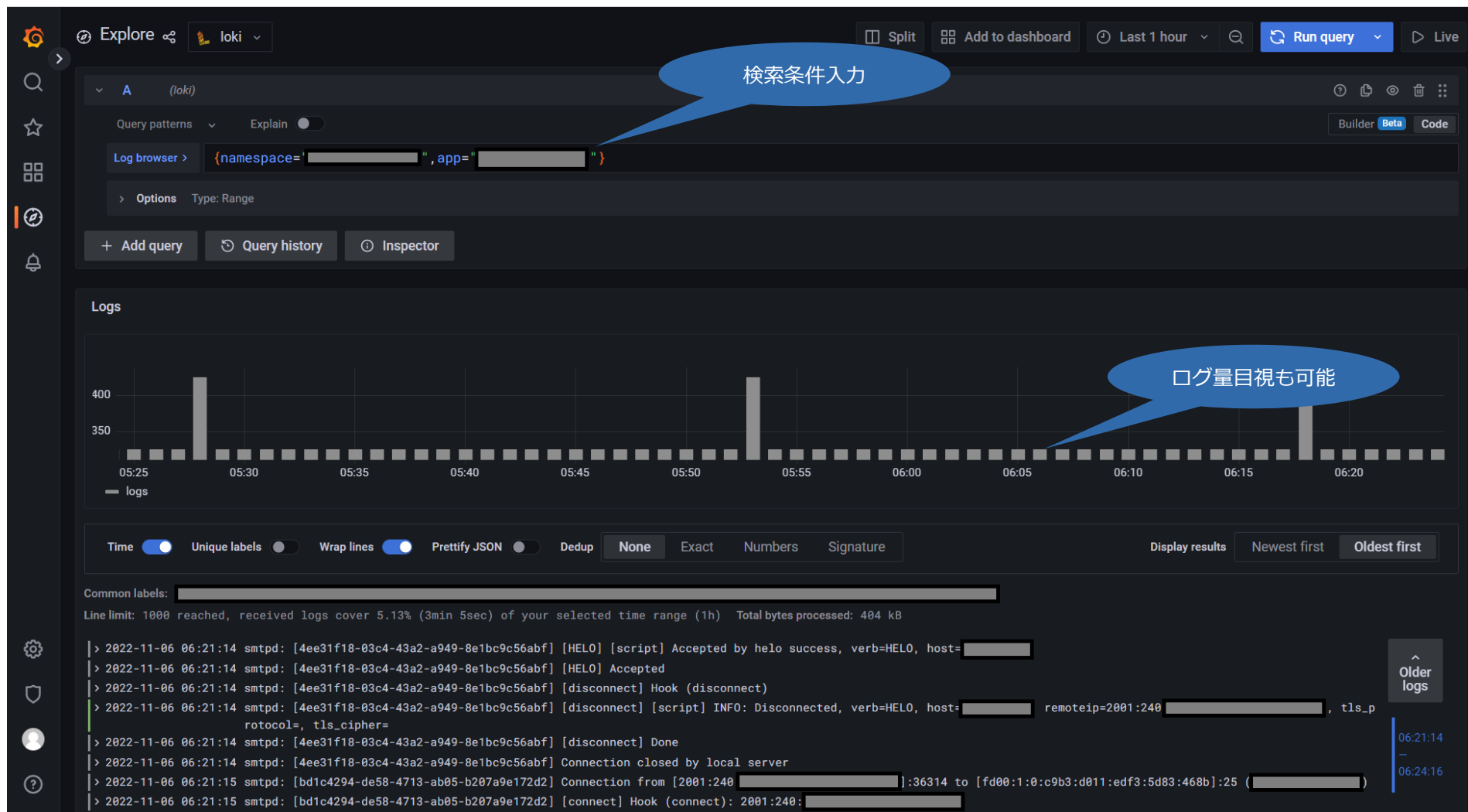
メールアプリ監視は基本自作
Kubernetes側でのカバー範囲多し(PODダウン・リソース監視・ヘルスチェック失敗等)

コンテナログ収集はlokiで集約



<https://grafana.com/oss/loki/>

Grafanaで串刺し検索



Kubeは統合ログ検索必須 (Loki・Splunk等必須)

LokiのログはS3互換ストレージに保管可能

- IJオブジェクトストレージサービス
 - minio
 - AWS S3
 - Scality S3
- 他

MINIO



SCALITY

工夫したところ

- デフォルト設定では名前解決「しすぎ」です！

ndots: 1 は必須(と信じてる)

POD デフォルトの/etc/resolv.conf(ndots: 5)

```
search default.svc.cluster.local svc.cluster.local cluster.local test.example.com example.com
nameserver 10.96.0.10
options ndots:5
```

ndotsとは

```
$ man resolv.conf
```

```
....
```

```
ndots:n
```

Sets a threshold for the number of dots which must appear in a name given to `res_query(3)` (see `resolver(3)`) before an initial absolute query will be made. The default for `n` is 1, meaning that if there are any dots in a name, the name will be tried first as an absolute name before any search list elements are appended to it. The value for this option is silently capped to 15.

ndots:

- `/etc/resolv.conf`で定義したドメイン補完 動作有無を「ドット」の数で判断
- **OS標準 ndots: 1**
 - 「ドット」が1つ含まれていれば補完しない

ndots:5(Kubernetes default) でij.ad.jpにメールを投げると . . .

1. omgi.ij.ad.jp.default.svc.cluster.local (A)
2. omgi.ij.ad.jp.default.svc.cluster.local (AAAA)
3. omgi.ij.ad.jp.svc.cluster.local (A)
4. omgi.ij.ad.jp.svc.cluster.local (AAAA)
5. omgi.ij.ad.jp.cluster.local (A)
6. omgi.ij.ad.jp.cluster.local (AAAA)
7. omgi.ij.ad.jp.test.example.com (A)
8. omgi.ij.ad.jp.test.example.com (AAAA)
9. omgi.ij.ad.jp.example.com (A)
10. omgi.ij.ad.jp.example.com (AAAA)
11. omgi.ij.ad.jp(A)
12. omgi.ij.ad.jp(AAAA)

* omgi.ij.ad.jp: ij.ad.jpのMXレコード

マジでやばいです

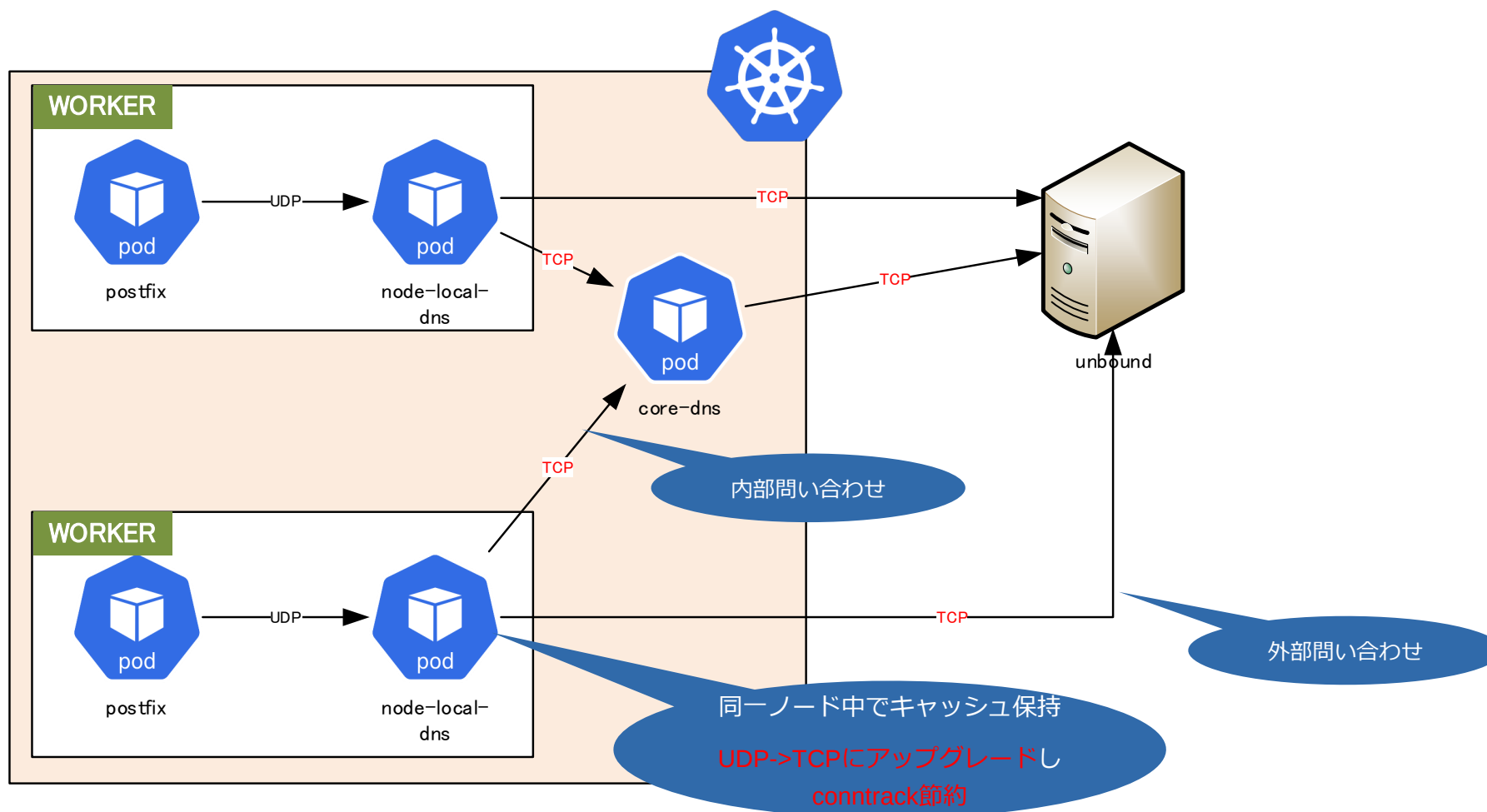
ndots:1 なら

1. omgi.iij.ad.jp(A)
2. omgi.iij.ad.jp(AAAA)

* omgi.iij.ad.jp: iij.ad.jpのMXレコード

すっきり（普通・・・）

名前解決命なので、DNSキャッシュも必須 node-local-dnsという素晴らしいキャッシュソフト(daemonset)有り



unbound側でTCP問い合わせ上限を増やさないと名前解決が飽和します



Internet Initiative Japan Inc.

Unbound移行時(後)のトラブル

- TCP無応答問題
 - 移行直後から、たまにTCPクエリをこぼす
 - incoming-num-tcp のデフォルト値(10)が小さすぎて詰まっていた
 - 当時open resolverだったり、網内にopen forwarderが大量にいた所為もあると思われる
 - incoming-num-tcp: 1000 にして解消

© 2015 Internet Initiative Japan Inc. 13

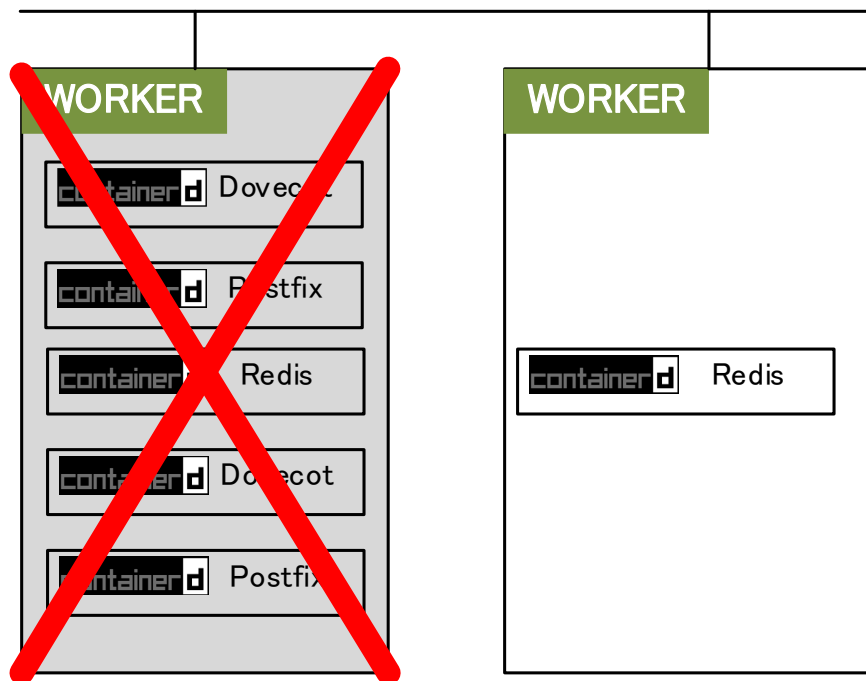
DNSのアップストリームがunboundの場合、
node-local-dnsはUDP->TCPに強制アップグレードするのではまります

- 各種最適化のためにKernelパラメータも随時調整
- PODレベル
 - kernel.msgmax
 - kernel.msgmnb
 - net.core.somaxconn
 - net.ipv4.ip_local_port_range
 - net.ipv4.tcp_tw_reuse
 -
- ノードレベル
 - fs.inotify.max_user_instances
 - fs.inotify.max_user_watches
 - ...

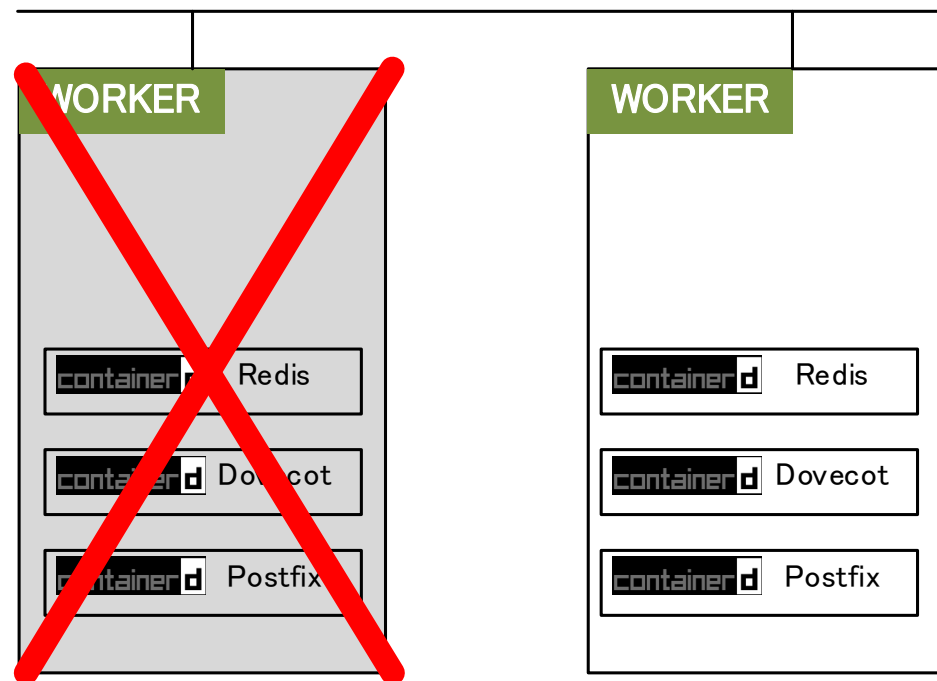
allowed-unsafe-sysctls で定義できるパラメータの追加が必要
<https://kubernetes.io/docs/tasks/administer-cluster/sysctl-cluster/>

Anti Affinityコントロール

- ・ 同一種別のPODは別ノードで起動させる
 - ・ ラックレベルで分ける
- 等



障害時影響大



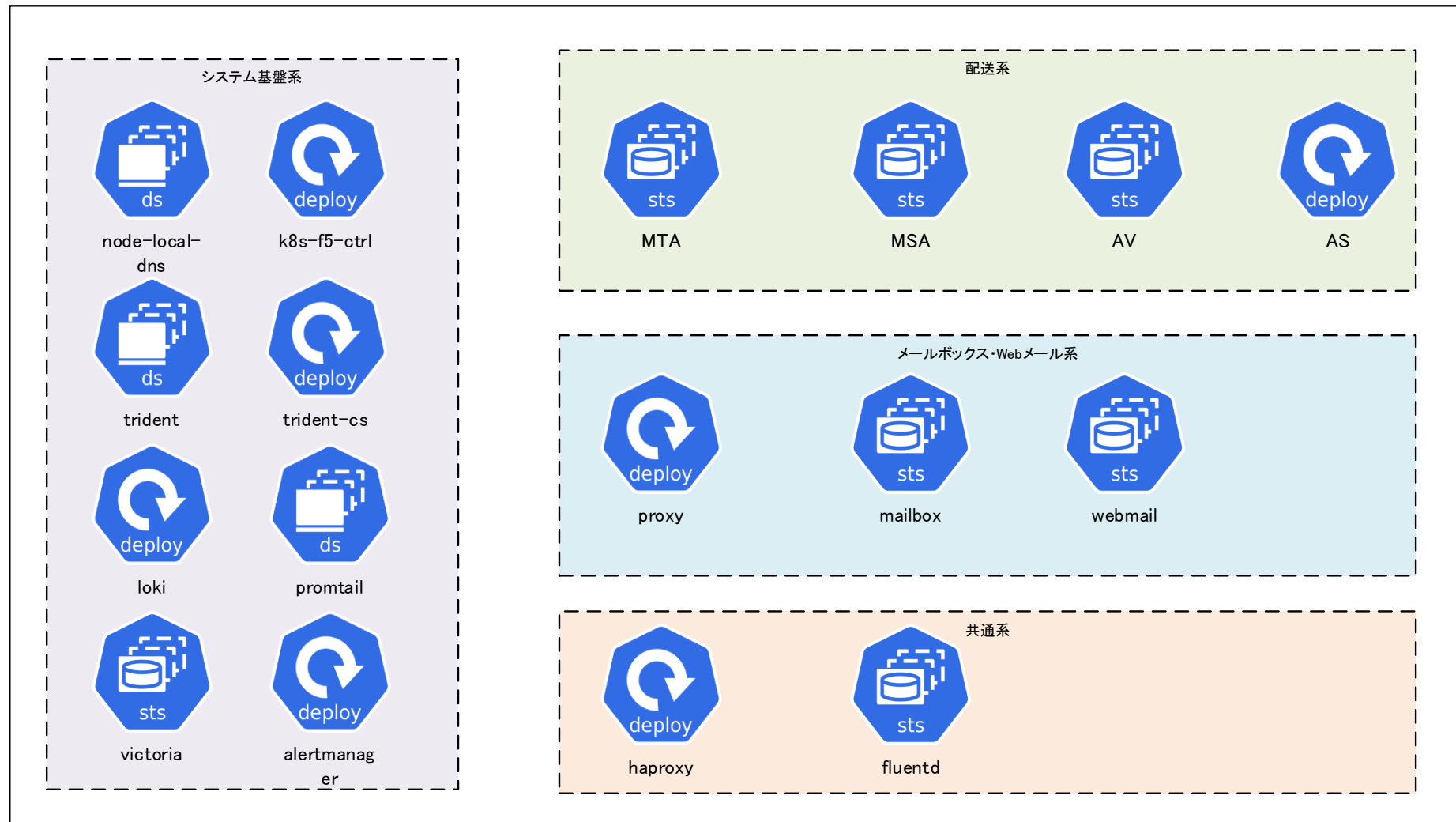
障害時影響低減

- **(重要)コンテナ化しても性能は(大して)変わらない**
- **ノードリソース使い切りは危険**
 - リソースの取り合い
- **CPU・I/O・NWを激しく使うコンテナは特に注意**
 - HEAVY PODがノードリソースを占有して全体がスローダウン
 - MTA・メールボックスは同居させない等

入念なスケジューリング設定が必要

最後に

作り上げるとこういうメールシステムになります (書ききれないので概要です)



LDAP

KVS

オブジェクト
ストレージ

検索エンジン

DB

ストレージ

LB

1. (重要) 世代交代できた！

- 若手が活躍！
- Kubernetesでの作り替えは世代交代に有益

2. 開発速度が上がった！

- インフラとアプリの分離は偉大
- 汎用性が高い点も◎

1. k8sは安定してる?:

A: 安定しています。

しっかり作ってテストやりきれば大丈夫。

2. どうしてオンプレミス?

A: 既存設備連携・IP流用・ストレージDR・出口IP制御(NAT制御)・カーネルパラメータ・FTP/FTPS提供等・・・

全部対応したらオンプレができあがっていた

3. 導入は大変か?

A: 学習コストは高いが、運用が楽になり結果的に楽

4. ソフトメーカーのKubeサポートはどの程度か?

A: メーカー依存。努力と交渉が必要なケースも。

1.Eメールサービス(+HPサービス)をKube化しました

2.導入難度は高いが、構成・運用が今風になり世代交代できました。

3.安定稼働。安心です。

ご清聴ありがとうございました。

本書には、株式会社インターネットイニシアティブに権利の帰属する秘密情報が含まれています。本書の著作権は、当社に帰属し、日本の著作権法及び国際条約により保護されており、著作権者の事前の書面による許諾がなければ、複製・翻案・公衆送信等できません。IIJ、Internet Initiative Japanは、株式会社インターネットイニシアティブの商標または登録商標です。その他、本書に掲載されている商品名、会社名等は各会社の商号、商標または登録商標です。本文中では™、®マークは表示していません。

©Internet Initiative Japan Inc. All rights reserved. 本サービスの仕様、及び本書に記載されている事柄は、将来予告なしに変更することがあります。