New Relic

Kubernetesとアプリをつないだ可視化を実現しよう！
-クラウドネイティブアプリの最新運用管理術-

New Relic 株式会社
ソリューションコンサルタント
佐々木 千枝

自己紹介

New Relic 株式会社　ソリューションコンサルタント
佐々木 千枝 (ささき ちえ)

デジタルビジネスに可観測性を提供するためのSaaSプラットフォーム、
New Relic のソリューションをお客様にご提供しています

■ソリューション例

・クラウドネイティブな環境への移行と運用(今日のテーマ)

・DevOpsの計測

・カスタマーエクスペリエンスの向上

ご興味のある方はお気軽にご相談ください！

csasaki@newrelic.com

某ゲームキャラと同姓同名です
(画像はニコニコ大百科から引用)

こっちの佐々木は
ボーカルでなく
キーボード弾きです

本日のアジェンダ

1. クラウドネイティブの意義と課題
2. クラウドネイティブのObservability(可観測性)の実現手段
3. New Relicによる可観測性ソリューション
4. まとめ

1. **クラウドネイティブの意義と課題**
2. クラウドネイティブのObservability(可観測性)の実現手段
3. New Relicによる可観測性ソリューション
4. まとめ

What is “Cloud Native” ?

“クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実行するための能力を組織にもたらします。 ”

From CNCF Cloud Native Definition v1.0(https://github.com/cncf/toc/blob/master/DEFINITION.md)

What is “Cloud Native” ?
(個人的な見解)

非クラウドネイティブ

予め定められた場所
(物理リソース)
から出られない

アプリさん(仮)

クラウドネイティブ

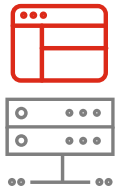
アプリがどこでも
自由に動ける

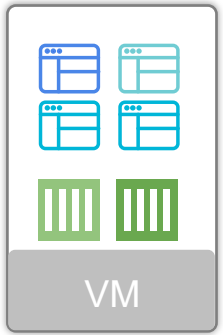
アプリさん(仮)

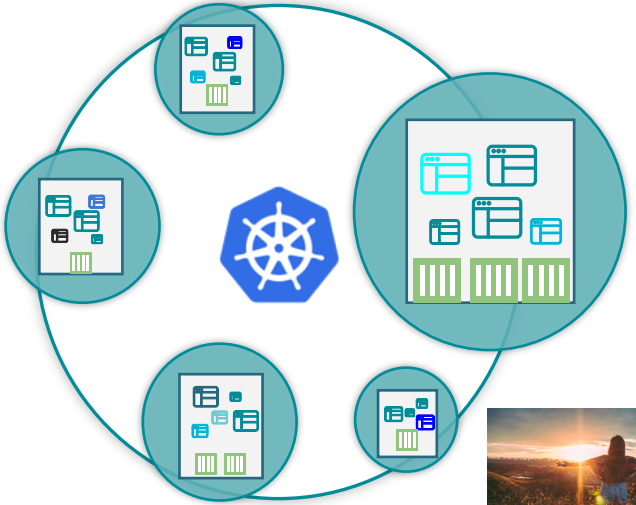
アプリケーションを物理的な制約から解き放つ

New Relic.

アプリケーションを物理的な制約から解き放つ
～物理サーバーからKubernetesへ～







物理サーバー上の
アプリケーション

物理サーバーによる制約

仮想マシン上の
アプリケーション

物理リソースから分離される

仮想マシン内の
コンテナ上の
アプリケーション

可搬性を得る

オーケストレーションされた
コンテナ上のアプリケーション

どこででも稼働できる

自由になったアプリーションはどうなるか？

アプリケーションの作りが変わる

非クラウドネイティブ

1つのモノリシックなアプリケーション

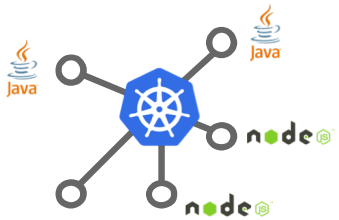
3サーバー

1 データベース

オンプレミス

四半期に1回デプロイ

クラウドネイティブ



250 のマイクロサービス

2,500 コンテナ

サービスごとのデータベース

複数クラウド

1日に15回デプロイ

誰かがちゃんとアプリたちを見てあげる必要がある

Trail Mapでも、4番目に”Observability”として定義

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape l.cncf.io has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

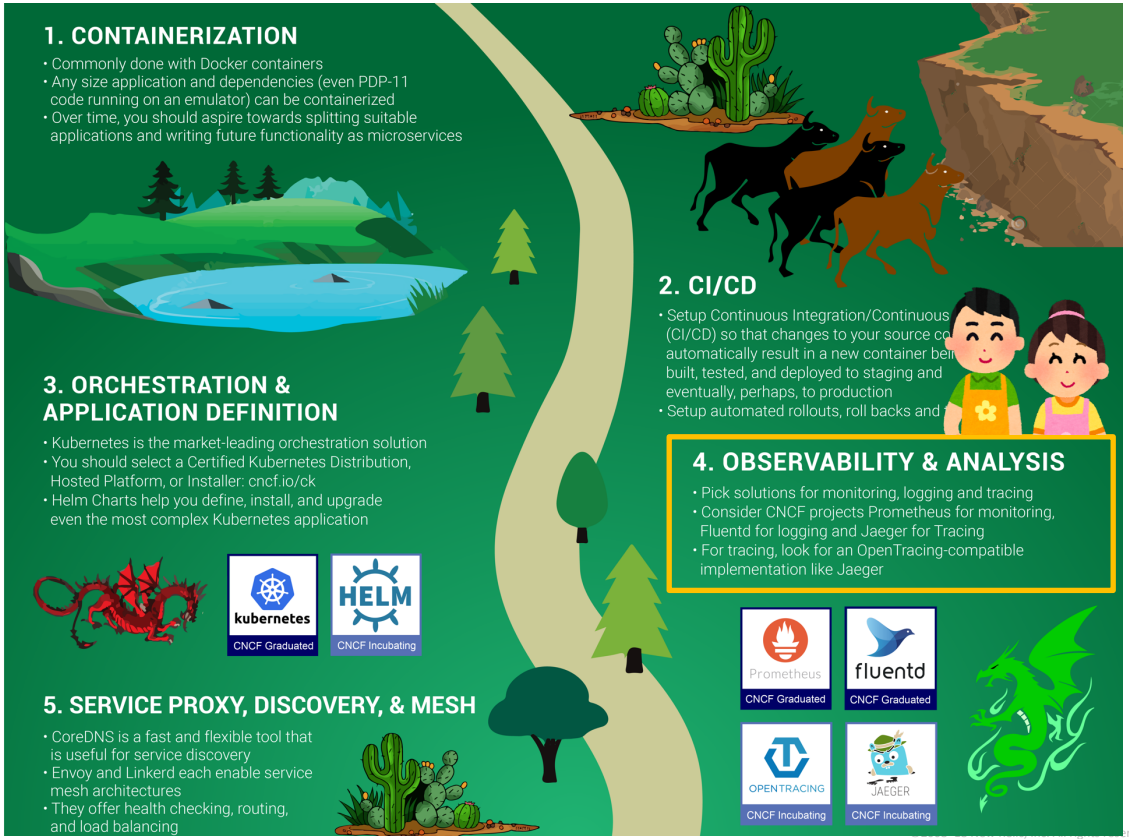
cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

C. Join CNCF's End User



ただし、見るべきポイントは今までの監視と違う

非クラウドネイティブ

- 見るべき対象は1つ
- 勝手にどこかに行かない
- 変化もあまりない
- リソースがきちんと割り当てられればわりとちゃんと動く

リソースセントリックに見ることが重要(だった)

クラウドネイティブ

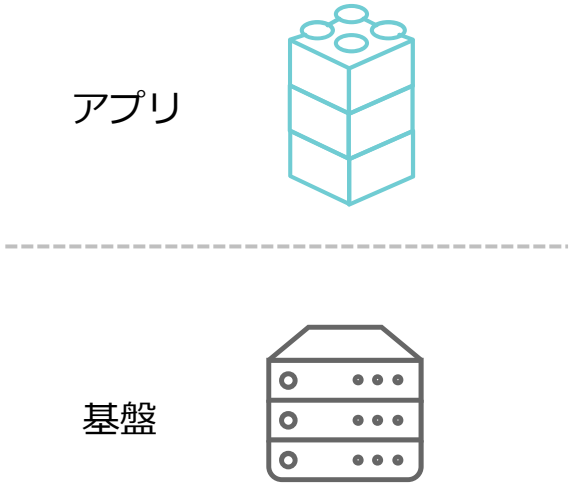
- 見るべき対象はたくさん
- どこで動いているかわからない
- 日々デプロイがあり変化も激しい
- アプリ同士の関係性も複雑

アプリケーションセントリックに見ることが重要

1. クラウドネイティブの意義と課題
2. **クラウドネイティブのObservability(可観測性)の実現手段**
3. New Relicによる可観測性ソリューション
4. まとめ

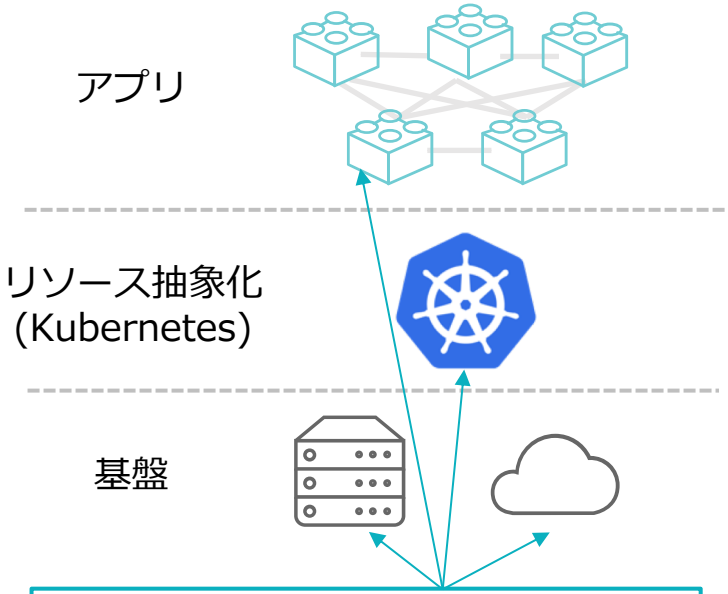
何をObserveするのか？

これまでの監視



リソースや死活といった**状態**を監視

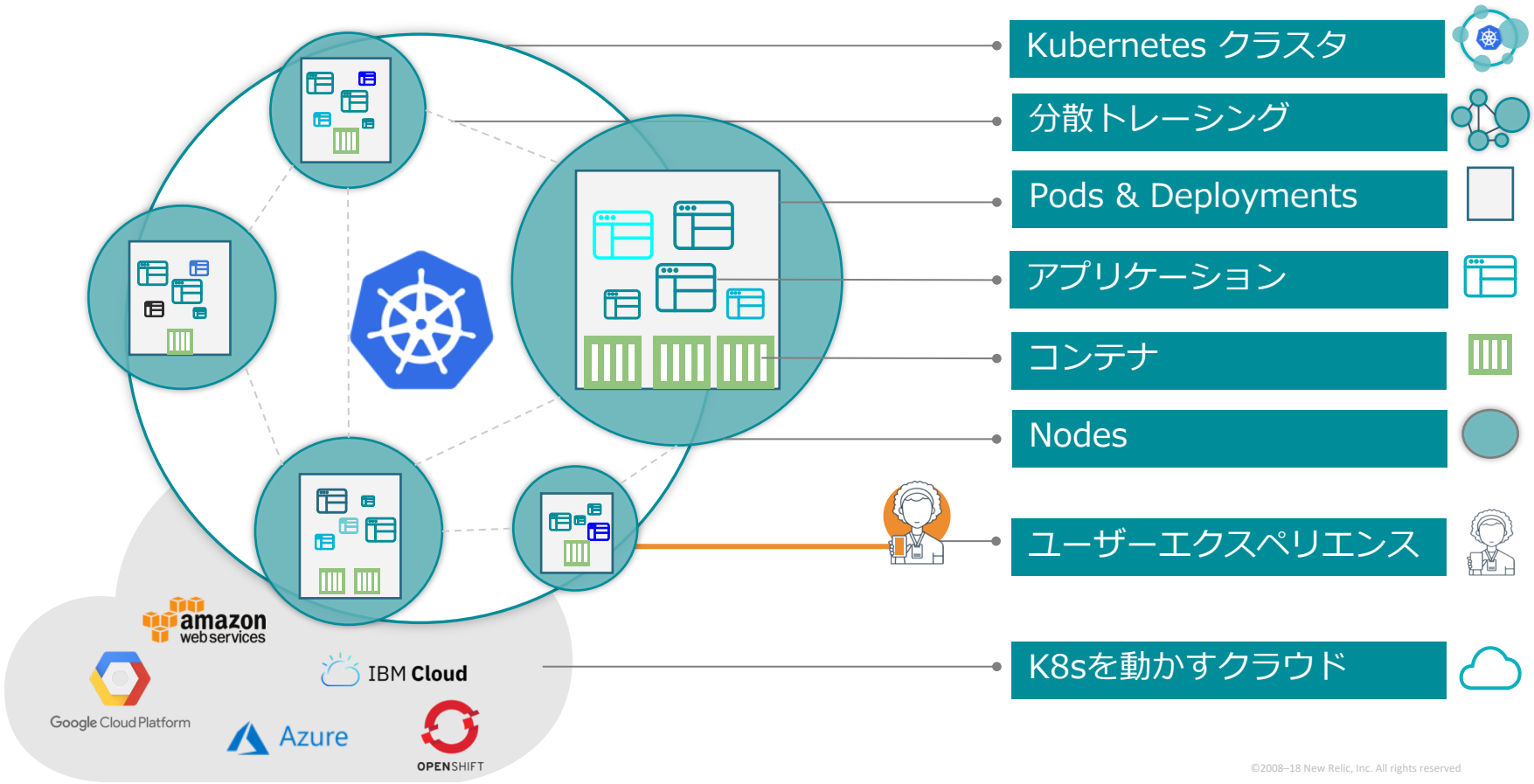
クラウドネイティブの Observablity



状態を見るだけではなく、関連する
コンポーネント群の**関係性**を把握

New Relic.

アプリを中心に、動的な環境を多面的に見る



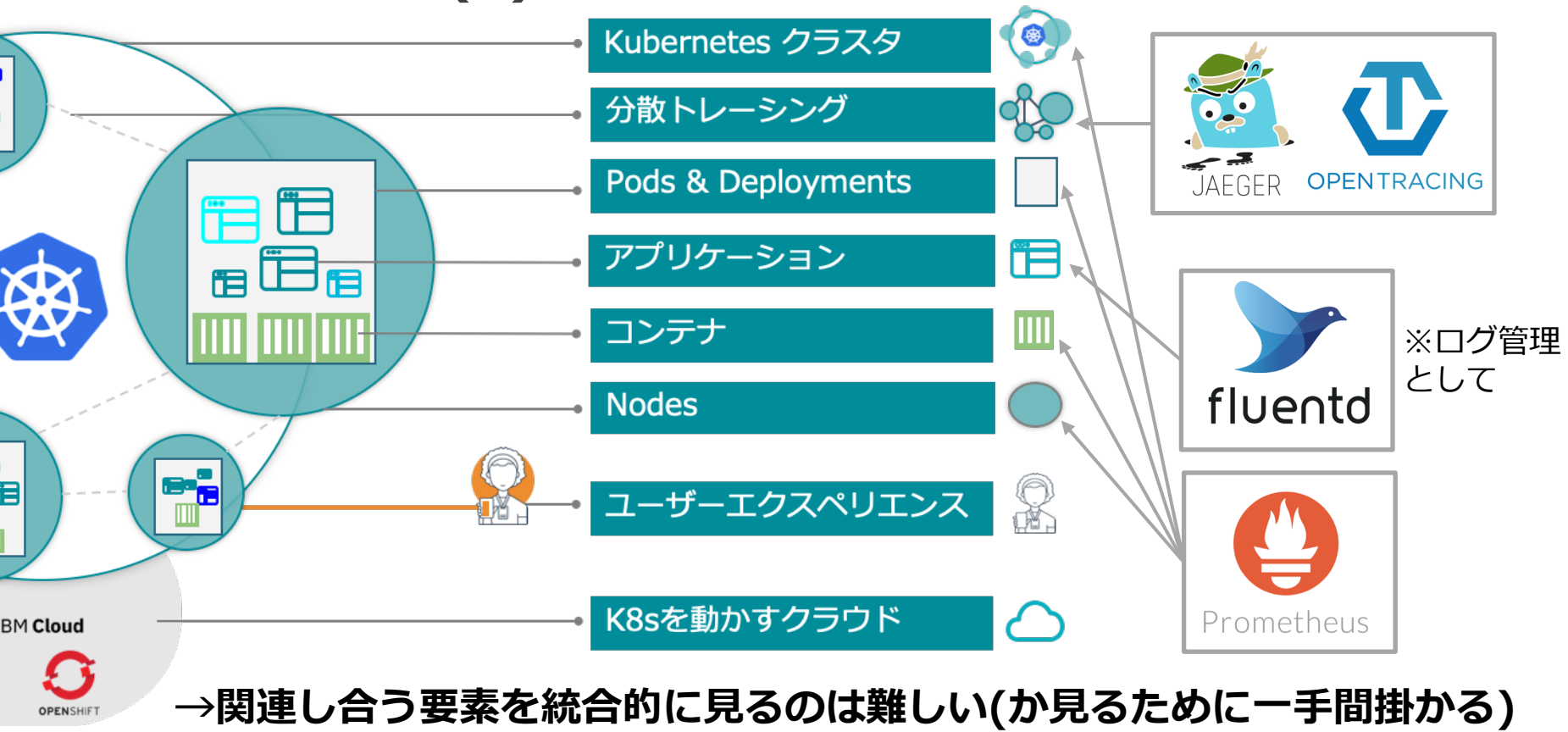
New Relic.

ここで、こんなこと思っていませんか？

(´_ゝ`)

でもObservabilityってOSSで実現できるんでしょ？

OSSの特徴(1)それぞれ得意分野を明確に持っている



→関連し合う要素を統合的に見るのは難しい(か見るために一手間掛かる)

OSSの特徴(2)自分で好きなようにカスタマイズする前提

例: Prometheusであればgrafanaを使って可視化する前提

→様々なツールを自分たちの手で導入・設定・運用するスキルと工数が必要

OSSでも可観測性は得られるが・・・

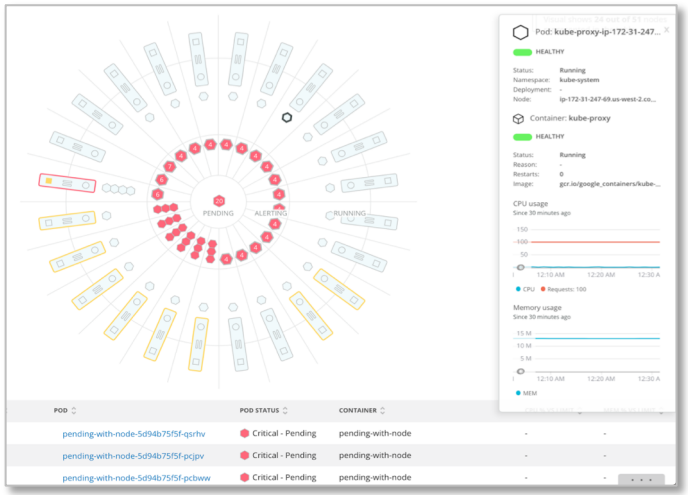
“作るのではなく買う” という選択肢もある

(1)クラウドネイティブの運用がこなれていない段階で、ベンダーのノウハウが詰まった統合的なビューを活用してみたいとき

(2)導入・設定・運用するスキルと工数の人的コストがツールの購入コストを上回るとき

New Relicはクラウドネイティブの可観測性実現に最適

- Kubernetes クラスタ
- 分散トレーシング
- Pods & Deployments
- アプリケーション
- コンテナ
- Nodes
- ユーザーエクスペリエンス
- K8sを動かすクラウド



SaaSで提供しており設定が簡単

Kubernetesとアプリをつないだ統合的なビューを提供

1. クラウドネイティブの意義と課題
2. クラウドネイティブのObservability(可観測性)の実現手段
3. **New Relicによる可観測性ソリューション**
4. まとめ

すべてを計測し、見通すための場所

New Relic.

New Relicによる可観測性ソリューション

導入編

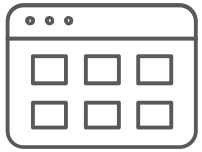
まずは見るための
初期設定をする

実践編

見るべき指標を知り、
それに沿ったビューの
見方を理解する

応用編

自分の環境にあった
ビューをカスタマイズする



New Relicによる可観測性ソリューション

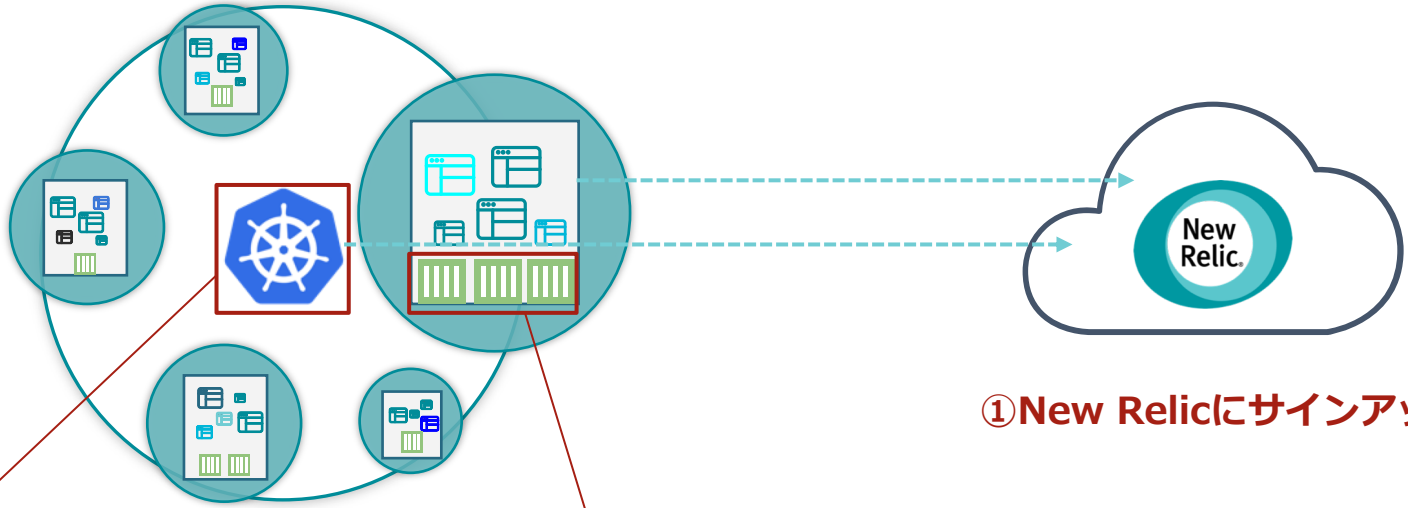
導入編 | 実践編 | 応用編

まずは見るための
初期設定をする

導入手順はたったの3ステップ

自分の環境

New Relic (SaaS)



①New Relicにサインアップ

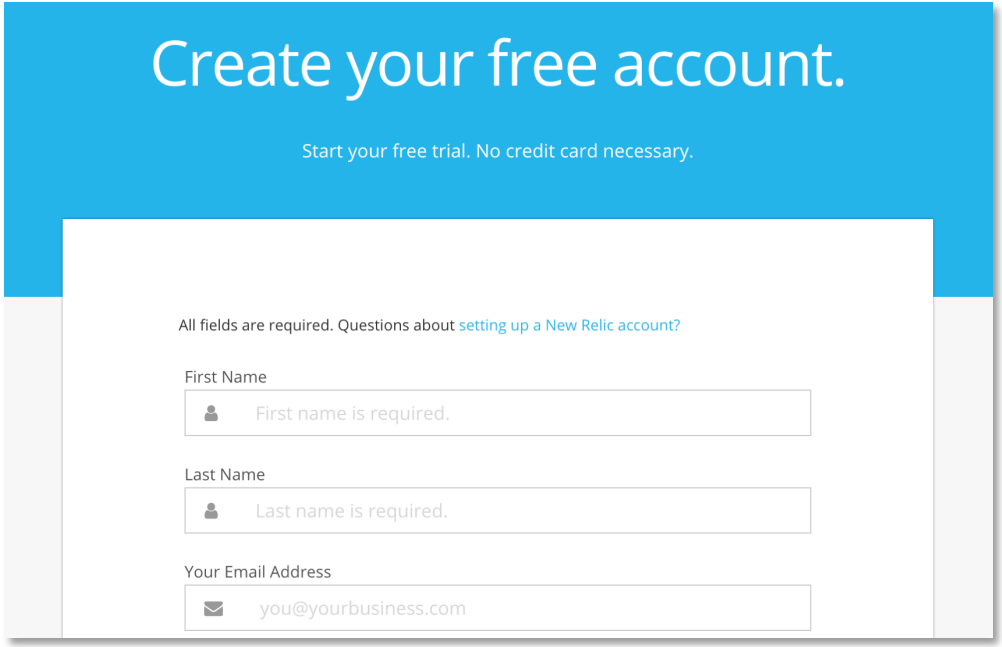
②Kubernetesクラスタに
Kubernetes Integrationを
インストール

③コンテナイメージに各言語の
エージェントをインストール

①New Relicにサインアップ

弊社サイトからサインアップ(2週間無料トライアルできます)

https://newrelic.com/signup



New Relic.

②KubernetesクラスタにKubernetes Integrationをインストール

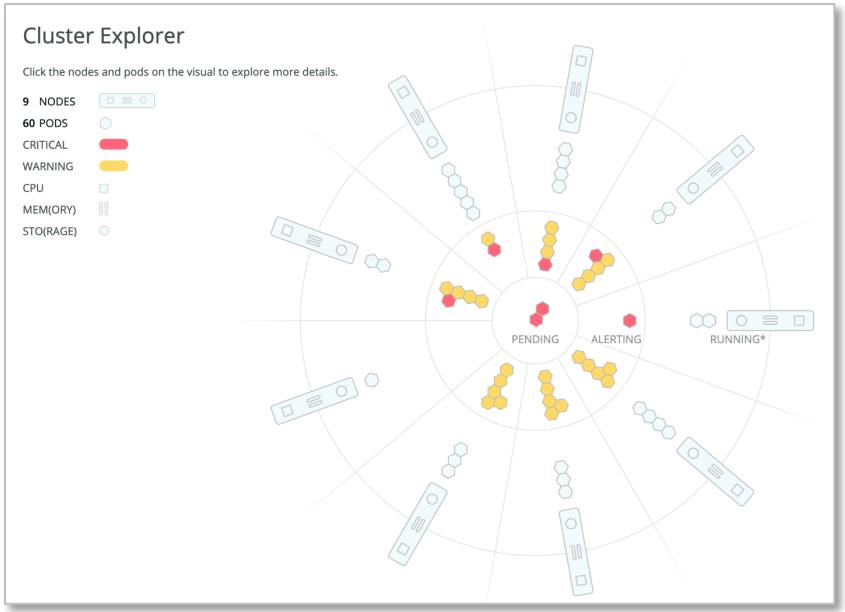
Kube-state-metricのインストール

```
# kubectl apply -f kube-state-metrics-release-1.6/kubernetes
clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created
clusterrole.rbac.authorization.k8s.io/kube-state-metrics created
deployment.apps/kube-state-metrics created
rolebinding.rbac.authorization.k8s.io/kube-state-metrics created
role.rbac.authorization.k8s.io/kube-state-metrics-resizer created
serviceaccount/kube-state-metrics created
service/kube-state-metrics created
# kubectl get pods --all-namespaces | grep kube-state-metrics
kube-system   kube-state-metrics-7885dd5f7f-4gmq4   2/2     Running
0          16s
```

Daemon setのインストール

```
# kubectl create -f newrelic-infrastructure-k8s-latest.yaml
serviceaccount/newrelic created
clusterrole.rbac.authorization.k8s.io/newrelic created
clusterrolebinding.rbac.authorization.k8s.io/newrelic created
daemonset.extensions/newrelic-infra created
# kubectl get daemonsets
NAME             DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR   AGE
newrelic-infra   3         3         3       3            3           <none>
8s
```

あとは数分待つだけでクラスタがこんな感じに見えてきます！



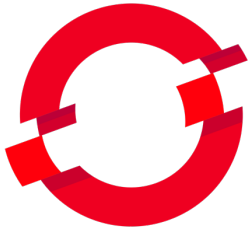
New Relic.

もちろん商用ディストリビューションでも使えます

Google Kubernetes Engine

Amazon EKS

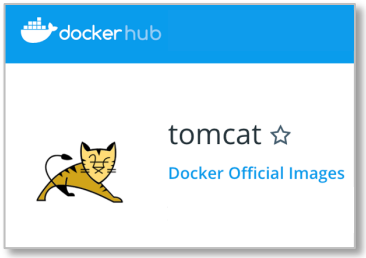
Azure Kubernetes Service (AKS)



OPENSHIFT

Pivotal Container Service™

③コンテナイメージに各言語のエージェントを インストール(tomcatの例)



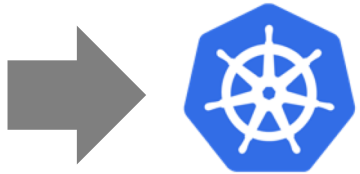
+

Dockerfile

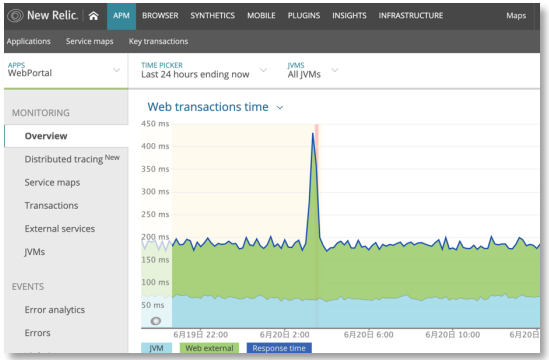
```
WORKDIR /usr/local/tomcat/

RUN curl -O "http://download.newrelic.com/newrelic/java-agent/newrelic-agent/current/newrelic-java.zip"

RUN ["apt-get", "install", "unzip"]

RUN ["unzip", "newrelic-java.zip", "-d", "/usr/local/tomcat"]

# change working directory to Tomcat
#WORKDIR /usr/local/tomcat/bin

# run the Tomcat Server
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

WORKDIR /usr/local/tomcat/webapps
RUN wget xxxx
```

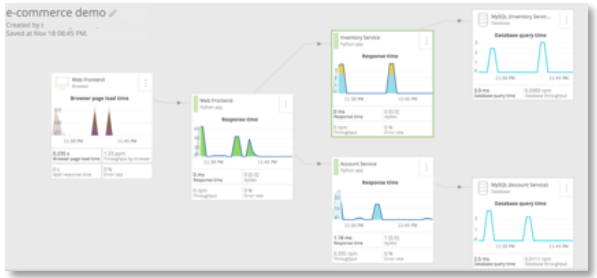
yaml内”containers”の定義

```
env:
        - name: JAVA_OPTS
          value: ”
-javaagent:/usr/local/tomcat/newrelic/newrelic.jar
-Dnewrelic.config.license_key=xxxxx
-Dnewrelic.config.app_name=AppName
-Dnewrelic.config.distributed_tracing.enabled=true"
```



あとは数分待つだけでアプリがこんな感じに見えてきます!!!





New Relic.

いろんな言語用のエージェントがあります
(導入手順も言語によって異なります)

C SDK

Go

Java

.NET

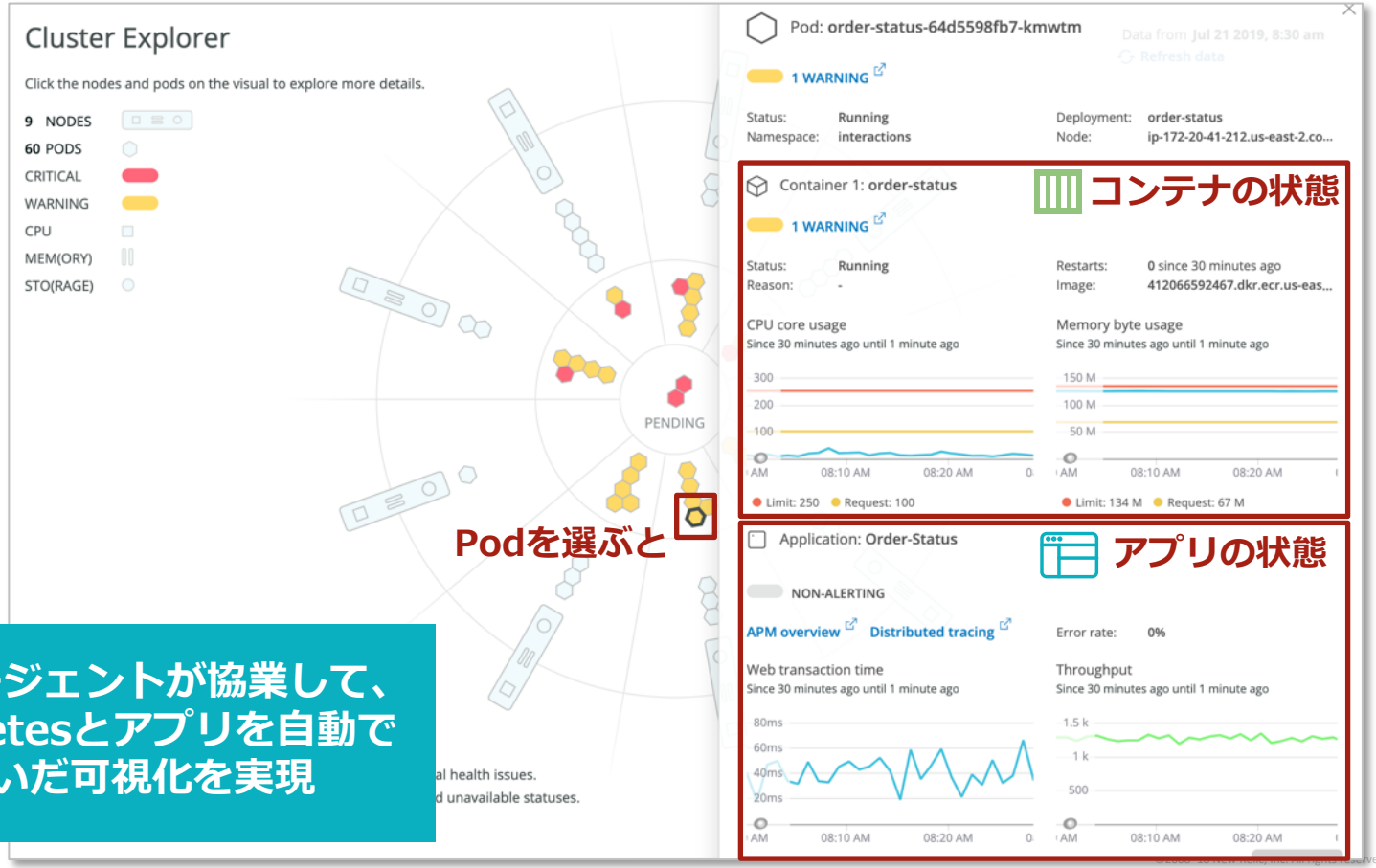
Node.js

PHP

Python

Ruby

さらに・・・



2つのエージェントが協業して、Kubernetesとアプリを自動でつないだ可視化を実現

New Relicによる可観測性ソリューション

導入編 | 実践編 | 応用編

見るべき指標を知り、
それに沿ったビューの
見方を理解する

クラウドネイティブで見るべき指標

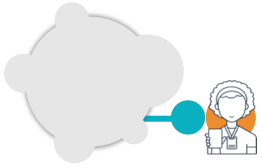
レイヤ	コンポーネント	見る目的	見るべき指標例
アプリ	ユーザーエクスペリエンス	クラウドネイティブ上で提供しているビジネスのユーザーが今どのような体験をしているか把握する	ページロードタイム、Javascriptエラー、外形監視
	アプリケーションパフォーマンス	Kubernetesの中で動的に実行環境が変化するアプリの応答速度やエラー発生状況を正しく把握する	応答時間、スループット、エラー率
	分散トレーシング	複数のPodにまたがって関連しあっているアプリの中でどこにボトルネックがあるのかを把握する	異常箇所の特定、外部API稼働状況
Kubernetes	Cluster	様々なコンポーネントから成り、複雑化・動的に変化する環境の現状を理解する	※指標ではないが、Clusterの構成要素の視覚的な把握
	Node	Clusterを構成するNodeのステータスのリソース使用状況が問題ないことを確認する	死活監視、リソース使用率
	Pod/Deployment	DeploymentやPodがyamlで定義されたとおりに正しく稼働しているかを確認する	Podステータス、レプリカセット数
	コンテナ	コンテナのリソース使用状況や、それに伴う異常稼働の有無を確認する	リソース使用率、再起動発生数
インフラ	クラウド	Kubernetesを支えるインフラ側から見たパフォーマンス状況を把握する	稼働率、サービスとしての遅延、スループット

クラウドネイティブで見るべき指標

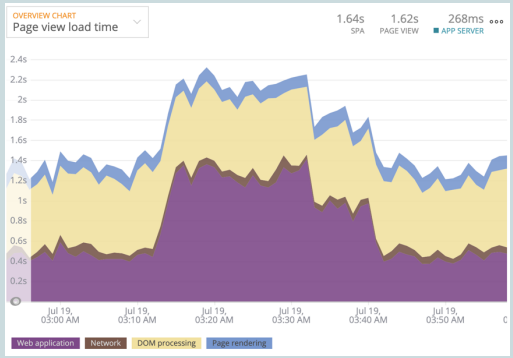
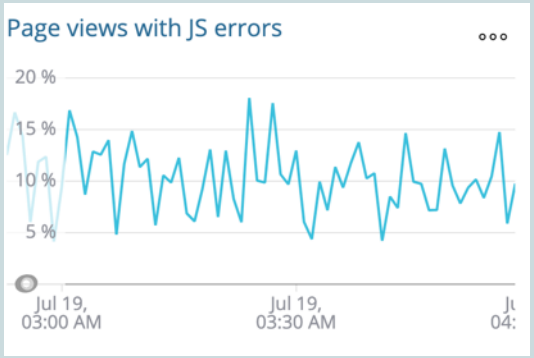
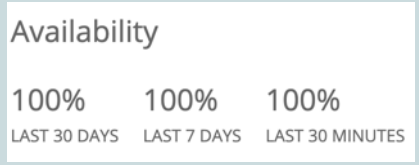
レイヤ	コンポーネント	見る目的	見るべき指標例
アプリ	ユーザーエクスペリエンス	クラウドネイティブ上で提供しているビジネスのユーザーが今どのような体験をしているか把握する	ページロードタイム、Javascriptエラー、外形監視
	アプリケーションパフォーマンス	Kubernetesの中で動的に実行環境が変化するアプリの応答速度やエラー発生状況を正しく把握する	応答時間、スループット、エラー率
	分散トレーシング	複数のPodにまたがって関連しあっているアプリの中でどこにボトルネックがあるのかを把握する	異常箇所の特定、外部API稼働状況
Kubernetes	Cluster	様々なコンポーネントから成り、複雑化、動的に変化する環境の現状を理解する	※指標ではないが、Clusterの構成要素の視覚的な把握
	Node	Clusterを構成するNodeの一つ一つのリソース使用状況が問題ないことを確認する	死活監視、リソース使用率
	Pod/Deployment	DeploymentやPodがyamlで定義されたとおりに正しく稼働しているかを確認する	Podステータス、レプリカセット数
	コンテナ	コンテナのリソース使用状況や、それに伴う異常稼働の有無を確認する	リソース使用率、再起動発生数
インフラ	クラウド	Kubernetesを支えるインフラ側から見たパフォーマンス状況を把握する	稼働率、サービスとしての遅延、スループット

New Relic

New Relicはこれらを標準機能で計測、可視化アラート設定も可能

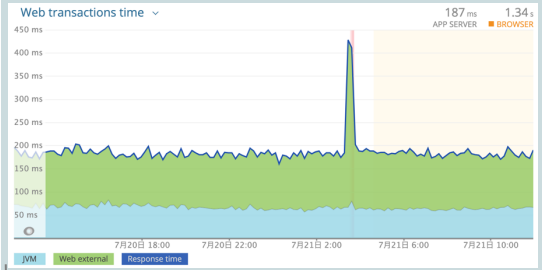
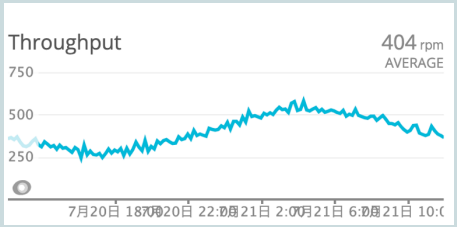
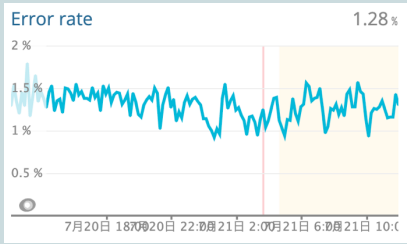


ユーザーエクスペリエンス

コンポーネント	ユーザーエクスペリエンス
見る目的	クラウドネイティブ上で提供しているビジネスのユーザーが今どのような体験をしているか把握する
ゴール	アプリケーションの性能がビジネスに与える悪影響を極力削減する
見るべき指標例	ページロードタイム  Javascriptエラー  外形監視 

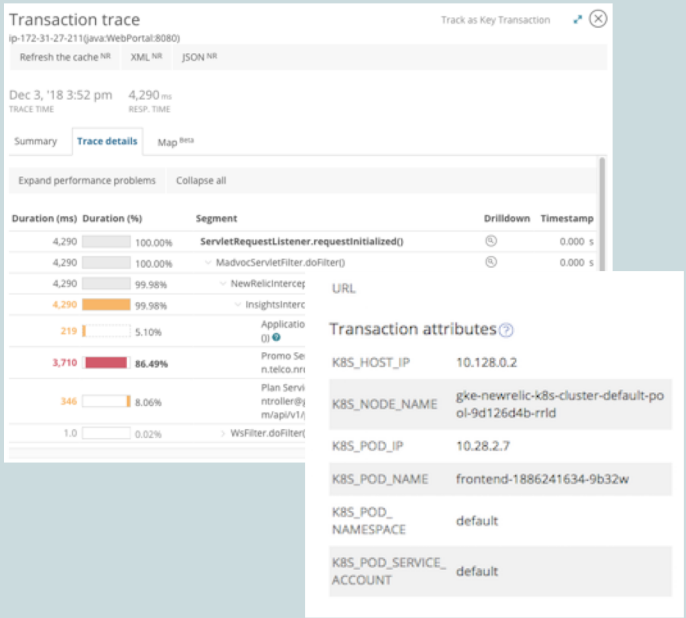
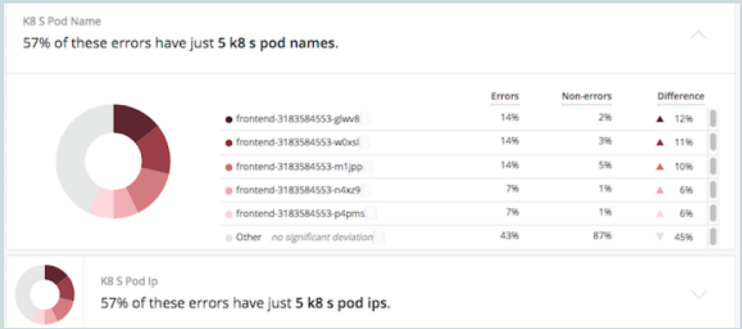
アプリケーション

開発者

コンポーネント	アプリケーション
見る目的	Kubernetesの中で動的に実行環境が変化するアプリの応答速度やエラー発生状況を正しく把握する
ゴール	アプリケーションの問題の特定に掛かる時間と労力を削減する
見るべき指標例	応答時間  スループット  エラー率  さらに根本原因を深掘りしていく(次ページ)

アプリケーション(続き)

開発者

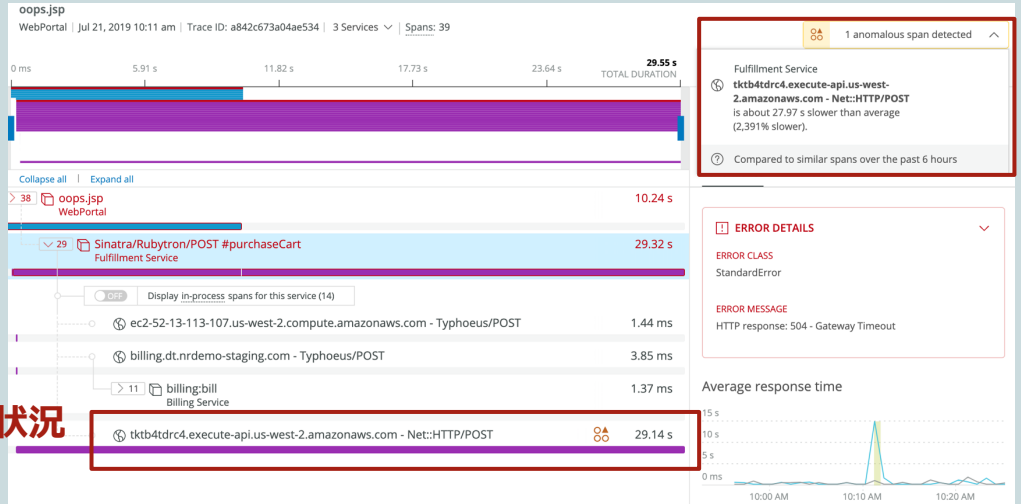
コンポーネント	アプリケーション
根本原因の分析例	処理に時間が掛かっているコードの箇所と発生Podを特定  エラーの発生傾向を分析 

New Relic.

分散トレーシング

開発者

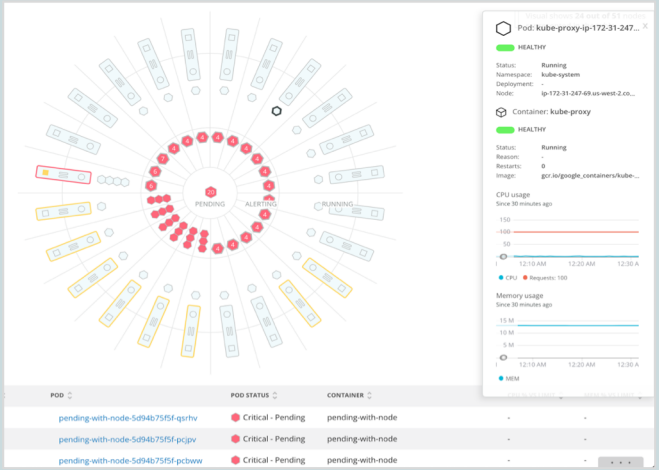
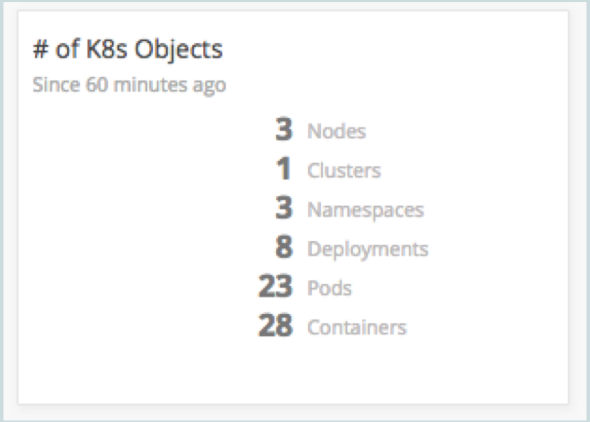
コンポーネント	分散トレーシング
見る目的	複数のPodにまたがって関連しあっているアプリの中でどこにボトルネックがあるのかを把握する
ゴール	アプリケーションの問題の特定に掛かる時間と労力を削減する
見るべき指標例	異常箇所の特定、外部API稼働状況



New Relic.

Cluster

クラスタ管理者

コンポーネント	Cluster
見る目的	様々なコンポーネントから成り、複雑化・動的に変化する環境の現状を理解する
ゴール	クラスタ内で問題が発生した場合、またはリスクがある際ににすべてを見渡し原因を早く突き止める
見るべき観点	すべてを見渡すビュー  定量的な評価 

Node

クラスタ管理者

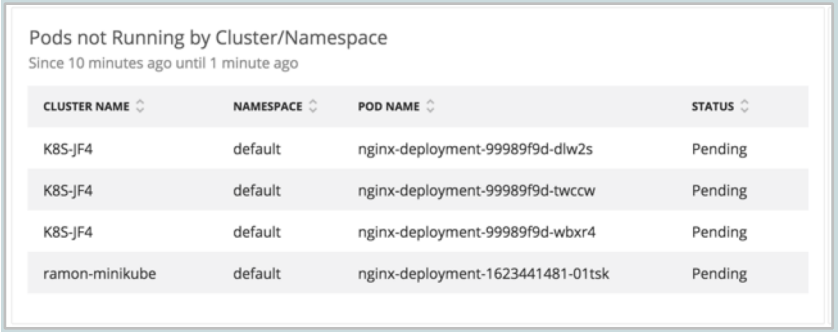
コンポーネント	Node
見る目的	Clusterを構成するNodeのステータスのリソース使用状況が問題ないことを確認する
ゴール	Clusterが提供する基盤リソースに不足がない状態を維持する
見るべき指標例	死活監視、リソース使用率

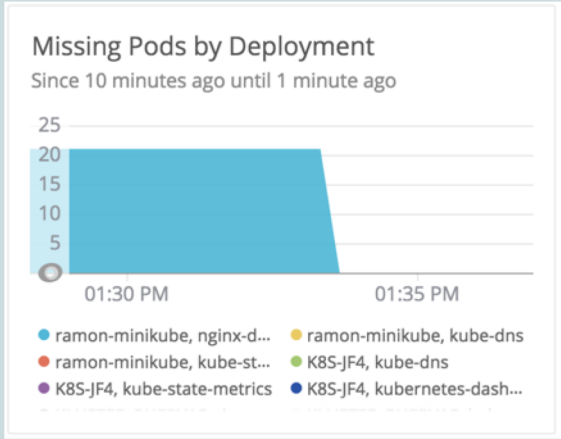
New Relic.

Pod/Deployment

運用者

コンポーネント	Pod/Deployment
見る目的	DeploymentやPodがyamlで定義されたとおりに正しく稼働しているかを確認する
ゴール	Kubernetesのレイヤで発生している論理的な問題を早急に知り、回復する
見るべき指標例	Podステータス(Runningでないもの) レプリカセットの設定に基づいて不足しているPod数

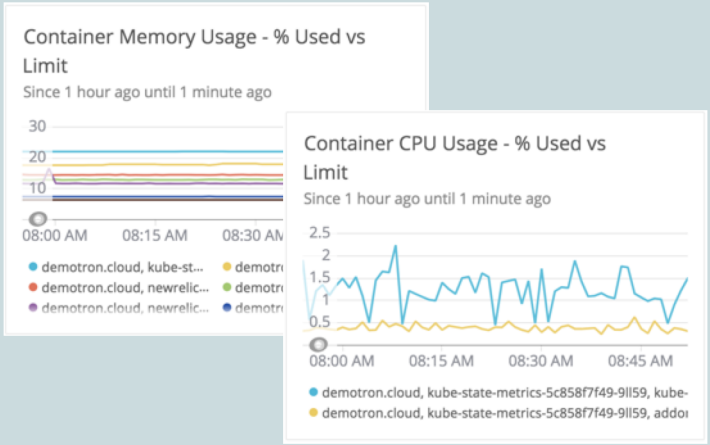
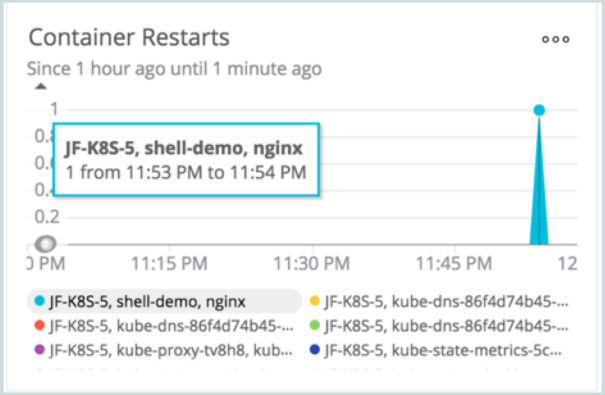




New Relic.

コンテナ

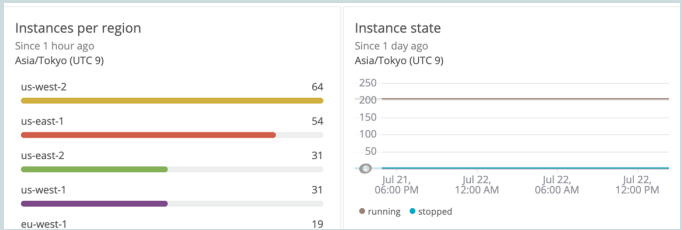
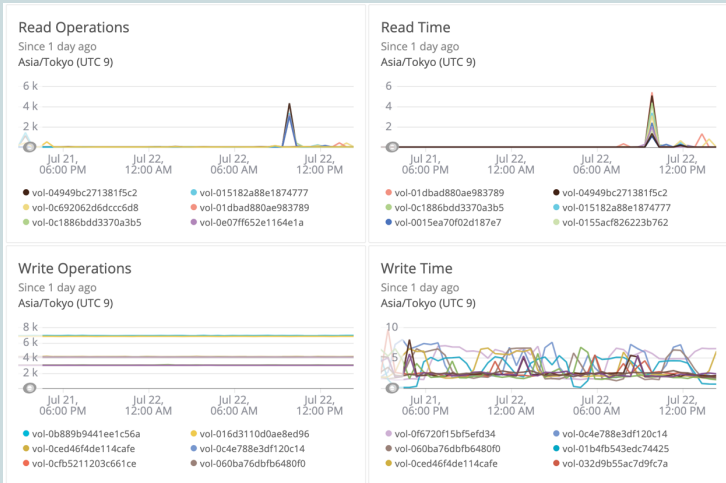
開発者/
運用者

コンポーネント	コンテナ
見る目的	コンテナのリソース使用状況や、異常稼働の有無を確認する
ゴール	コンテナで発生しているリソース不足や問題を早急に知り、対応する
見るべき指標例	リソース使用率  再起動発生数 

New Relic.

クラウド

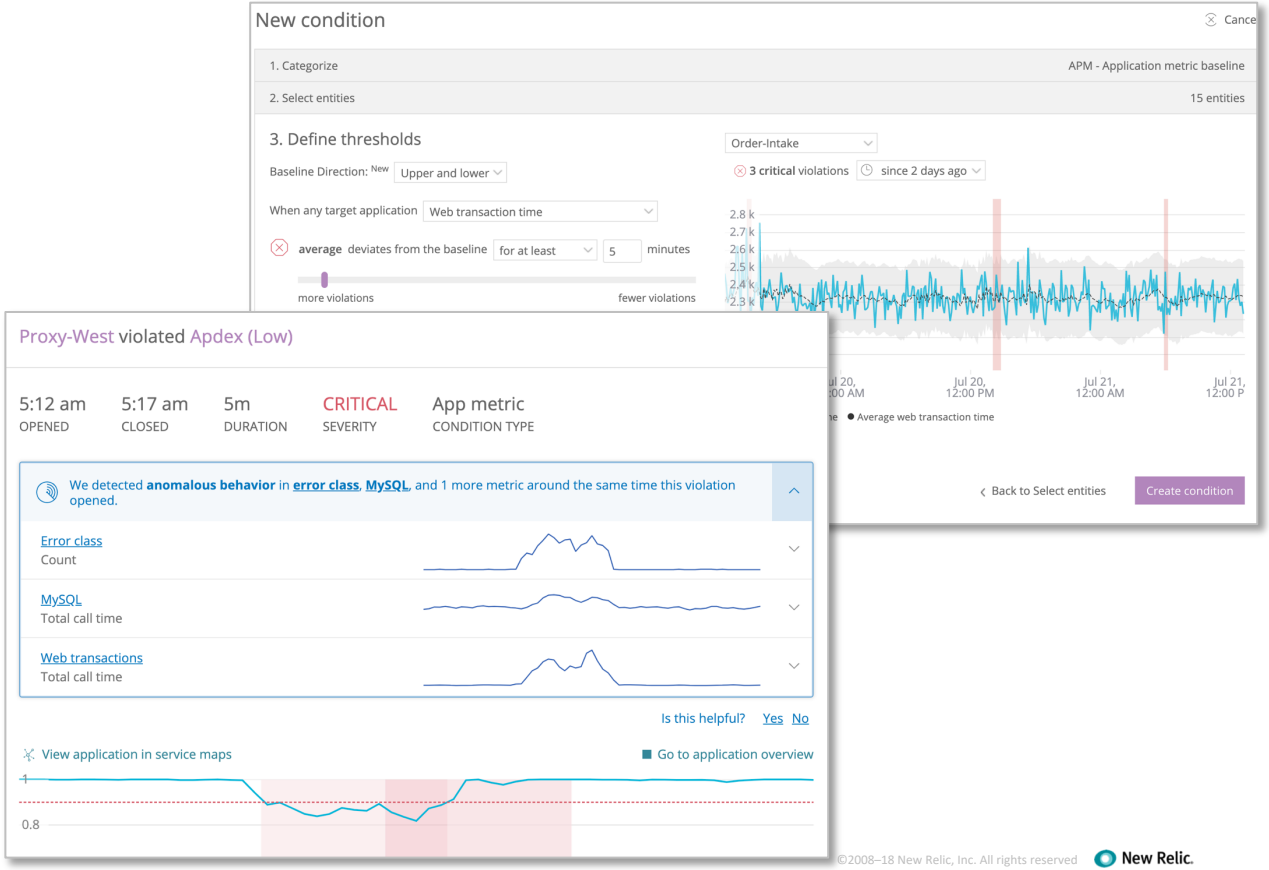
クラウド
管理者

コンポーネント	(Kubernetesを動かす基盤としての)クラウド
見る目的	Kubernetesを支えるインフラ側から見たパフォーマンス状況を把握する
ゴール	インフラ側でのパフォーマンス不足を防ぐ
見るべき指標例	稼働率  サービスとしての遅延、スループット 

New Relic.

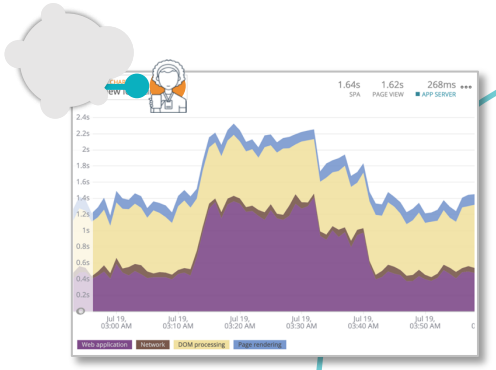
アラートの設定も簡単、便利

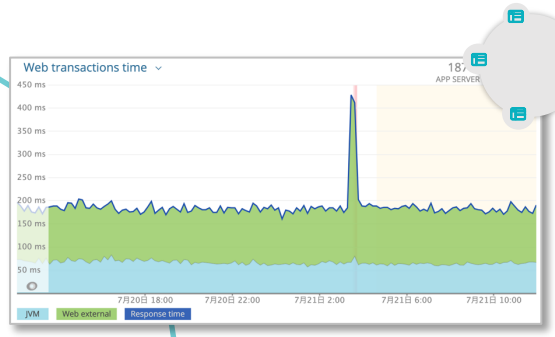
- 静的/動的閾値に対応
- 様々な通知と連携(メール、Slack、webhookなど)
- アラート発生時に、原因になっていそうな箇所を自動で判別



New Relic.

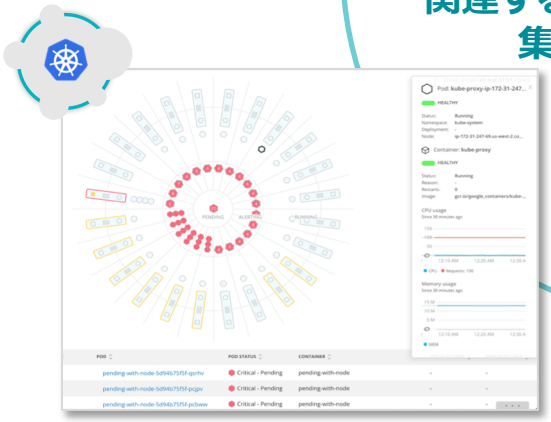
加えて重要なのは、関係性を把握すること

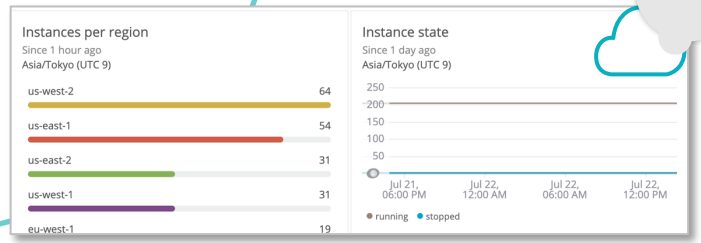




New Relic

関連するコンポーネントの情報を
集約し、関係性を把握



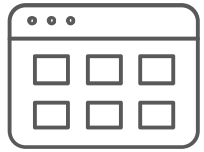


New Relic.

New Relicによる可観測性ソリューション

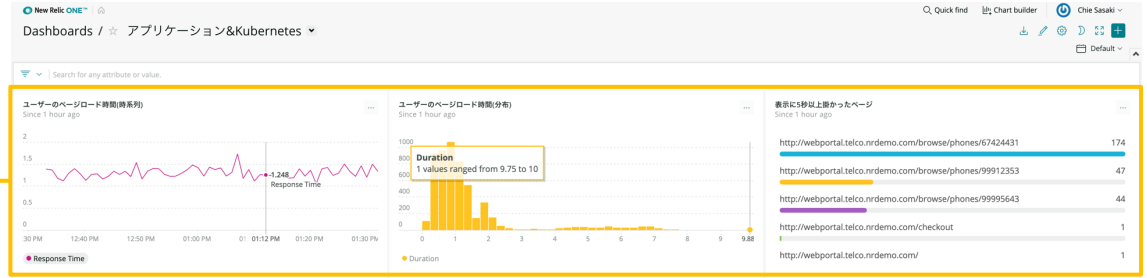
導入編 実践編 **応用編**

自分の環境にあったビューをカスタマイズする



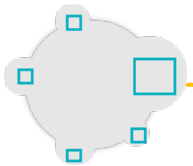
環境の特性に応じて、見たい指標をダッシュボード化

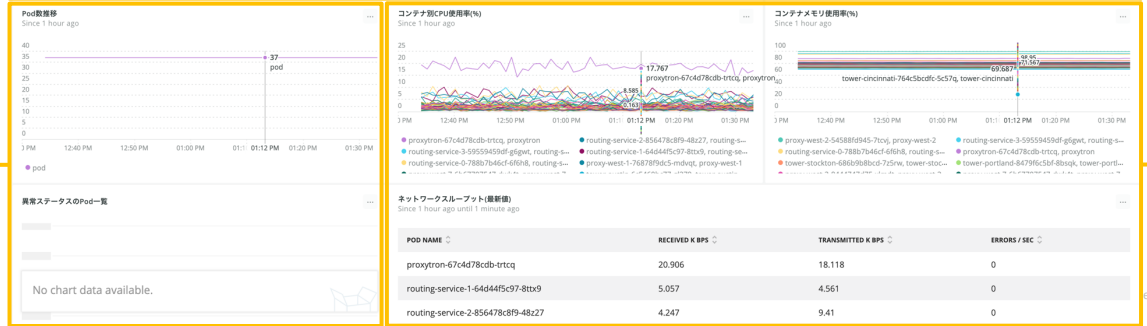
ユーザー
エクスペリエンス



アプリケーション

Pod/Deployment

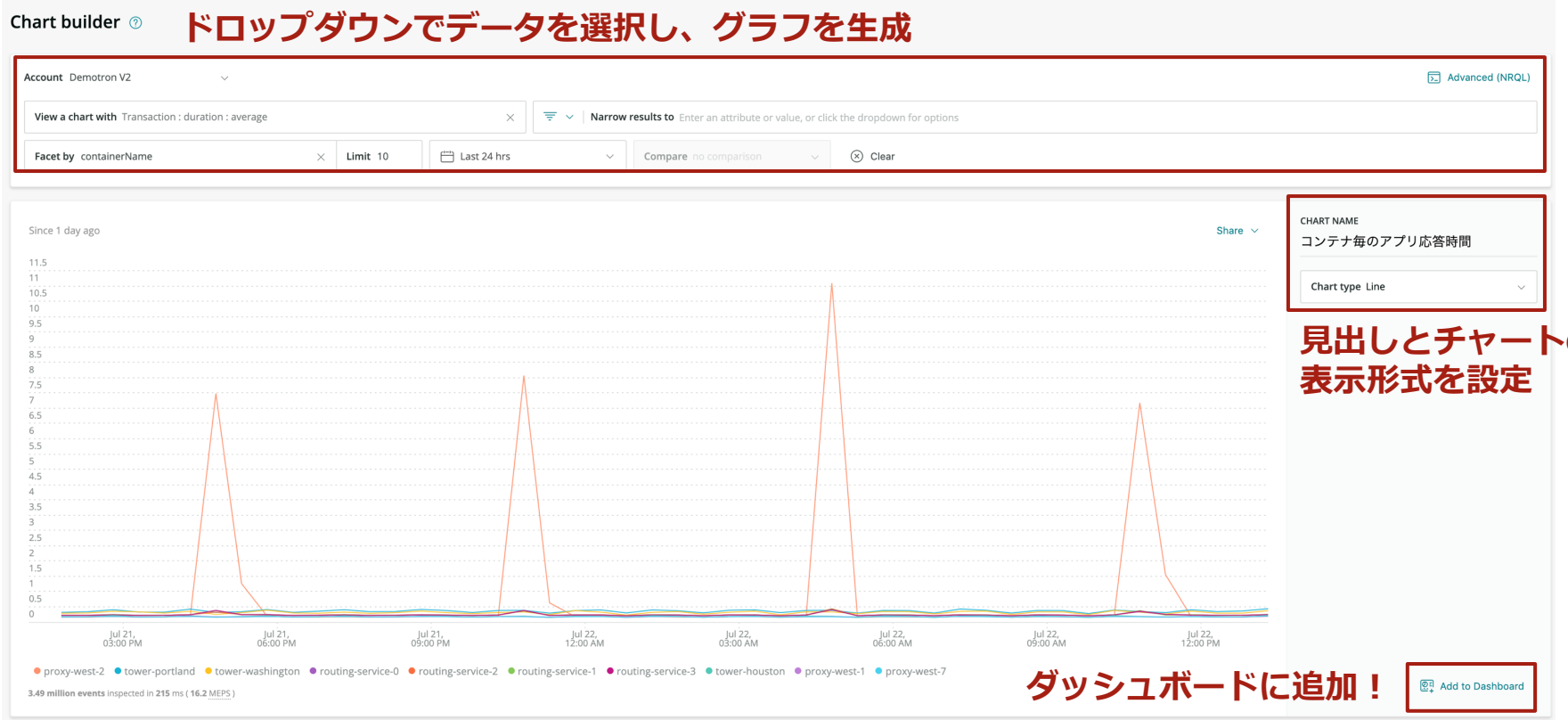




コンテナ

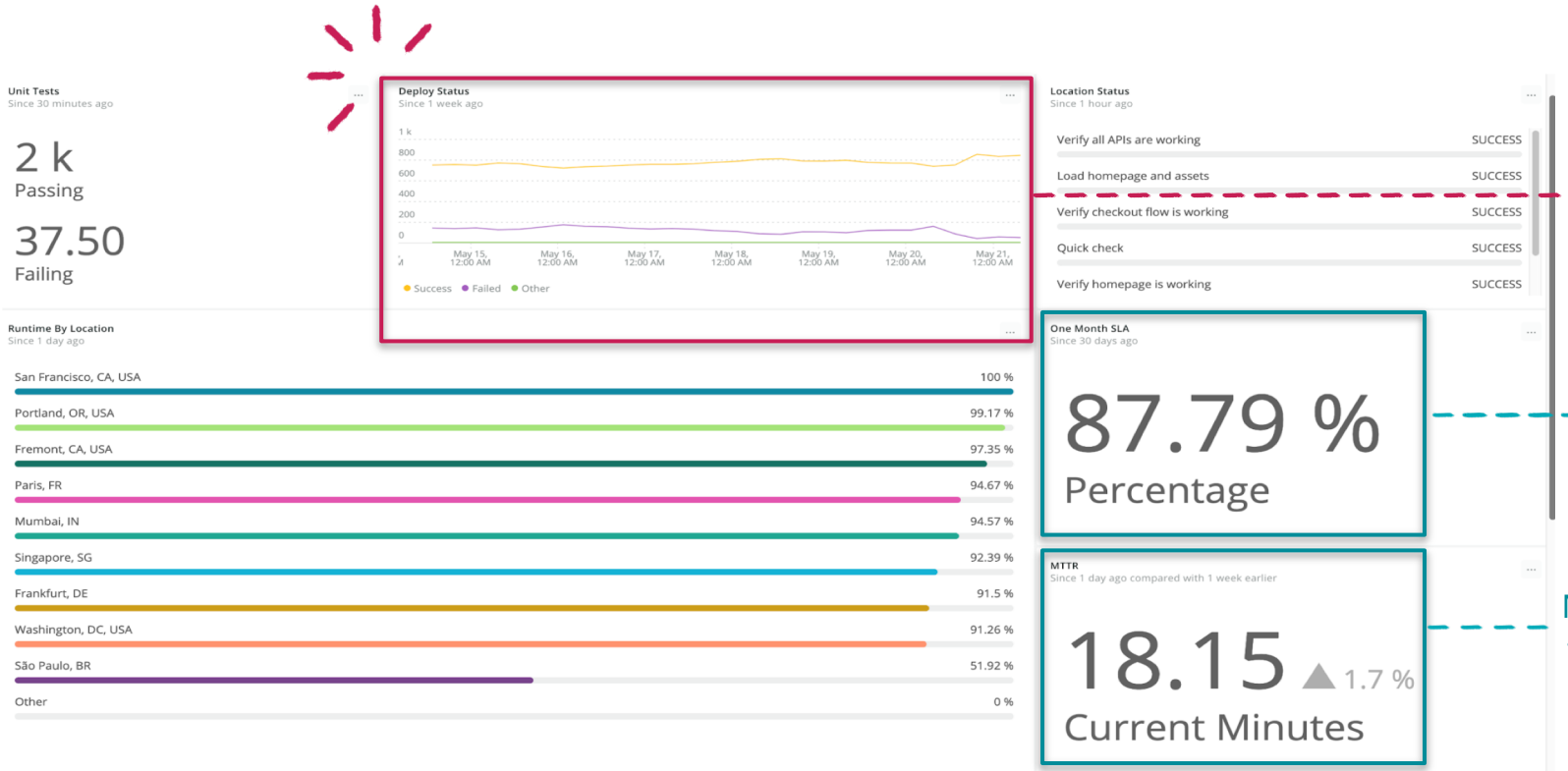
New Relic.

GUIで簡単にグラフを作成し、ダッシュボードに追加



New Relic.

分析次第で様々な観点のダッシュボードを作成可能
例. DevOpsの改善



New Relic.

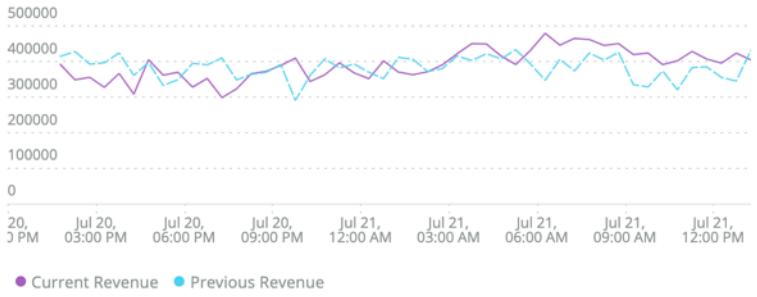
分析次第で様々な観点のダッシュボードを作成可能
例. ビジネスKPI

現在までの受注件数到達度 --> 1 日 3500 件目標
Since 1 week ago

2.49k /3.5k
ORDERS

71%

時間毎の売上額推移
Since 1 day ago compared with 1 day earlier



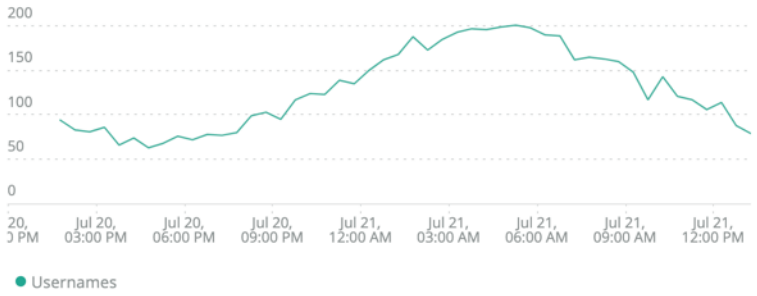
エラーの影響を受けたユーザー数
Since 1 day ago

340
人

アクセスユーザー数（前日比）
Since 1 day ago compared with 1 day earlier

18.7 k ▼ 1.9 %
Current 人

アクセスユーザー数　推移
Since 1 day ago



エラーの影響を受けた総売上額
Since 1 day ago

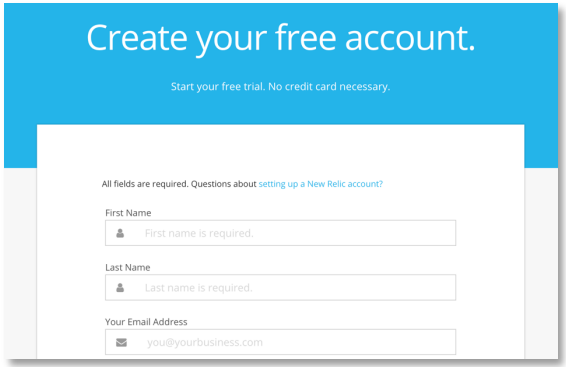
46713.608
円

New Relic.

1. クラウドネイティブの意義と課題
2. クラウドネイティブのObservability(可観測性)の実現手段
3. New Relicによる可観測性ソリューション
4. **まとめ**

まとめ

- クラウドネイティブ技術は、アプリケーションに自由を与えたが、一方でそのふるまいをアプリケーション中心に観測する必要性ももたらした
- 動的なKubernetes環境ではアプリケーションを中心として多次元的な観測が必要である
- New Relicは、Kubernetesとアプリをつないだ可視化を実現し、クラウドネイティブの観測に最適なSaaSプラットフォームである
- 導入から応用的な利用まで簡単にできるので、ぜひお気軽にお試しください！



・サインアップ
https://newrelic.com/signup

・ドキュメント(Kubernetes環境の設定)
https://docs.newrelic.co.jp/docs/integrations/kubernetes-integration/get-started/introduction-kubernetes-integration

New Relic.

Thank You

New Relic Japan のFacebook ページ

New Relic.