MC31103

vmware EXPLORE

開発者としてアプリを デプロイしたい、継続的に ～ Tanzu Application Platform で デプロイを加速させよう

ヴイエムウェア株式会社

VMware Tanzu Labs
シニアスタッフクラウドネイティブアーキテクト
槙 俊明

免責事項

- 本プレゼンテーションには、
  現在開発中の製品/サービスの特徴または機能が含まれている場合があります。
- 本セッションで紹介する新しいテクノロジーについて、
  VMware が製品/サービスにこれらの機能を搭載することを約束するものではありません。
- 機能は変更される場合があるため、
  いかなる種類の契約書、発注書/注文書、または販売契約書に記述されないものとします。
- 技術的な問題と市場の需要により、
  最終的に出荷される製品/サービスでは機能が変わる場合があります。
- ここで検討および提示されている新しいテクノロジーまたは機能の価格とパッケージは、
  決定されたものではありません。

製品名/サービス名 一覧

正式名称	略称	正式名称	略称	正式名称	略称	正式名称	略称
VMware vSphere®	vSphere	VMware Aria™	VMware Aria	VMware Tanzu® Build Service™	Tanzu Build Service	VMware NSX® Advanced Threat Prevention™	NSX Advanced Threat Prevention
VMware vSphere® Distributed Resource Scheduler™	vSphere DRS・DRS	VMware Aria Automation™	VMware Aria Automation	VMware Tanzu® CloudHealth®	Tanzu CloudHealth	VMware NSX® Distributed Firewall™	NSX Distributed Firewall
VMware vSphere® Distributed Switch™	vSphere Distributed Switch・VDS	VMware Aria Automation Orchestrator™	VMware Aria Automation Orchestrator	VMware Tanzu® for Kubernetes Operations	Tanzu for Kubernetes Operations	VMware NSX® Distributed IDS/IPS™	NSX Distributed IDS/IPS
VMware vSphere® Fault Tolerance	vSphere Fault Tolerance ・vSphere FT	VMware Aria Operations™	VMware Aria Operations	VMware Tanzu® GemFire®	Tanzu GemFire	VMware NSX® Gateway Firewall™	NSX Gateway Firewall
VMware vSphere® High Availability	vSphere HA	VMware Aria Operations™ for Applications	VMware Aria Operations for Applications	VMware Tanzu® Greenplum®	Tanzu Greenplum	VMware NSX® Intelligence™	NSX Intelligence
VMware vSphere® vMotion®	vSphere vMotion	VMware Aria Operations™ for Logs	VMware Aria Operations for Logs	VMware Tanzu® Guardrails	Tanzu Guardrails	VMware NSX® NDR™	NSX NDR
VMware vSphere® with Kubernetes	vSphere with Kubernetes	VMware Aria Operations™ for Networks	VMware Aria Operations for Networks	VMware Tanzu® Kubernetes Grid™	Tanzu Kubernetes Grid	VMware NSX® Service Mesh™	NSX Service Mesh
VMware vSphere® with VMware Tanzu®	vSphere with Tanzu	VMware Aria Suite™	VMware Aria Suite	VMware Tanzu® Hub™	Tanzu Hub	VMware DPU-based Acceleration™ for VMware NSX®	VMware DPU-based Acceleration
VMware vCenter®	vCenter	VMware Aria Universal Suite™	VMware Aria Universal Suite	VMware Tanzu Labs™	Tanzu Labs	VMware HCX®	VMware HCX
VMware vSAN™	vSAN	VMware Aria vCloud Suite™	VMware Aria vCloud Suite	VMware Tanzu® Mission Control™	Tanzu Mission Control	VMware Container Networking™ with Antrea™	VMware Container Networking
VMware Edge Compute Stack™	VMware Edge Compute Stack	VMware Skyline™	VMware Skyline	VMware Tanzu® RabbitMQ®	Tanzu RabbitMQ	VMware Cloud Foundation™	VMware Cloud Foundation
VMware Edge Network Intelligence™	VMware Edge Network Intelligence	VMware Horizon®	Horizon	VMware Tanzu® Service Mesh™ Advanced Edition	Tanzu Service Mesh	VMware Cloud™ on AWS	VMware Cloud on AWS
VMware Secure Access™	VMware Secure Access	VMware Horizon® Cloud Service™	Horizon Cloud	VMware Workspace ONE®	Workspace ONE	Azure VMware® Solution	Azure VMware Solution
VMware Marketplace™	VMware Marketplace	VMware Horizon® Cloud Service™ on IBM Cloud	Horizon Cloud Service on IBM Cloud	VMware Workspace ONE® Access™	Workspace ONE Access	Google Cloud VMware® Engine	Google Cloud VMware Engine
VMware® Integrated OpenStack	VMware Integrated OpenStack	VMware Horizon® Cloud Service™ on Microsoft Azure	Horizon Cloud Service on Microsoft Azure	VMware Workspace ONE® Assist™	Workspace ONE Assist	Oracle Cloud VMware® Solution	Oracle Cloud VMware Solution
VMware Carbon Black Cloud™	VMware Carbon Black Cloud	VMware Horizon® Accelerator™	Horizon Accelerator	VMware Workspace ONE® Boxer	Workspace ONE Boxer	IBM Cloud® for VMware Solutions™	IBM Cloud for VMware Solutions
VMware Carbon Black App Control™	Carbon Black App Control	VMware Horizon® Apps	Horizon Apps	VMware Workspace ONE® Intelligence™	Workspace ONE Intelligence	VMware Cloud Disaster Recovery™	VMware Cloud Disaster Recovery
VMware Carbon Black Container™	Carbon Black Container	VMware App Volumes™	App Volumes	VMware Workspace ONE® Intelligence™ for Consumer Apps	Workspace ONE Intelligence for Consumer Apps	VMware Cloud Flex Storage™	VMware Cloud Flex Storage
VMware Carbon Black Endpoint™	Carbon Black Endpoint	VMware Dynamic Environment Manager™	VMware Dynamic Environment Manager	VMware Workspace ONE® Intelligent Hub	Workspace ONE Intelligent Hub	VMware Cloud Director®	VMware Cloud Director
VMware Carbon Black Workload™	Carbon Black Workload	VMware Tanzu®	VMware Tanzu	VMware Workspace ONE® Launcher™	Workspace ONE Launcher	VMware Cloud Universal™	VMware Cloud Universal
VMware SASE™	VMware SASE	VMware Tanzu® Application Catalog ™	Tanzu Application Catalog	VMware Workspace ONE® UEM	Workspace ONE UEM	VMware Cloud Packs™	VMware Cloud Packs
VMware SD-WAN™	VMware SD-WAN	VMware Tanzu® Application Platform™	Tanzu Application Platform	VMware NSX®	NSX		
		VMware Tanzu® Application Service™	Tanzu Application Service	VMware NSX® Advanced Load Balancer™	NSX Advanced Load Balancer		

あなたの組織で、アプリのデプロイ頻度はどのくらいですか？

1. １日数回
2. 週に数回
3. 月に数回
4. 年に数回
5. 初回のみ

あなたの組織で、アプリ初回デプロイまでのリードタイムはどのくらいですか？

1. 数分
2. 数時間
3. 数日
4. 数週間
5. 数ヶ月

Agenda

1. デプロイし続けられる必要性
2. 開発者から見た Tanzu Application Platform
3. 運用者から見た Tanzu Application Platform

Agenda

1. **デプロイし続けられる必要性**
2. 開発者から見た Tanzu Application Platform
3. 運用者から見た Tanzu Application Platform

デプロイ != リリース

デプロイ

ソフトウェアのある状態を特定の環境に配置し、利用可能な状態にすること。

ユーザー体験に対する変更がなくても良い。

リリース

全て、または特定のユーザーに対して、新たなユーザー体験 (新機能、バグFixなど) を利用可能な状態にすること。

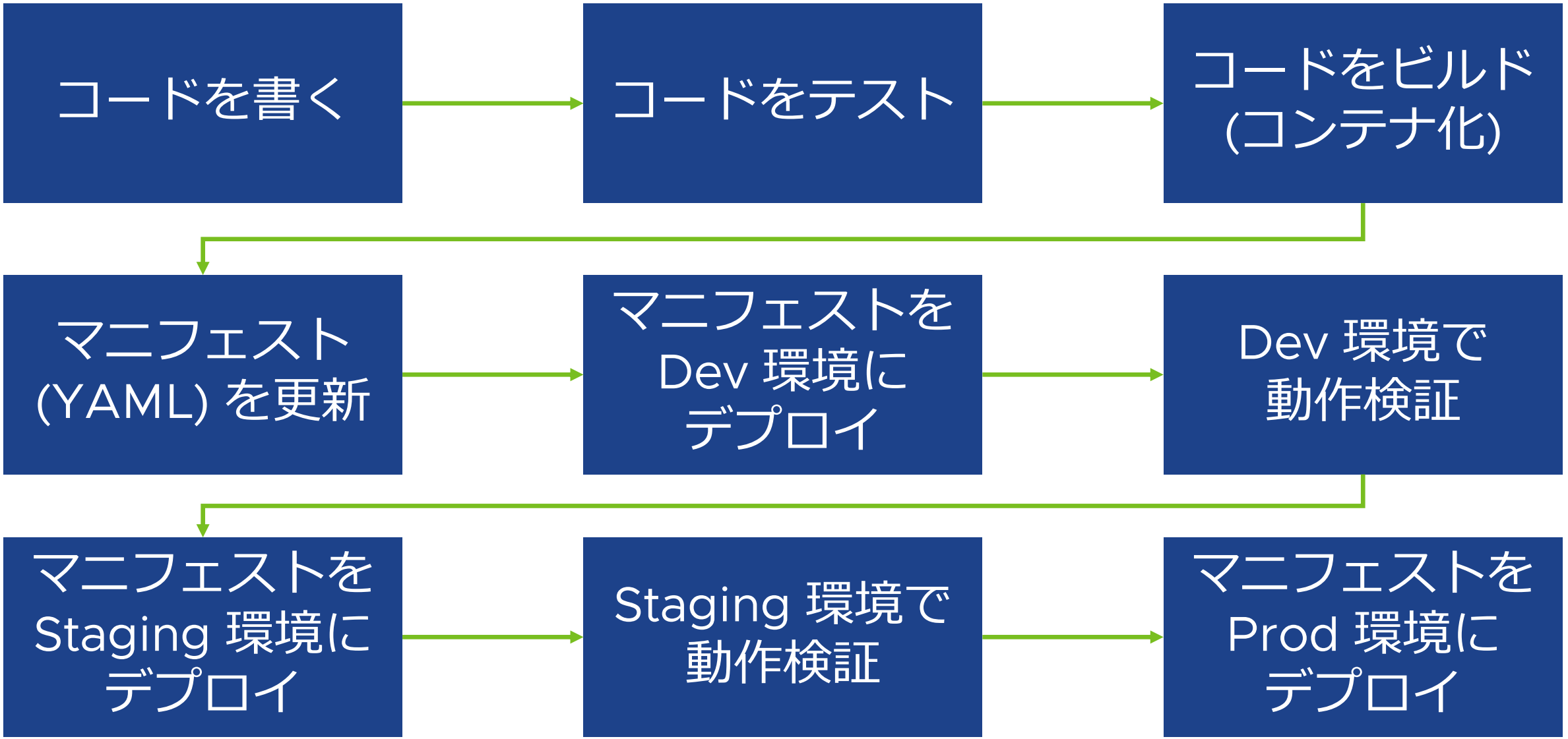
デプロイ != リリース

リリースしないからと言って、技術的にデプロイできる状態にしなくて良いわけではない

緊急度の高いバグ、
セキュリティ対応

今日、ユーザーに届けることで
企業価値を上げられる新機能

"素朴な"ソフトウェアデリバリーのフロー



"素朴な"ソフトウェアデリバリーのフロー

初回のセットアップが大変
(アカウントの設定、ネットワークの設定、環境構築、Dockerfile、IaC?)

継続的にデプロイする仕組みを作るのが大変
(CI/CD、モニタリングなど)

企業のコンプライアンス遵守のための設定...

デプロイ

デプロイ

開発者視点での “ユーザーストーリー”

As a developer,
I want to deploy my app continuously.
So that I can deliver values to customers fast forever.

出典元：https://en.wikipedia.org/wiki/User_story

開発者視点での “ユーザーストーリー”

開発者として、
アプリをデプロイしたい、継続的に。
顧客に価値を素早く提供し続けられるため。

出典元：https://en.wikipedia.org/wiki/User_story

NEWLY EXPANDED

VMware Tanzu® Application Platform™

Backstage CARVEL TEKTON

CARTOGRAPHER Buildpacks.io flux

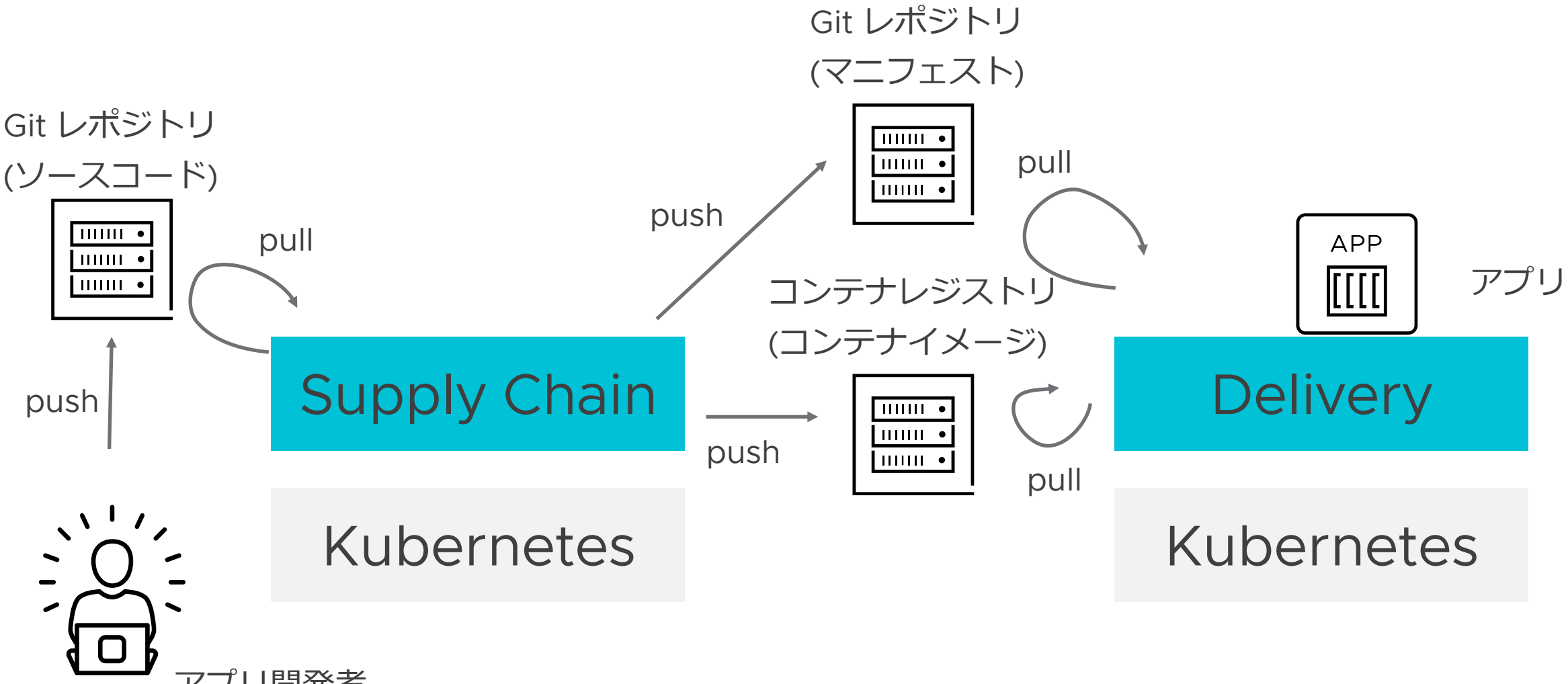
- アプリケーションを一貫してデリバリおよびデプロイする
- 開発者のエクスペリエンスを向上させ、生産性を向上させる
- アプリケーションの継続的なセキュリティを確保する
- Kubernetes 上のプラットフォーム。クラウド上でもオンプレ上でも。主要なディストリビューションをサポート。

vmware®

Agenda

1. デプロイし続けられる必要性
2. **開発者から見た Tanzu Application Platform**
3. 運用者から見た Tanzu Application Platform

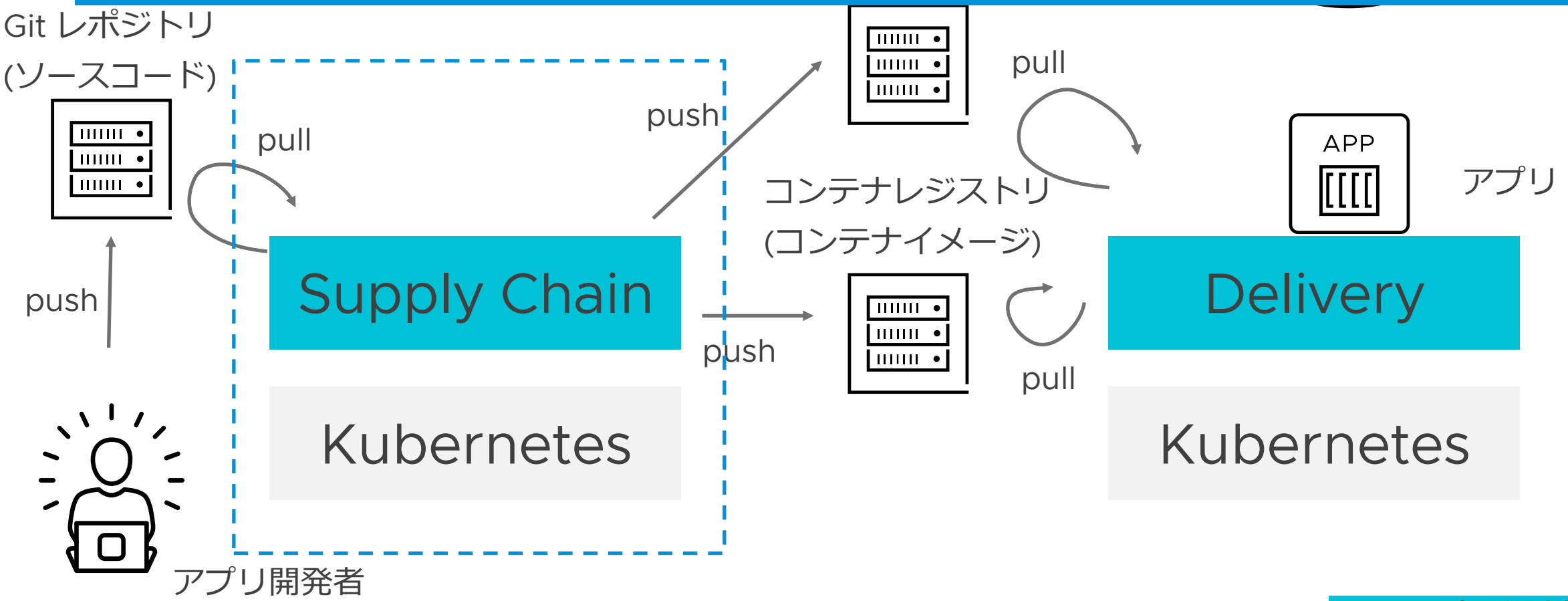
Tanzu Application Platform によるソフトウェアデリバリー



TAP 提供機能

Tanzu Application Platform によるソフトウェアデリバリー

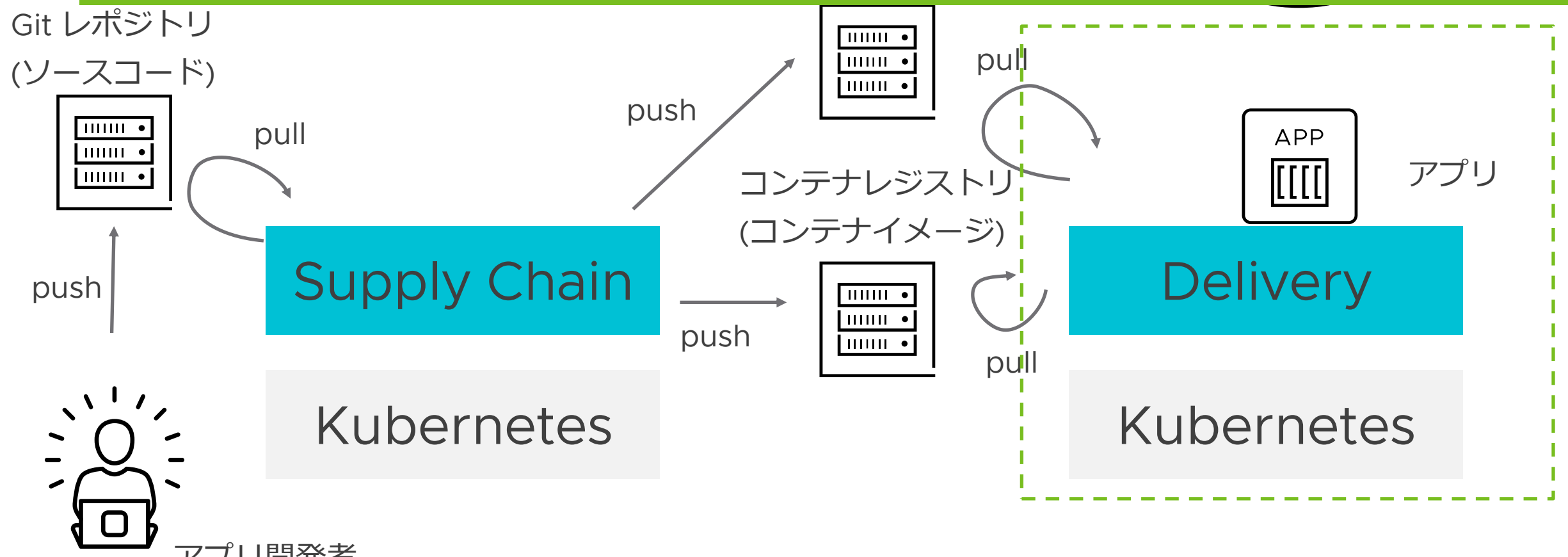
アプリのコンテナイメージと K8s マニフェストの生成し push する、CI相当の作業を行う K8s クラスタ (Build クラスタ)



TAP 提供機能

Tanzu Application Platform によるソフトウェアデリバリー

生成されたイメージとマニフェストでアプリをデプロイする、
CD 相当の作業を行う K8s クラスタ (Run クラスタ)



TAP提供機能

Tanzu Application Platform を使ったアプリのシンプルなデプロイ

```
tanzu apps workload apply hello-tap ¥
  --app hello-tap ¥
  --git-repo https://github.com/making/hello-tap ¥
  --git-branch main ¥
  --type web ¥
  -n demo
```


tmaki — -zsh — 94×12
$ curl https://hello-tap.demo.dev.tap.maki.lol
running on hello-tap-00001-deployment-64cb8bb896-cjf8z
$

Tanzu Application Platform でできること

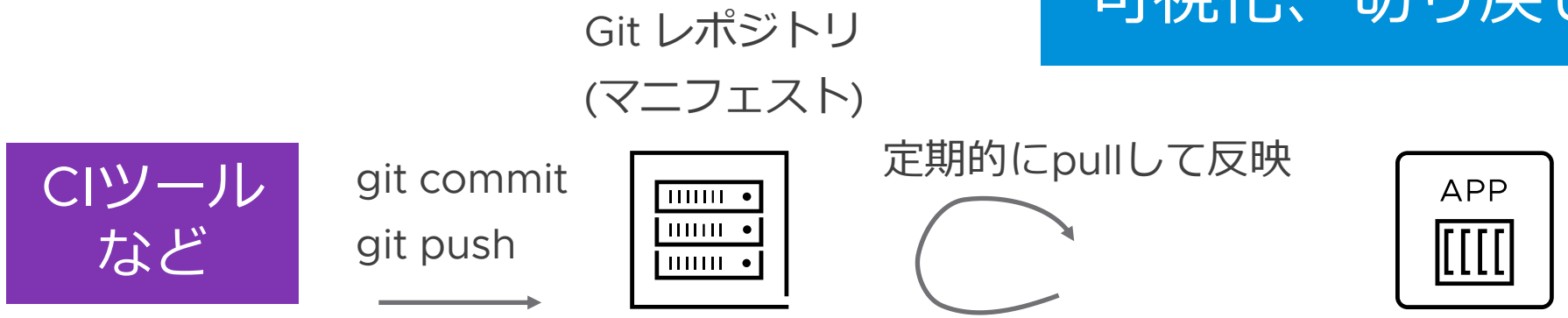
- 初回のデプロイが1 (または数) コマンドで完了
- 二回目以降のデプロイは GitOps により自動化
- 環境のプロモーション (Dev → Staging → Prod) を Git の操作のみで行える

プラットフォームチームが整備してくれた道
（“Golden Path” by Spotify）上に簡単に、継続的にデプロイできる

GitOps

ソフトウェアデリバリーのアプローチの１つ。
マニフェスト (YAML) の変更を Git で管理し、Git を定期的に pull してデプロイすることで、Git 上のマニフェストの状態と実際にデプロイされているアプリの状態を同期させる。

変更のトラッキング、
可視化、切り戻しが容易



GitOpsによるアプリのデプロイ (初回)

```
tanzu apps workload apply hello-tap ¥
  --app hello-tap ¥
  --git-repo https://github.com/making/hello-tap ¥
  --git-branch main ¥
  --type web ¥
  --param gitops_branch=main ¥
  --param gitops_server_address=https://github.com ¥
  --param gitops_repository_owner=making ¥
  --param gitops_repository_name=tap-gitops-manifests ¥
  --param gitops_ssh_secret=git-basic ¥
  -n demo
```

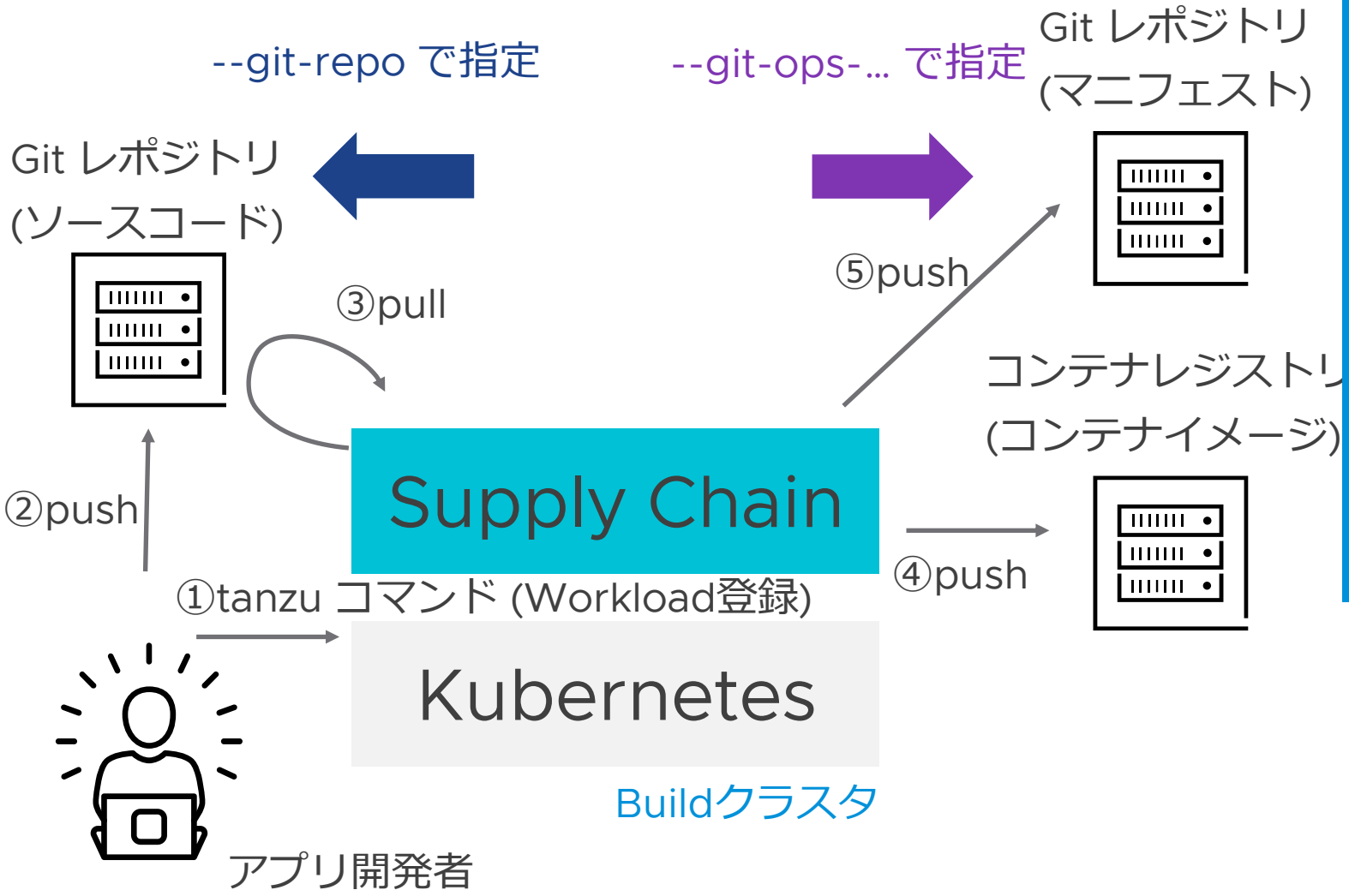
GitOpsによるアプリのデプロイ (初回)

```
tanzu apps workload apply hello-tap ¥
  --app hello-tap ¥
  --git-repo https://github.com/making/hello-tap ¥
  --git-branch main ¥
  --type web ¥
  --param gitops_branch=main ¥
  --param gitops_server_address=https://github.com ¥
  --param gitops_repository_owner=making ¥
  --param gitops_repository_name=tap-gitops-manifests ¥
  --param gitops_ssh_secret=git-basic ¥
  -n demo
```

取得するソースコードの場所

生成されるマニフェストの置き場所

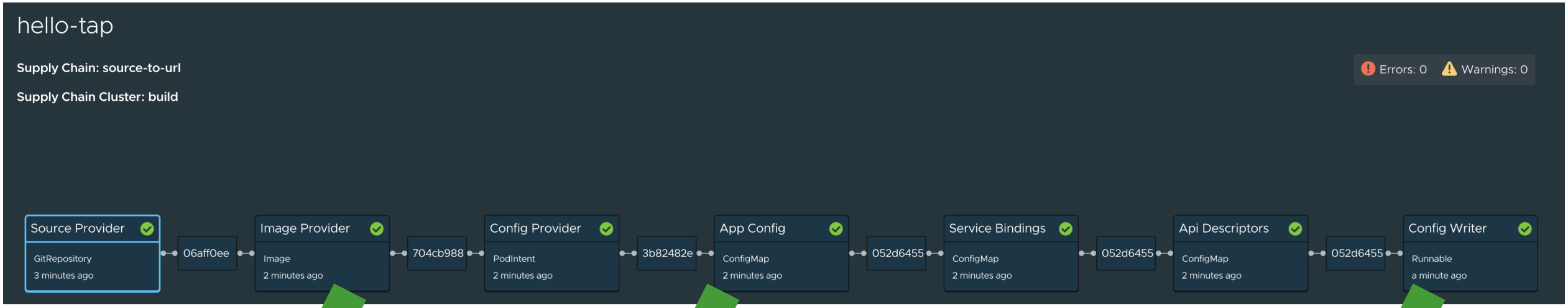
Tanzu Application Platformによるソフトウェアデリバリー



tanzu コマンドで
Workload が登録されれば、
以降ソースコードを
push するだけで、
コンテナイメージと
マニフェストが自動で更新
される

①の作業は初回だけ

Developer Portalで状態確認



コードをビルド (コンテナ化)

マニフェスト (YAML) を更新

マニフェストを git レポジトリに push

デモ

1. Build クラスタに Workload の登録すると、コンテナイメージがビルドされ、マニフェストが git push される

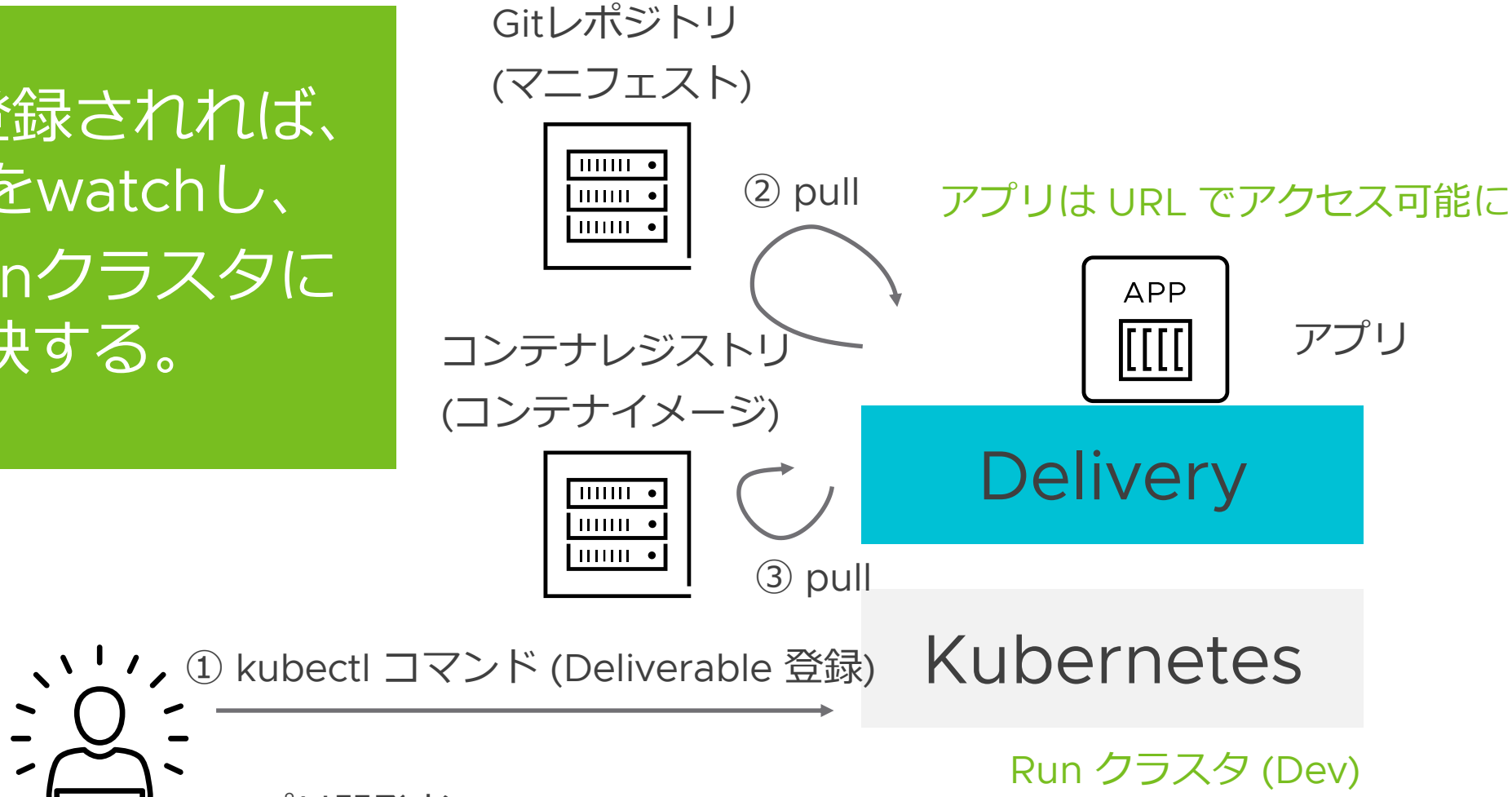
https://youtu.be/F1OIMSHK6s4

```
(~/tap/1.6.3)[tap-build-admin] $
```

Tanzu Application Platform によるソフトウェアデリバリー

Deliverableが登録されれば、
マニフェストをwatchし、
gitの状態をRunクラスタに
自動で反映する。

①の作業は初回だけ



Run クラスタに投入するマニフェスト (Deliverable)

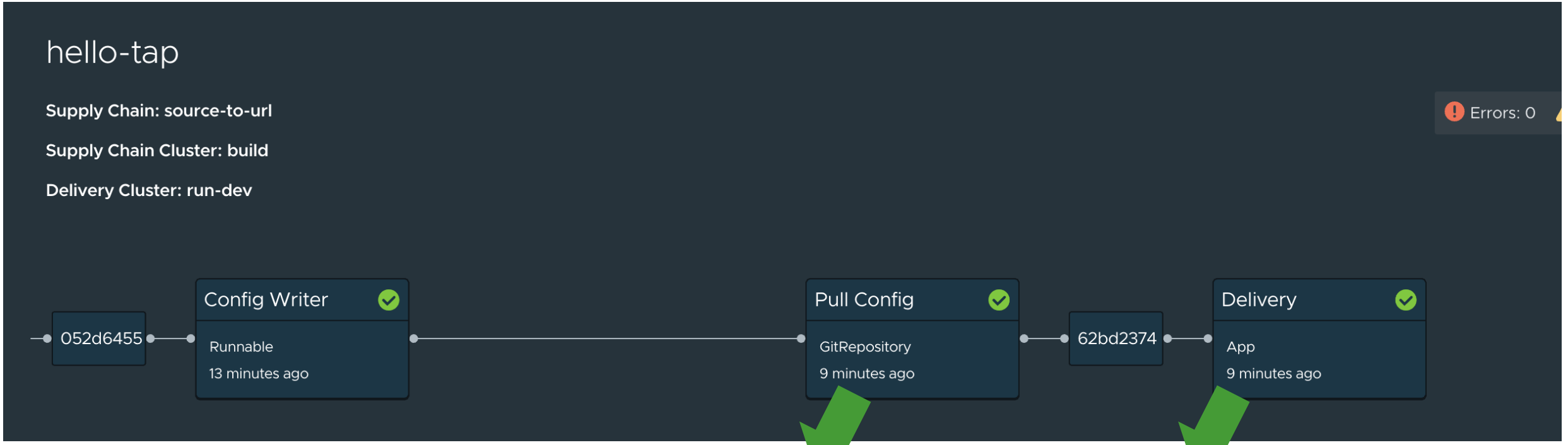
```
apiVersion: carto.run/v1alpha1
kind: Deliverable
metadata:
  name: hello-tap
  labels:
    app.kubernetes.io/part-of: hello-tap
    app.tanzu.vmware.com/deliverable-type: web
    carto.run/workload-name: hello-tap
    carto.run/workload-namespace: demo
spec:
  params:
  - name: gitops_ssh_secret
    value: git-basic
  source:
    git:
      url: https://github.com/making/tap-gitops-manifests.git
      ref:
        branch: main
    subPath: config/demo/hello-tap
```

Run クラスタに投入するマニフェスト (Deliverable)

```
apiVersion: carto.run/v1alpha1
kind: Deliverable
metadata:
  name: hello-tap
  labels:
    app.kubernetes.io/part-of: hello-tap
    app.tanzu.vmware.com/deliverable-type: web
    carto.run/workload-name: hello-tap
    carto.run/workload-namespace: demo
spec:
  params:
  - name: gitops_ssh_secret
    value: git-basic
  source:
    git:
      url: https://github.com/making/tap-gitops-manifests.git
      ref:
        branch: main
    subPath: config/demo/hello-tap
```

デプロイするアプリのマニフェストの場所

Developer Portal で状態確認



マニフェストを git レポジトリから pull

マニフェストを Dev 環境にデプロイ

デモ

1. Run クラスタに Deliverable を登録すると、マニフェストが pull され、アプリがデプロイされる。
2. Git 上のソースコードを更新すると、コンテナイメージとマニフェストが更新され、新しいバージョンのアプリがデプロイされる

https://youtu.be/TdVtiHB2fog

```
(~/tap/1.6.3)[tap-run-dev-admin] $
```

Namespace demo

Tasks

Id
5d8c1ec5-29d9-4b32-ab3b-3026e9fb15bf

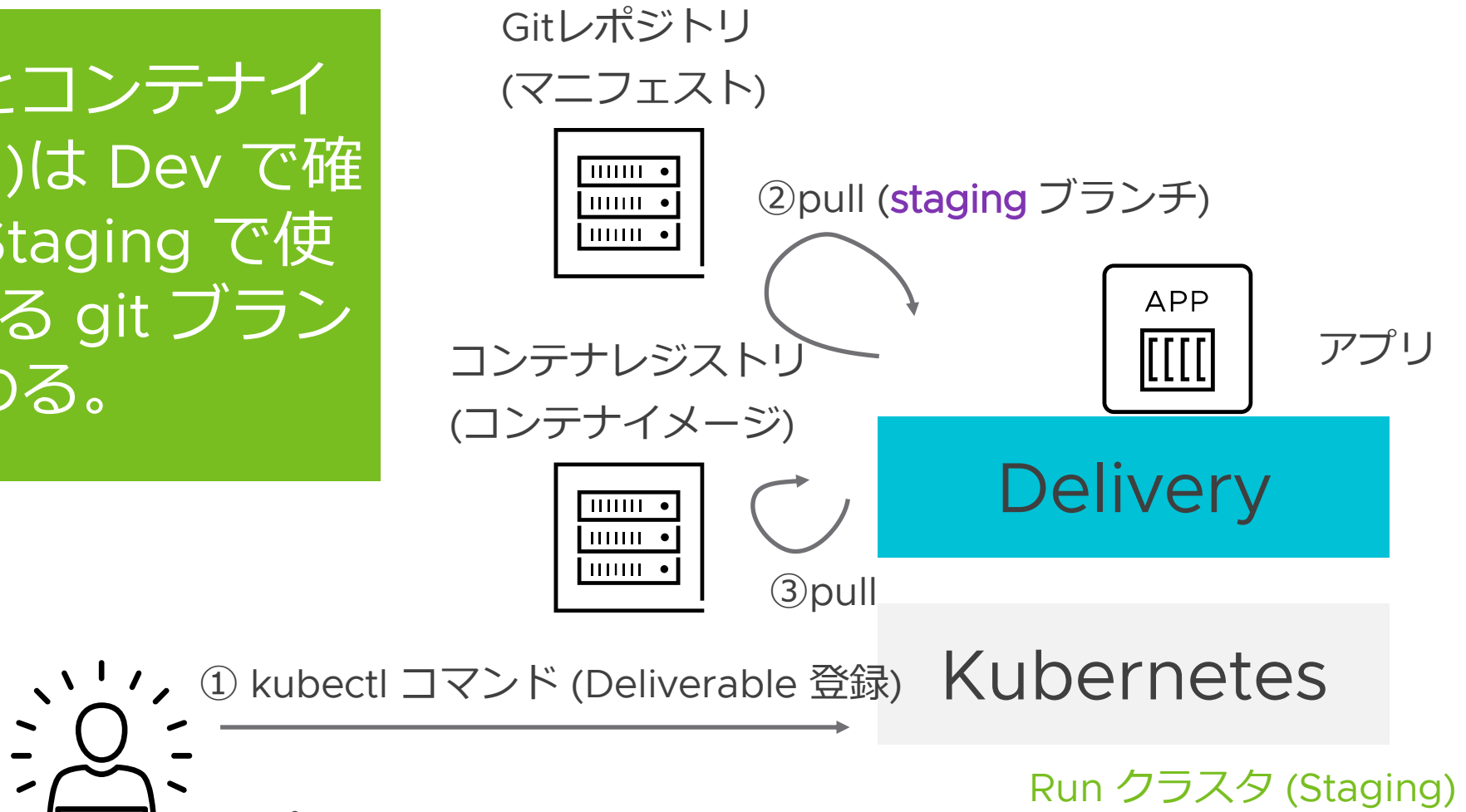
ERROR- TIMEOUT | gp | git clone https://github.com/rd-1-2022/ai-openai-h | demo.txt | SELECT right(bytea_val::text, -1)::bit(64)::bigint

```
42 ①·Deliverableの確認
43
44 bat·deliverable-dev.yaml
45
46 ②·Deliverableの登録
47
48 kubectl·apply·-f·deliverable-dev.yaml·-n·demo
49
50 ③·Developer·Portalの確認
51
52 https://tap-gui.view.tap.maki.lol
53
54 ④·アプリにアクセス
55
56 curl·https://hello-tap.demo.dev.tap.maki.lol
57
58 ⑤·ソースコードを変更
59
60 https://github.com/making/hello-tap
61
62 ⑥·Developer·Portalの確認
63
64 https://tap-gui.view.tap.maki.lol
```

Line 42, Column 17 Spaces: 2 Plain Text

Staging 環境へのデプロイ

マニフェストとコンテナイメージ(バイナリ)は Dev で確認したものを Staging で使用する。参照する git ブランチが変わる。

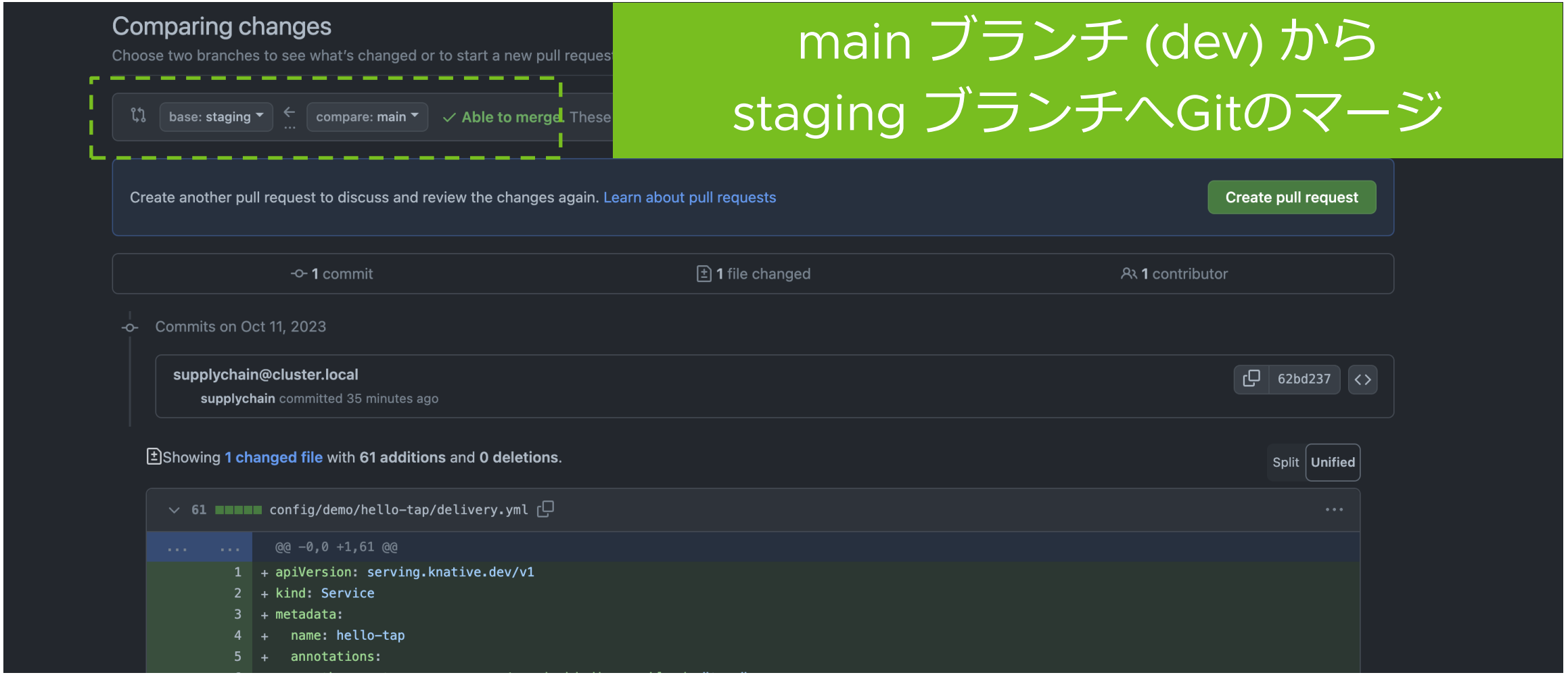


Run (Staging) クラスタに投入するマニフェスト (Deliverable)

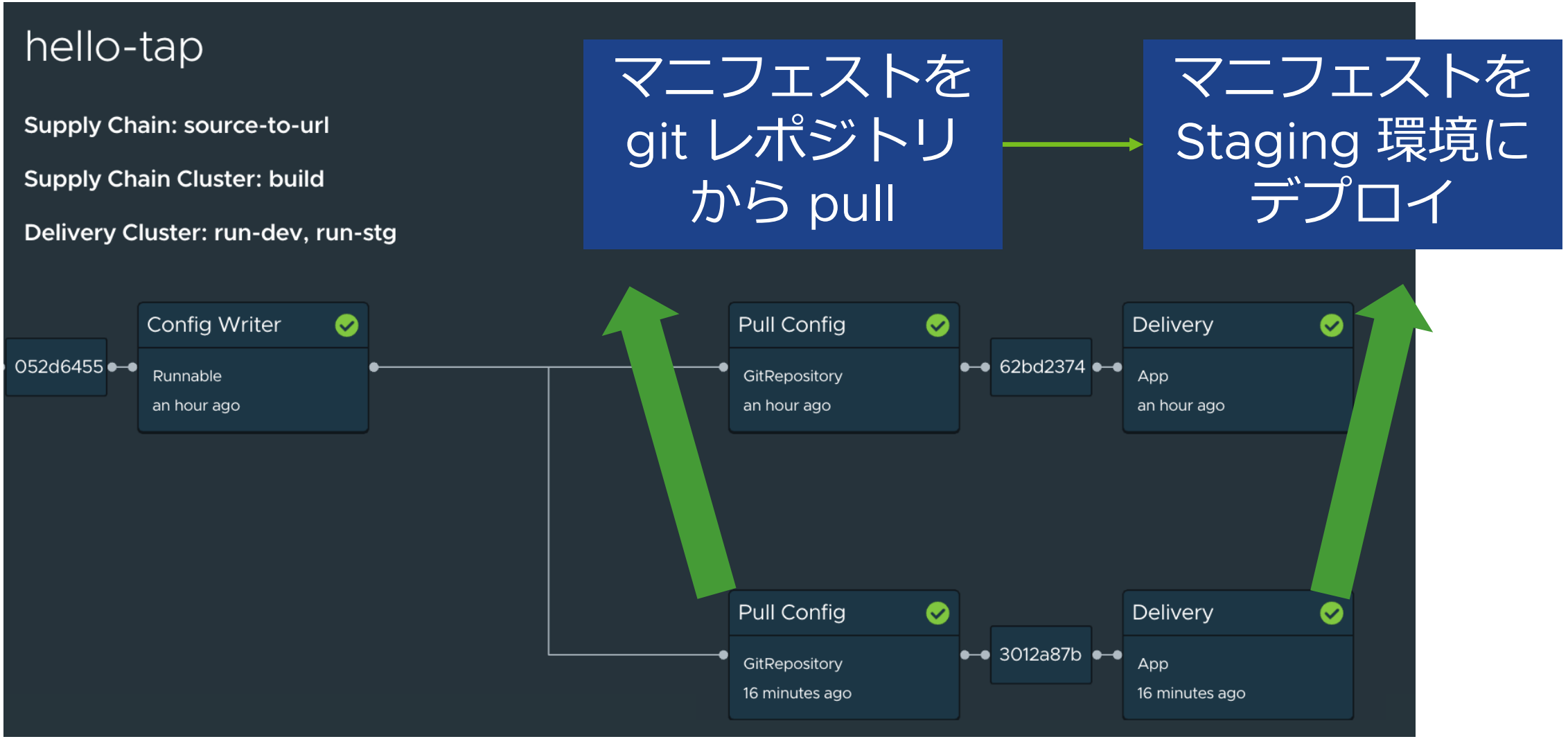
```
apiVersion: carto.run/v1alpha1
kind: Deliverable
metadata:
  name: hello-tap
  labels:
    app.kubernetes.io/part-of: hello-tap
    app.tanzu.vmware.com/deliverable-type: web
    carto.run/workload-name: hello-tap
    carto.run/workload-namespace: demo
spec:
  params:
  - name: gitops_ssh_secret
    value: git-basic
  source:
    git:
      url: https://github.com/making/tap-gitops-manifests.git
      ref:
        branch: staging
    subPath: config/demo/hello-tap
```

Staging 用のブランチに向き先を変える

Dev → Staging へのプロモーション



Developer Portal で状態確認



デモ

1. Staging 環境の Run クラスタに Deliverable を登録し、マニフェストを main ブランチから staging ブランチにマージすると、 Staging 環境にアプリがデプロイされる

https://youtu.be/u-MuyEHmDIk

```
(~/tap/1.6.3)[tap-run-stg-admin] $
```

ERROR- TIMEOUT | gp | git clone https://github.com/rd-1-2022/ai-openai-h | demo.txt | SELECT right(bytea_val::text, -1)::bit(64)::bigint

```
79 kubectl config use-context tap-run-stg-admin
80
81 ① Deliverableの確認
82
83 bat deliverable-stg.yaml
84
85 ② Deliverableの登録
86
87 kubectl apply -f deliverable-stg.yaml -n demo
88
89 ③ Dev -> Staging
90
91 Githubでpull request
92
93 ④ Developer Portalの確認
94
95 https://tap-gui.view.tap.maki.lol
96
97 ⑤ アプリにアクセス
98
99 curl https://hello-tap.demo.stg.tap.maki.lol
100
101
102
```

Line 84, Column 1 | Spaces: 2 | Plain Text

Stage Detail: Source Provide

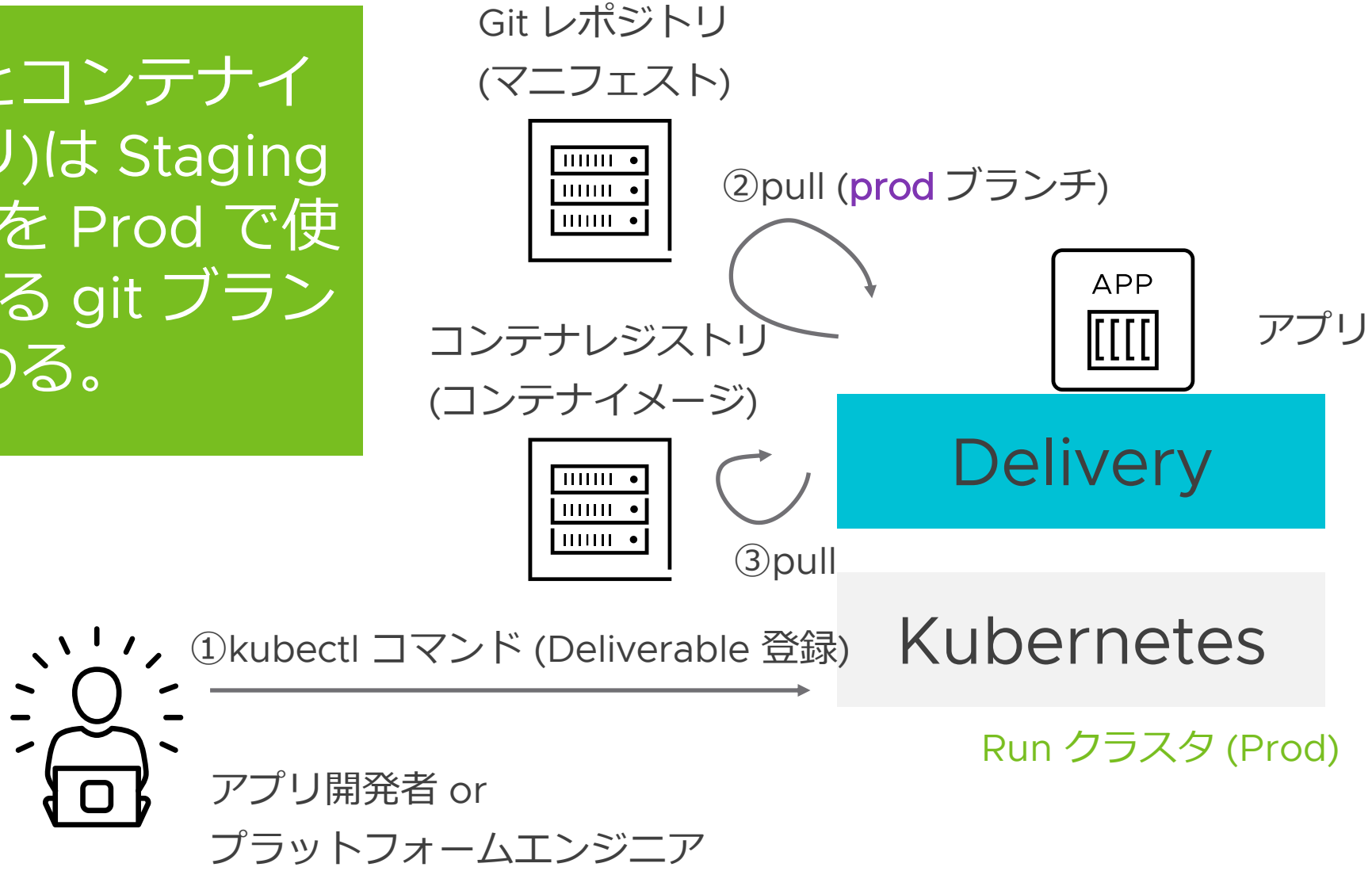
3 minutes ago

Overview

Name	hello-tap
Kind	GitRepository
Source URL	https://github.com/making/
Version	source.toolkit.fluxcd.io/v1bet
Namespace	demo

Prod 環境へのデプロイ

マニフェストとコンテナイメージ(バイナリ)は Staging で確認したものを Prod で使用する。参照する git ブランチが変わる。



Run (Prod) クラスタに投入するマニフェスト (Deliverable)

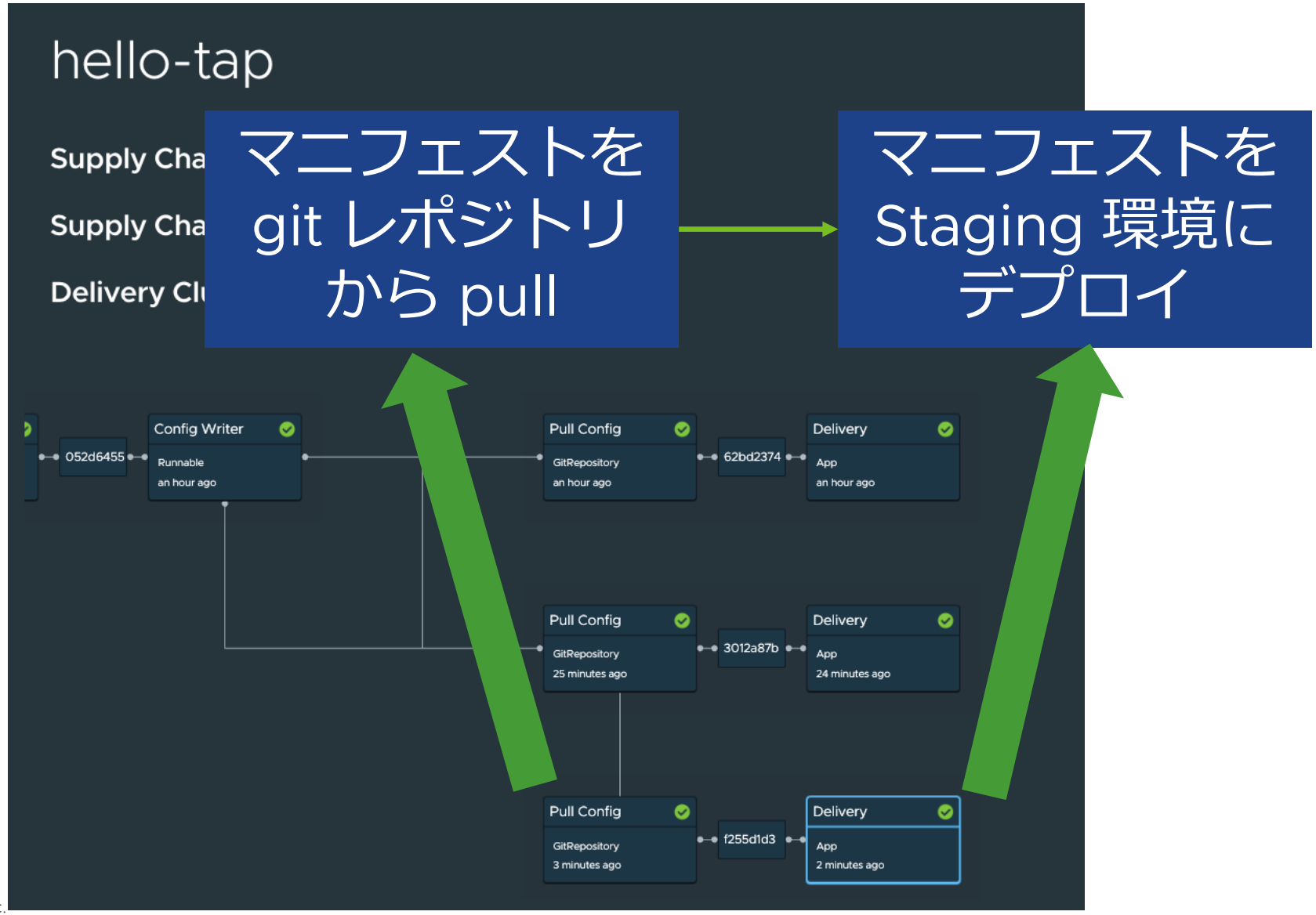
```
apiVersion: carto.run/v1alpha1
kind: Deliverable
metadata:
  name: hello-tap
  labels:
    app.kubernetes.io/part-of: hello-tap
    app.tanzu.vmware.com/deliverable-type: web
    carto.run/workload-name: hello-tap
    carto.run/workload-namespace: demo
spec:
  params:
  - name: gitops_ssh_secret
    value: git-basic
  source:
    git:
      url: https://github.com/making/tap-gitops-manifests.git
      ref:
        branch: prod
    subPath: config/demo/hello-tap
```

Prod 用のブランチに向き先を変える

Staging → Prod へのプロモーション

Staging ブランチから
Dev ブランチへ Git のマージ

Developer Portal で状態確認



デモ

1. Prod 環境の Run クラスタに Deliverable を登録し、マニフェストを staging ブランチから prod ブランチにマージすると、Prod 環境にアプリがデプロイされる

https://youtu.be/EkNiKg2NtKI

```
(~/tap/1.6.3)[tap-run-prd-admin] $
```

ERROR- TIMEOUT | gp | git clone https://github.com/rd-1-2022/ai-openai-h | demo.txt | SELECT right(bytea_val::text, -1)::bit(64)::bigint

```
109
110 ①·Deliverableの確認
111
112 bat·deliverable-prd.yaml
113
114 ②·Deliverableの登録
115
116 kubectl·apply·-f·deliverable-prd.yaml·-n·demo
117
118 ③·Staging·->·Prod
119
120 Githubでpull·request
121
122 ④·Developer·Portalの確認
123
124 https://tap-gui.view.tap.maki.lol
125
126 ⑤·アプリにアクセス
127
128 curl·https://hello-tap.demo.stg.tap.maki.lol
129
130
131
```

2 lines, 45 characters selected | Spaces: 2 | Plain Text

開発者が TAP 上で継続的にデプロイするために必要なこと、まとめ

サーバや、
CI/CD パイプラインの
セットアップは不要！

- **初回デプロイ時**

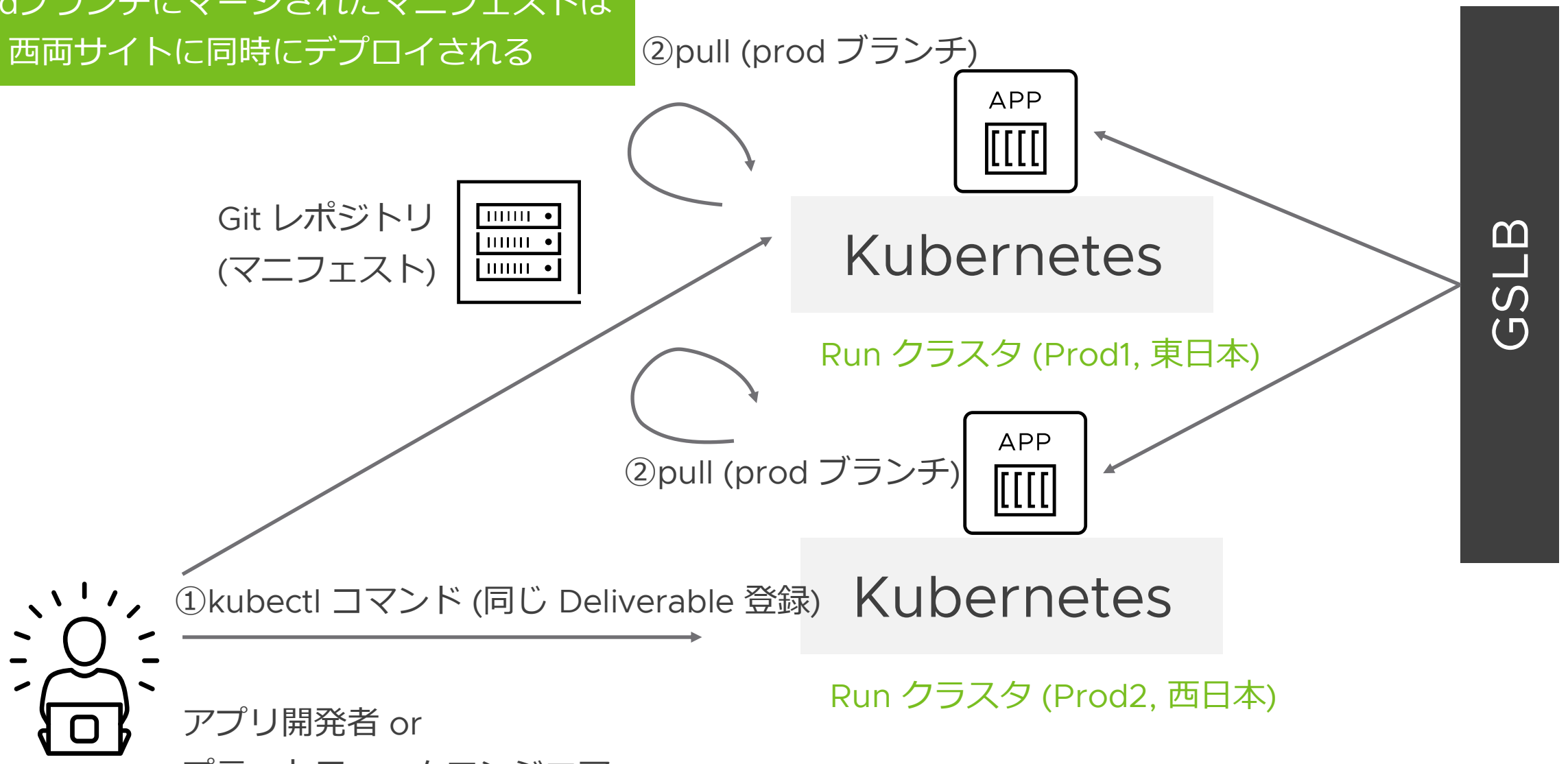
1. Build クラスタに Workload を設定
2. 対象のRunクラスタに Deliverable を設定
   (プラットフォームエンジニアがやっても良い)

- **二回目以降のデプロイ**

1. Git にコードを push
2. 対象の Run クラスタのブランチに git をマージ

応用編: 2サイトの Prod 環境に同時にデプロイ

Prodブランチにマージされたマニフェストは
東・西両サイトに同時にデプロイされる



デモ

1. Prod2 環境の Run クラスタに Deliverable を登録する
2. 東日本と西日本それぞれからアプリにアクセスすると、地理的に近いクラスタにルーティングされる
3. 東日本の Prod 環境に障害を起こし、西日本の Prod2 環境へフェールオーバーする

※ デモでは GSLB として、 Azure の Cross Region Load Balancer を使用

AKS での設定方法

https://youtu.be/R7W9EBhkbko

```
(~/tap/1.6.3)[tap-run-prd2-admin] $
```

ERROR- TIMEOUT | gp | git clone https://github.com/rd-1-2022/ai-openai-h | demo.txt | SELECT right(bytea_val::text, -1)::bit(64)::bigint

```
138 ① Deliverableの確認
139
140 bat deliverable-prd.yaml
141
142 ② Deliverableの登録
143
144 kubectl apply -f deliverable-prd.yaml -n demo
145
146 ③ Developer Portalの確認
147
148 https://tap-gui.view.tap.maki.lol
149
150 ④ PodとURLを確認
151
152 Prod1、東日本
153 kubectl get pod,ksvc -n demo --context tap-run-prd-admin
154
155 Prod2、西日本
156 kubectl get pod,ksvc -n demo --context
    tap-run-prd2-admin
157
158 ⑤ アプリにアクセス
159
```

16 characters selected | Spaces: 2 | Plain Text

あなたの組織で、アプリのデプロイ頻度はどのくらいですか？

1. １日数回
2. 週に数回
3. 月に数回
4. 年に数回
5. 初回のみ

TAPなら！

あなたの組織で、アプリ初回デプロイまでのリードタイムはどのくらいですか？

1. 数分
2. 数時間
3. 数日
4. 数週間
5. 数ヶ月

TAPなら！

Tanzu Application Platform を使うことで、
素早く、継続的に、顧客へ価値を届け続けられる！

"Developer Sandbox" でまずはお試し

アプリ開発者として**無償**で何度でも試せる

短命な**マネージド** TAP (寿命8時間)

https://tanzu.academy/guides/developer-sandbox

Agenda

1. デプロイし続けられる必要性
2. 開発者から見た Tanzu Application Platform
3. **運用者から見た Tanzu Application Platform**

運用者が開発者に求めるソフトウェアデリバリー

テストされたソースコード!

脆弱性のないソースコード!

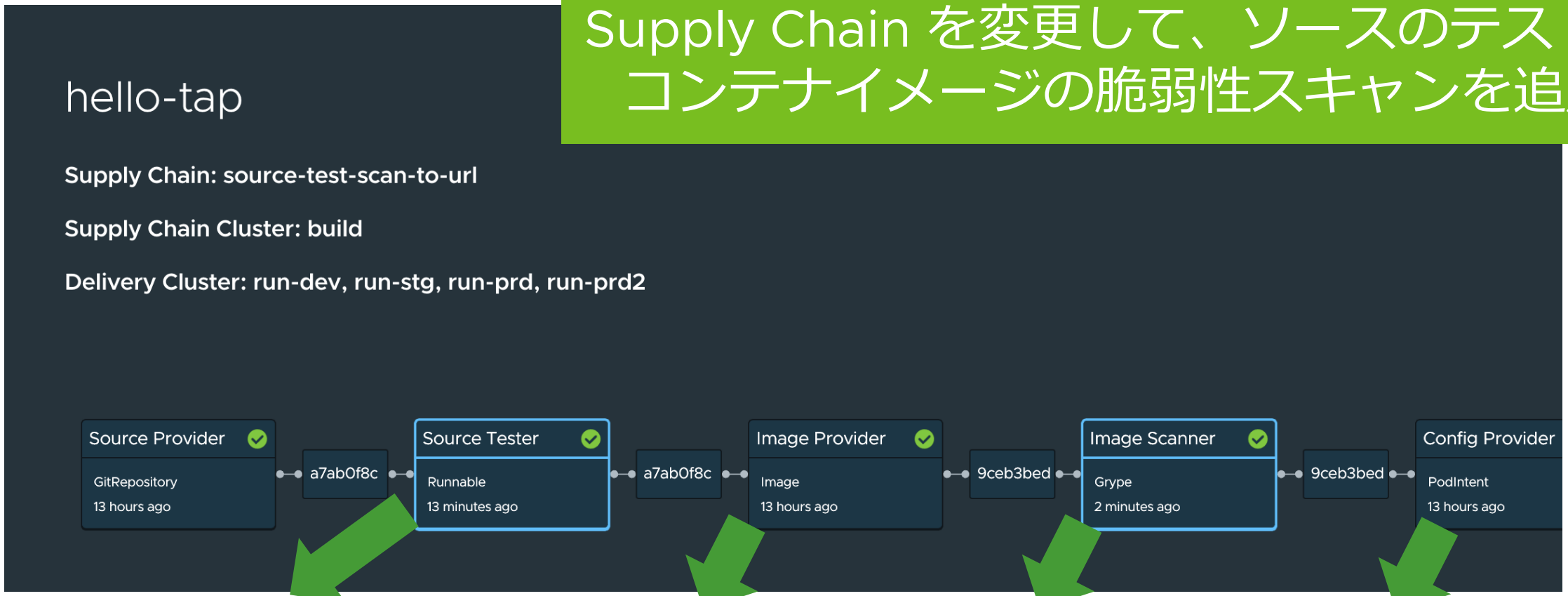
セキュアなコンテナイメージ!

社内標準のモニタリング!

…

Secure Path to Production

Supply Chain を変更して、ソースのテストとコンテナイメージの脆弱性スキャンを追加

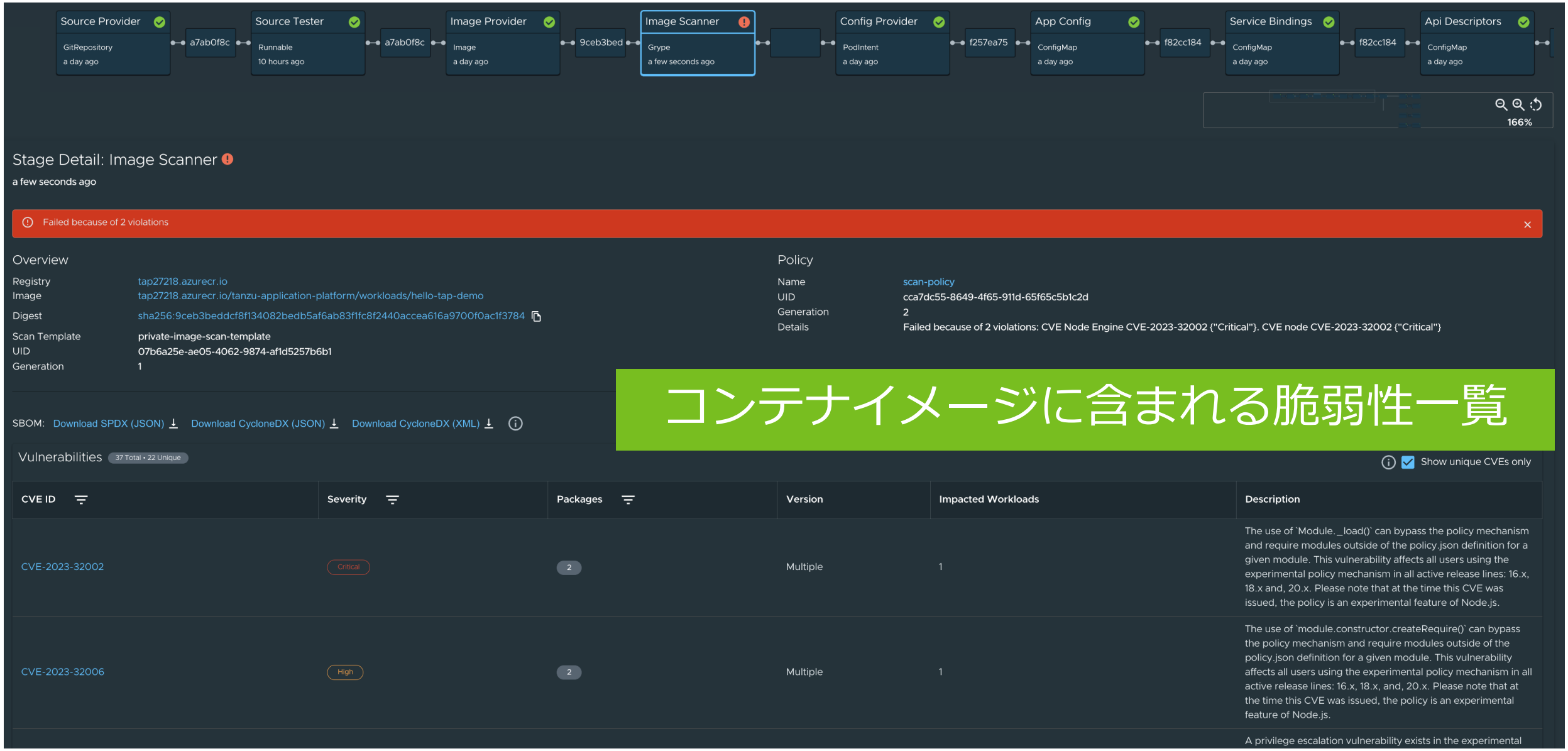


コードをテスト

コードをビルド (コンテナ化)

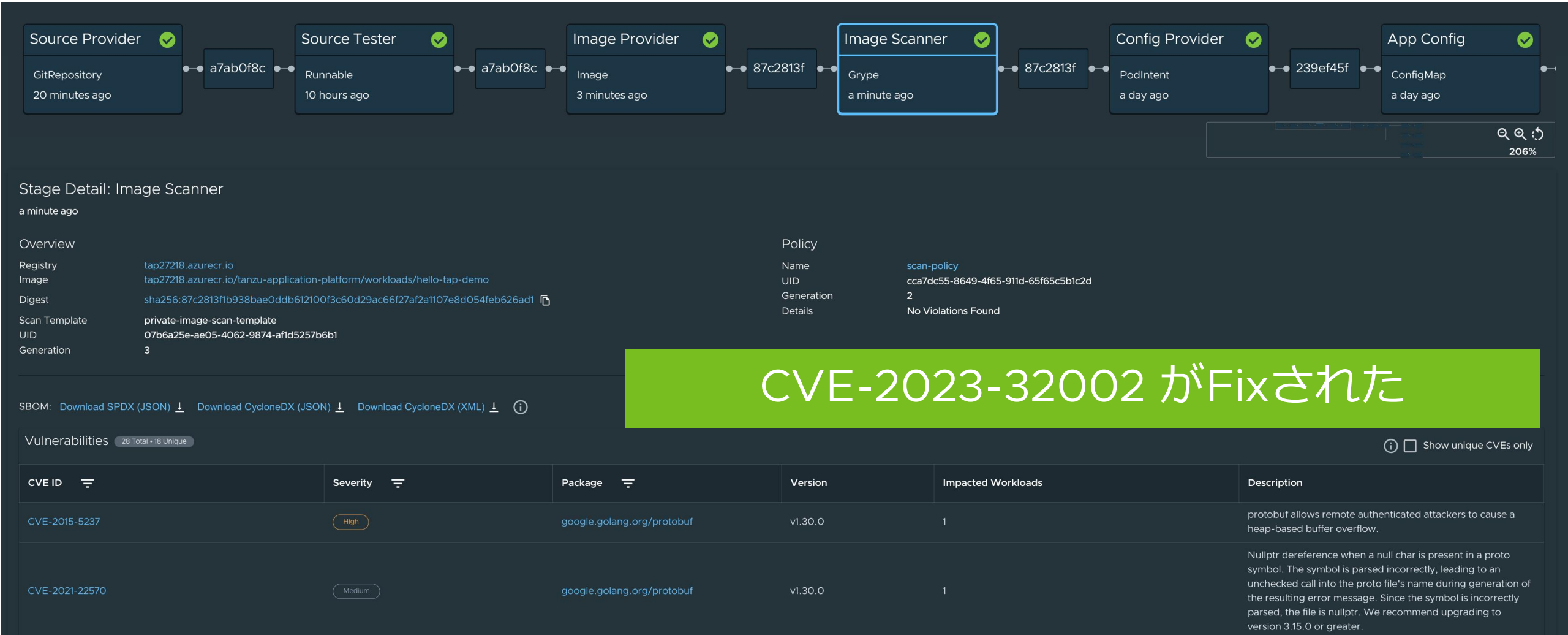
コンテナイメージの脆弱性スキャン

マニフェスト (YAML) を更新


Source Provider
GitRepository
a day ago
a7ab0f8c
Source Tester
Runnable
10 hours ago
a7ab0f8c
Image Provider
Image
a day ago
9ceb3bed
Image Scanner
Grype
a few seconds ago
Config Provider
PodIntent
a day ago
f257ea75
App Config
ConfigMap
a day ago
f82cc184
Service Bindings
ConfigMap
a day ago
f82cc184
Api Descriptors
ConfigMap
a day ago
166%
Stage Detail: Image Scanner
a few seconds ago
Failed because of 2 violations
Overview
Registry tap27218.azurecr.io
Image tap27218.azurecr.io/tanzu-application-platform/workloads/hello-tap-demo
Digest sha256:9ceb3beddcf8f134082bedb5af6ab83f1fc8f2440accea616a9700f0ac1f3784
Scan Template private-image-scan-template
UID 07b6a25e-ae05-4062-9874-af1d5257b6b1
Generation 1
Policy
Name scan-policy
UID cca7dc55-8649-4f65-911d-65f65c5b1c2d
Generation 2
Details Failed because of 2 violations: CVE Node Engine CVE-2023-32002 {"Critical"}. CVE node CVE-2023-32002 {"Critical"}
コンテナイメージに含まれる脆弱性一覧
SBOM: Download SPDX (JSON) Download CycloneDX (JSON) Download CycloneDX (XML)
Vulnerabilities 37 Total • 22 Unique
Show unique CVEs only
CVE ID
Severity
Packages
Version
Impacted Workloads
Description
CVE-2023-32002
Critical
2
Multiple
1
The use of `Module._load()` can bypass the policy mechanism and require modules outside of the policy.json definition for a given module. This vulnerability affects all users using the experimental policy mechanism in all active release lines: 16.x, 18.x and, 20.x. Please note that at the time this CVE was issued, the policy is an experimental feature of Node.js.
CVE-2023-32006
High
2
Multiple
1
The use of `module.constructor.createRequire()` can bypass the policy mechanism and require modules outside of the policy.json definition for a given module. This vulnerability affects all users using the experimental policy mechanism in all active release lines: 16.x, 18.x, and, 20.x. Please note that at the time this CVE was issued, the policy is an experimental feature of Node.js.
A privilege escalation vulnerability exists in the experimental

Details	Failed because of 2 violations: CVE Node Engine CVE-2023-32002 {"Critical"}. CVE node CVE-2023-32002 {"Critical"}

コンテナイメージに
CVE-2023-32002 Node.js の脆弱性 (Critical)
が含まれるため、Supply Chain の後続の処理をストップ



CVE-2023-32002 がFixされた

プラットフォームチームが整備した最適な道
(“Golden Path”)
の上で、アプリ開発者にデプロイしてもらえる

“TAP 良さそう！
でも Platform を構築するスキル、チームがない。
組織を巻き込めない…”

Tanzu Labs Platform Service

VMware のコンサルタントが、
お客様と一緒にアプリ開発者に使ってもらえる Platform と
それを提供し続けられるプラットフォームチームづくりを
お手伝いするサービスです。

一緒に最高の Platform を用意しましょう！

まとめ

- アプリ開発者にとって、Tanzu Application Platform を使うことで、素早く、継続的に、顧客へ価値を届け続けられる！
- プラットフォームエンジニアにとって、整備した最適な道（“Golden Path”）の上で、アプリ開発者にデプロイしてもらえる

本セッション受講の方へのお勧め

MC31103	11/14 17:00～	開発者としてアプリをデプロイしたい、継続的に ～ Tanzu Application Platformでデプロイを加速させよう ～
MC31106	11/15 17:00～	Tanzu Application Platform： ソフトウェア開発から展開までのプロセスをセキュアにする方法
MC32108	オン デマンド	開発者体験と開発効率向上の秘訣 - Tanzu Developer Portal
MC32107	オン デマンド	モダンアプリケーションの世界にようこそ -クラウド ネイティブ の基礎知識と Tanzu ポートフォリオについて
MC31105	11/14 15:00～	成功への加速: VMware Tanzu Labs が導くアプリケーションモダン化の道

vmware EXPLORE

ご清聴
ありがとうございました