JANOG48 Meeting

NaaS に関連する技術要素と "SP NaaS" を考える

v10_pub

Shunsuke Sasaki (shusasak@cisco.com)
Technical Solutions Architect, Cisco Systems
Jul 15, 2021

自己紹介

- 名前: 佐々木俊輔
- 趣味: 最近乗っていないロードバイク、旅行
- Twitter: @radiantmarch
- Company Blog:
  - https://gblogs.cisco.com/jp/author/shusasak/
- JANOGer 歴: JANOG36 (2015) あたりから参加
- 業務経歴
  - 2008- ネットワーク製品メーカー プリセールス SE (顧客領域: DC, SP WAN, Biz VPN, NFV, ...)
  - 2017- 通信事業者様向けの管理・自動化ソフトウェア製品の提案を行うアーキテクト
  - 2019- 次世代のネットワークや自動化・オーケストレーションのアーキテクチャの検討や提案も
    - ホワイトペーパー「シスコが考える 5G 時代のエンドツーエンドアーキテクチャ」第4章・6章執筆 [1] [2]
    - 電子情報通信学会ネットワーク 第三十回 仮想化特別研究専門委員会 招待講演[3]

[1] https://gblogs.cisco.com/jp/2020/11/published-two-whige-papers-about-5g/
[2] https://gblogs.cisco.com/jp/2020/01/service-provider-end-to-end-architecture-6-end-to-end-automation-in-5g-era-1/
[3] https://www.ieice.org/cs/nv/conference/

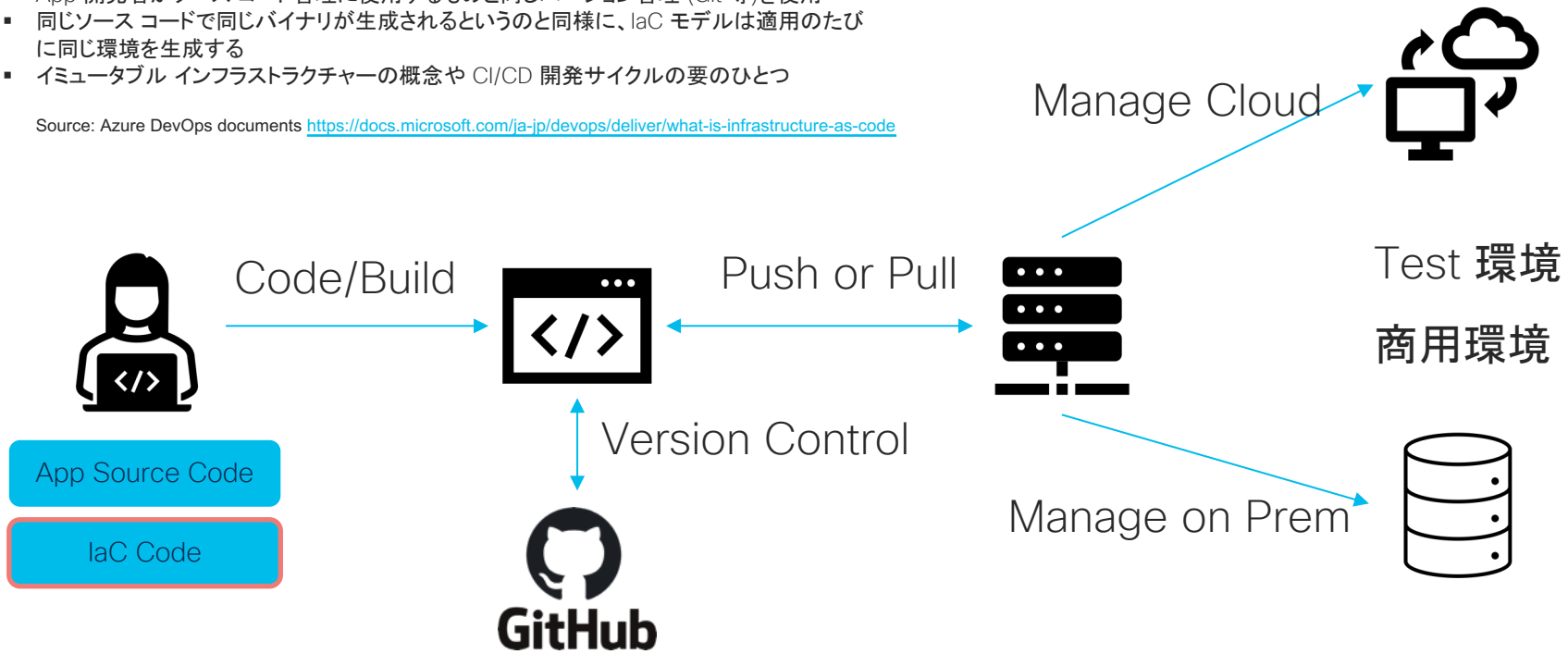
本日お話ししたいこと (12分)

- NaaS に関連する技術要素としての Infra as Code (IaC)
- SP が NaaS を提供するとしたら？ 必要な技術要素の検討
  - 抽象化、サービスモデル、階層化、可視化 (Observability)
- ディスカッション: "SP NaaS" サービスの実現性、課題、将来

Infrastructure as Code (IaC) とは

- 記述型モデルでのインフラストラクチャ (アプリ、コンテナ、仮想マシン、ロードバランサー、ネットワーク、接続トポロジ、etc..) の管理
- App 開発者がソース コード管理に使用するものと同じバージョン管理 (Git 等)を使用
- 同じソース コードで同じバイナリが生成されるというのと同様に、IaC モデルは適用のたびに同じ環境を生成する
- イミュータブル インフラストラクチャーの概念や CI/CD 開発サイクルの要のひとつ

Source: Azure DevOps documents https://docs.microsoft.com/ja-jp/devops/deliver/what-is-infrastructure-as-code

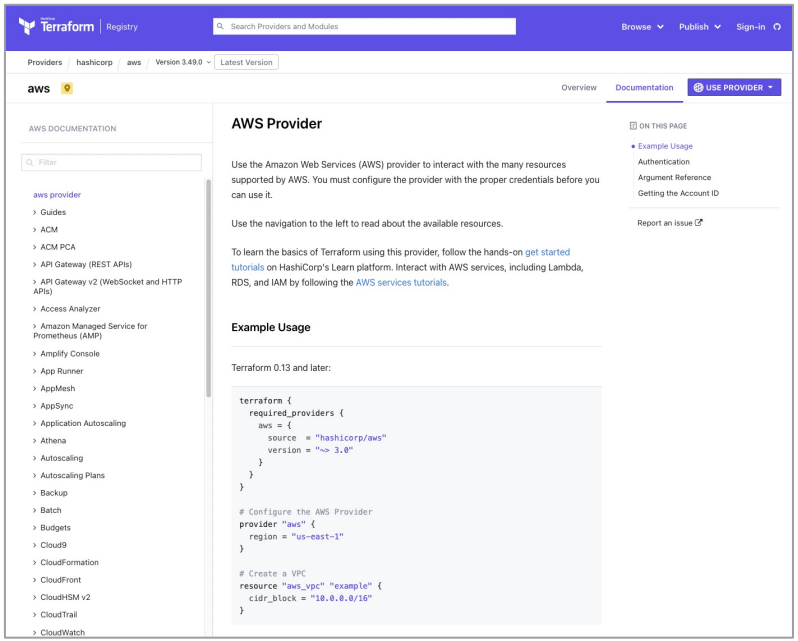


IaC ツールの一例: Terraform

- オープンソースのクラウド構成自動化ツール (HashiCorp 社による商用サポート)
- Terraform "Provider": リソースの API レイヤを抽象化するプラグイン
  - Provider を追加することで新しいインフラリソースをプロビ・管理することができる
  - 各種 CSP を始め、仮想化インフラベンダー、ネットワークベンダーなども自社製品に対する Provider を提供している

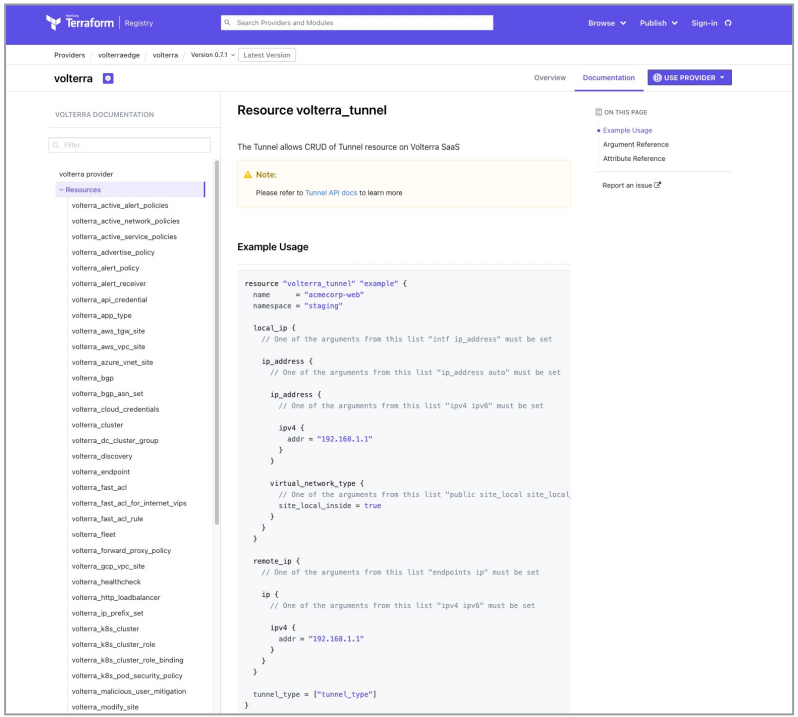
Terraform Provider Examples:

- AWS



Source: https://registry.terraform.io/providers/hashicorp/aws/latest/docs

- Volterra



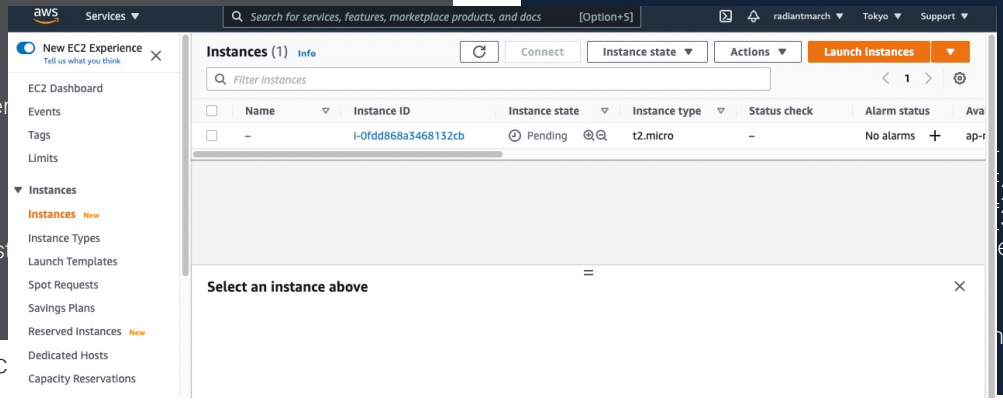
Terraform コードの例 (AWS)

main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}

resource "aws_instance" "app_ser
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInst
  }
}
```

実行

```
$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                          = "ami-830c94e3"
      + arn                          = (known after apply)
##...
```

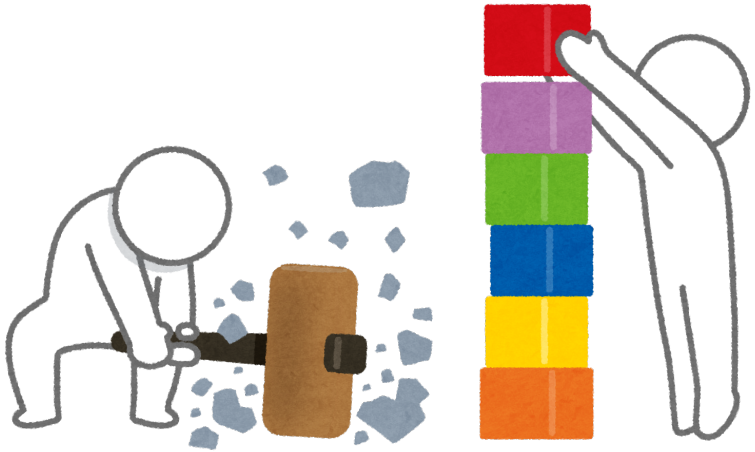



```
[10s elapsed]
[20s elapsed]
[30s elapsed]
ete after 36s [id=i-


nanged, 0 destroyed.
```

Source: https://learn.hashicorp.c

ここから個人的な想像

- SP の様々なネットワークサービスを IaC コードで記述できるようになったら企業ユーザ(アプリ開発者) は嬉しい？



想像: "IaC 対応 SP" ができた場合
Terraform コードのイメージ

main.tf

```
provider "SP-naas" {
  profile = "default"
  region  = "tokyo"
}

resource "l3vpn" "vpn1" {
  vpn_id              =  "automatic"
}

resource "l3vpn" "endpoint1" {
  parent              =  ${l3vpn.vpn1.id}
  pop_location        =  "tokyo_pop_1"
  endpoint_type       =  "physical_port"
  interface_id        =  "allocated"
  interface_type      =  "10G-SR"
  contract_bandwidth = "1G"
  burst_support = "true"
}

resource "traffic-engineering" "te1" {
  headend             = ${l3vpn.endpoint1.id}
  tailend             = ${l3vpn.endpoint2.id}
  request_type        = "low_latency"
  traffic_selection = {
                        "destination": [ "10.1.1.0/24", "10.2.1.0/24" ],
 }
}
```

provider を提供

L3VPN を構成

VPN エンドポイントを設定

Traffic Engineering の設定

実行

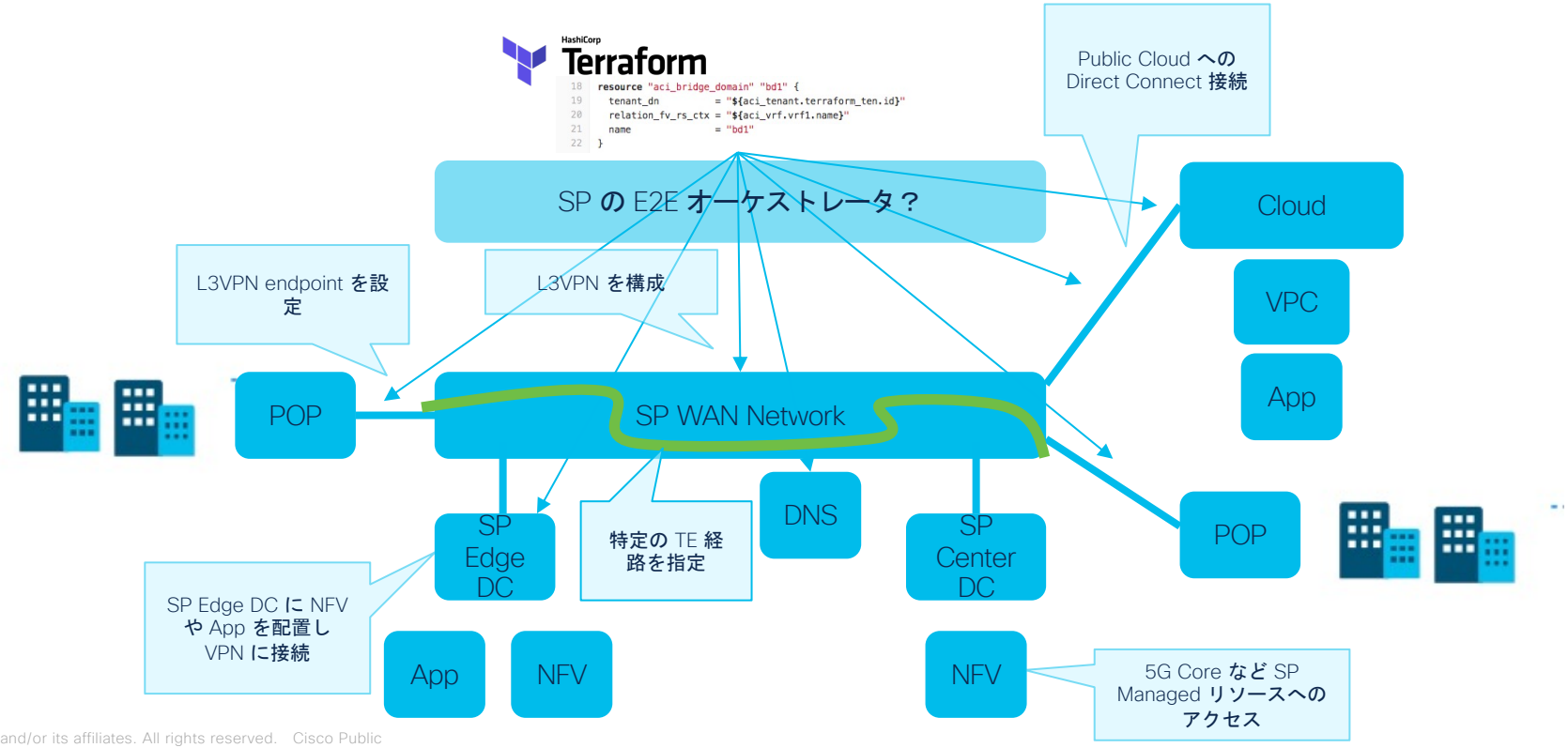
```
resource "edge_vpc" "vpc1" {
 location = "edge_dc1"
...
}

resource "nfv" "firewall" {
  location              = "edge_dc1"
  size                  = "small"
  attached_endpoint     =  ${l3vpn.endpoint2.id}
...
}

resource "firewall_rules" "fw_rule1" {
..
}

resource "nfv" "upf" {
   location             = "edge_dc1"
...
}

resource "servers" "vm1" {
  location              = "edge_dc1"
...
}

resource "dns" "dns1" {
```

SP エッジ DC を構成に組み込む

SP NFV サービスのデプロイ

Private 5G 向けのリソースへもアクセス

イメージ: "IaC 対応 SP NaaS サービス"



"SP NaaS" 実現のための E2E オーケストレーション技術検討要素

クロスドメイン プロビジョニング

SP 視点

See Appendix

- テクノロジ領域が RAN, DC, NFV, WAN, モバイルコアなどにまたがり一つのシステムからすべてを管理することは難しい
- 階層化コントローラによるエンドツーエンド オーケストレーションの実現
- 3GPP/5G E2E スライシングの議論と共通
- コントローラのインターフェイス標準化への期待 (3GPP/IETF)

サービスの 抽象化

企業のアプリ開発者視点

See Appendix

- IaC コードで構成を宣言的に記述する
- ネットワーク機器の設定項目を個別にAPI として公開するだけではアプリ開発者の使い勝手は悪い
- ユーザが記述しやすいようにリソースを抽象化し、内部のロジックを隠蔽する仕組みが必要
- サービスモデル / YANG ベースのサービス開発（例）

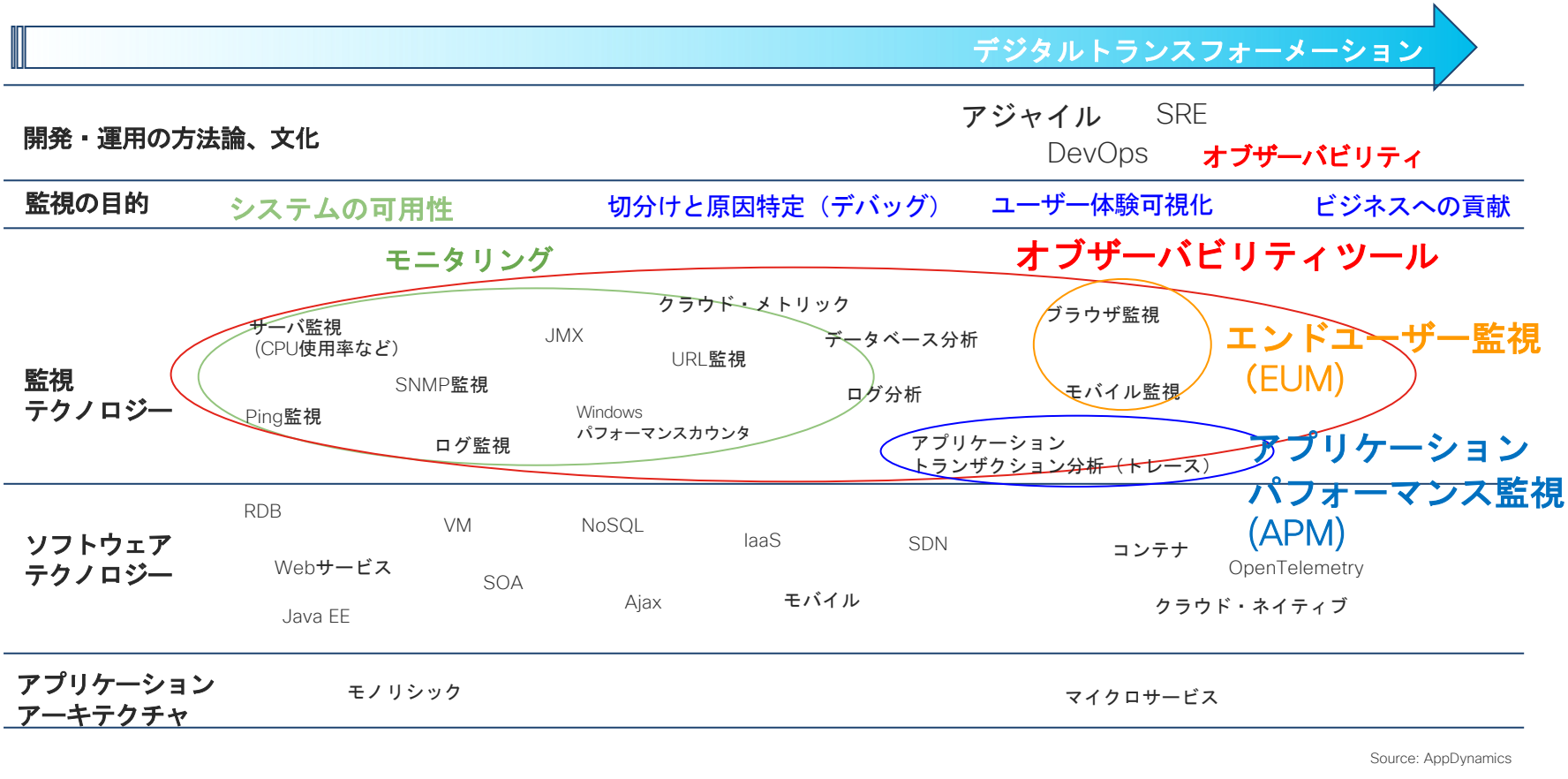
Today

Observability (可観測性)

企業のアプリ運用者視点

- アプリのユーザ体験に影響がある性能・品質問題を検知、分析できることがニーズ
- 単なるネットワーク監視ではなく、ネットワークの性能情報をアプリケーションの性能情報と紐付けられるような仕組みが必要

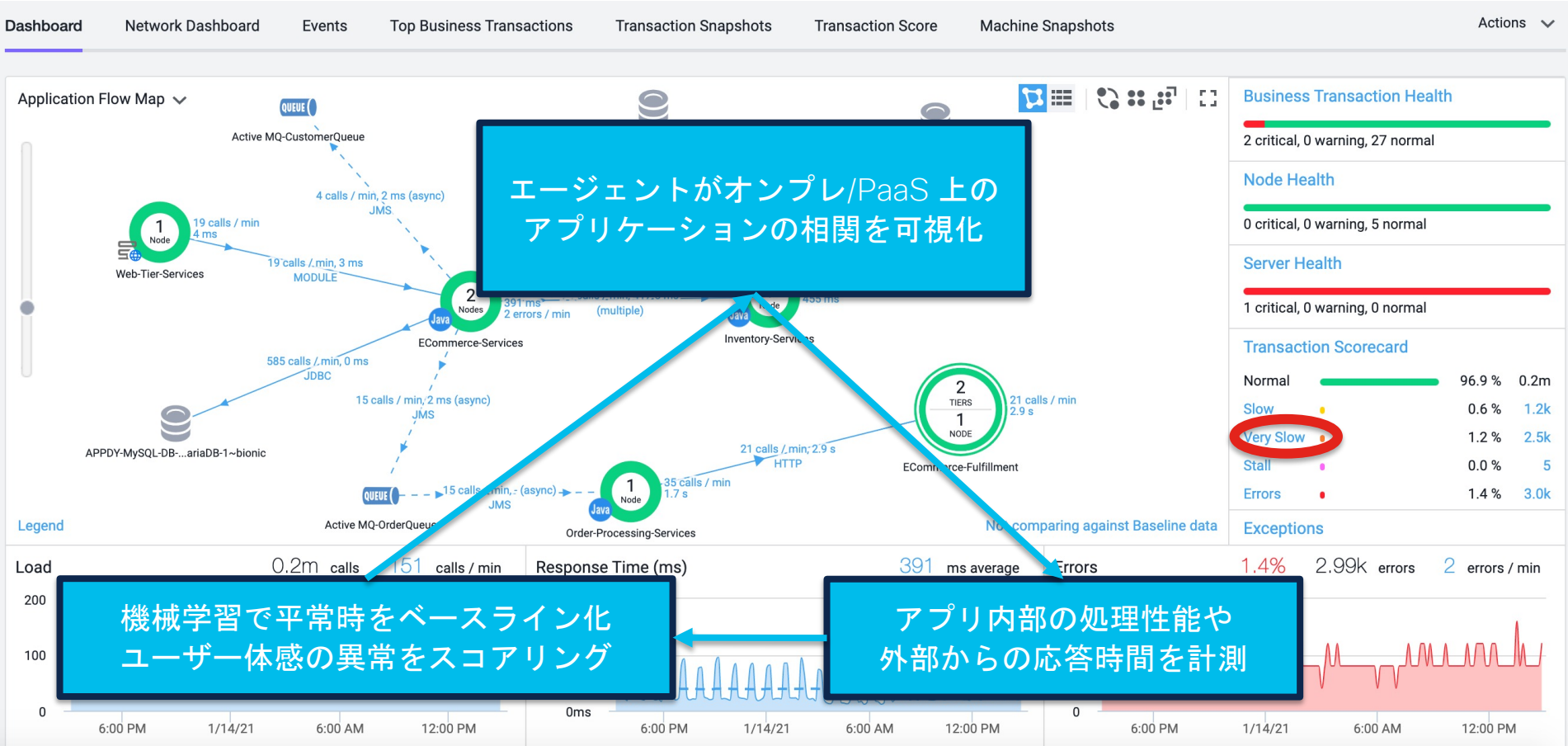
トレンド: モニタリングからオブザーバビリティへ



Source: AppDynamics

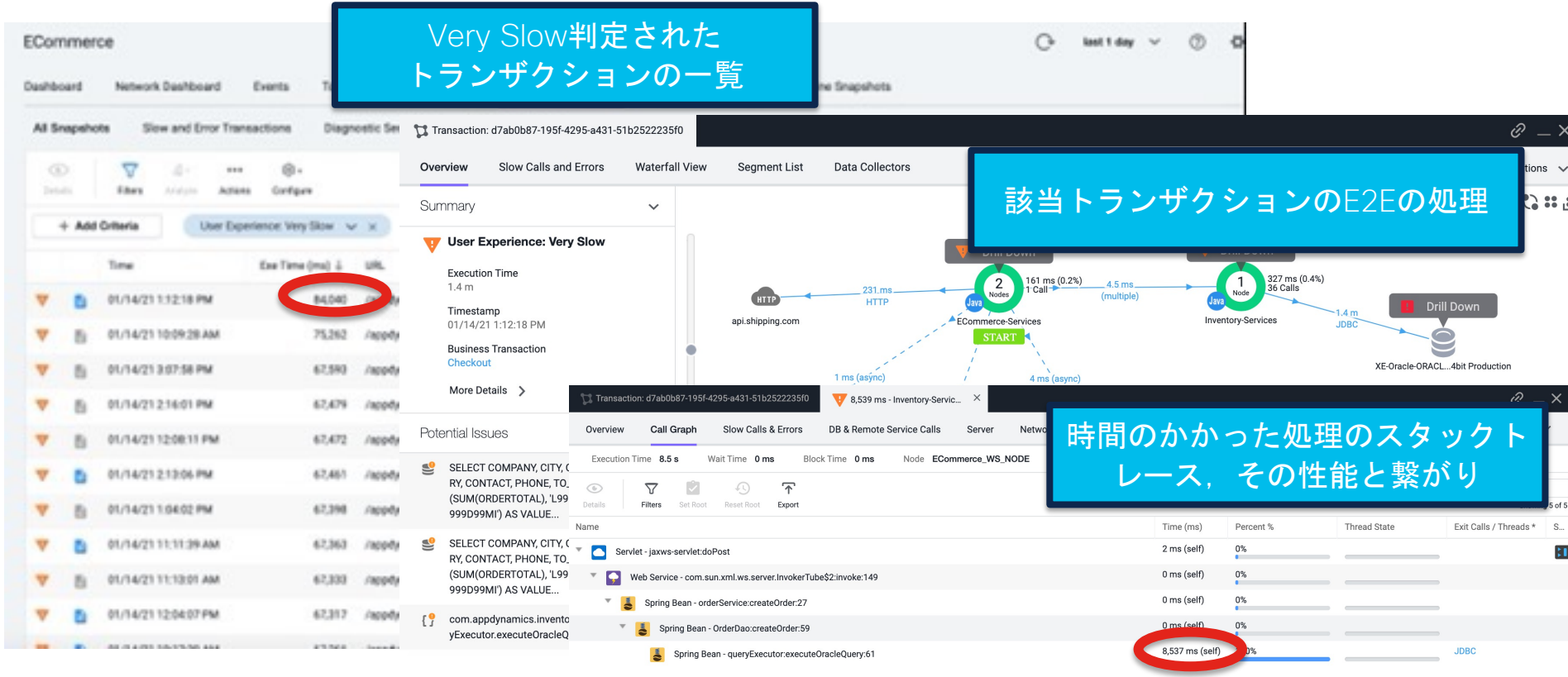
Application Performance Monitoring (APM) の例

source: AppDynamics

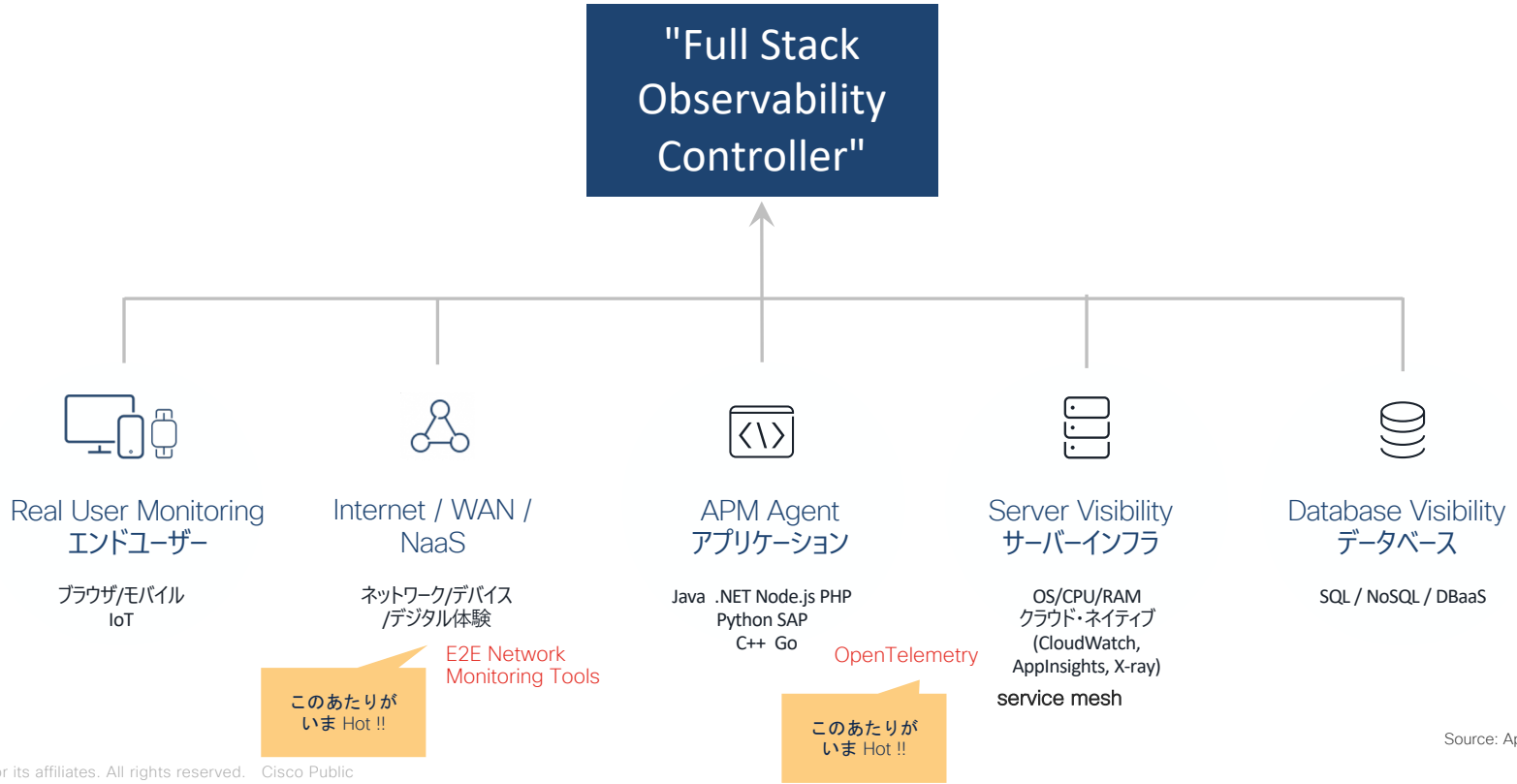


Application Performance Monitoring (APM) の例

source: AppDynamics



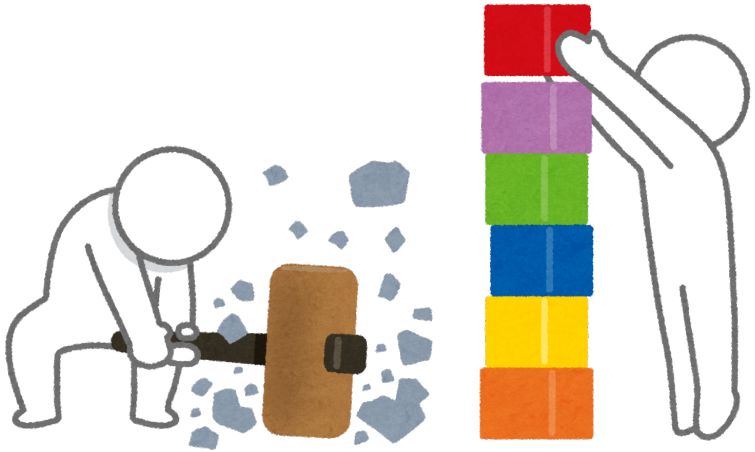
今後は "Full Stack" Observability が求められている



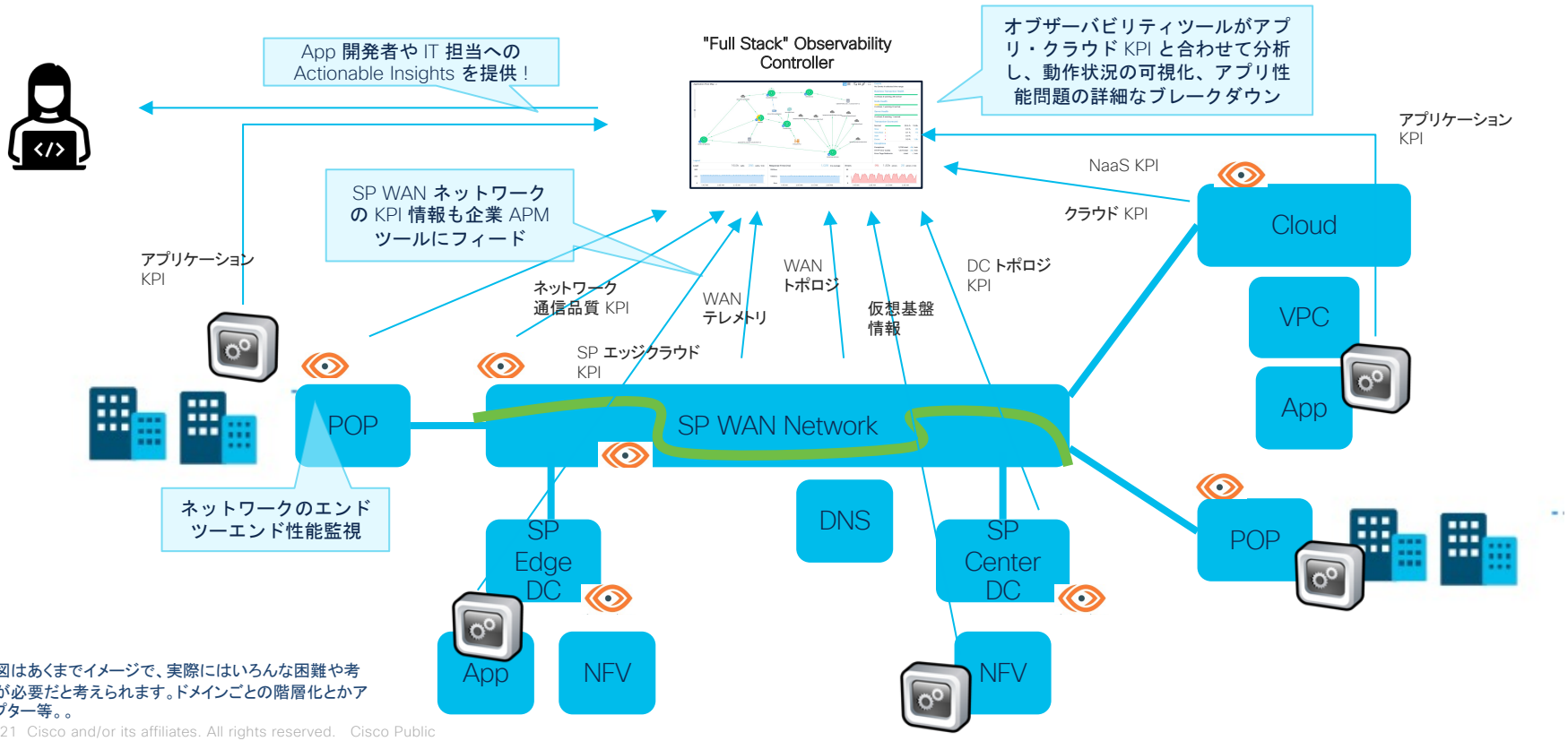
Source: AppDynamics

ふたたび個人的な想像

- SP が自身のネットワークの性能情報（テレメトリ）をオブザーバビリティツールにフィードできるようになったら、企業ユーザ(アプリ運用者) は嬉しい？



イメージ: "オブザーバビリティ対応 SP NaaS"

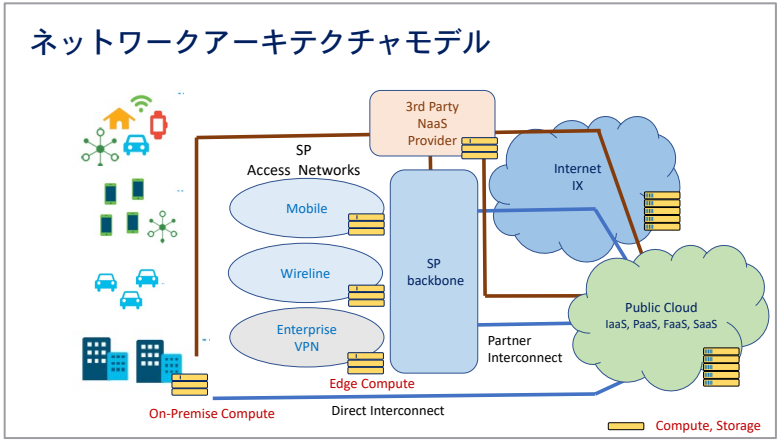


※図はあくまでイメージで、実際にはいろんな困難や考慮が必要だと考えられます。ドメインごとの階層化とかアダプター等。。

ここからは
ディスカッション

JANOGer の皆様への問いかけ

- SP の皆様: "SP NaaS" は提供できる？ 提供したい？ すでに実現されてていたりする？ そもそもニーズはある？
- NaaS ベンダーの皆様: NaaS 製品と SP がうまく組んでいる実例がありますか？ どのようにパートナーシップを組むことが考えられますか？
- SaaS ベンダーや企業ユーザ、アプリ開発者の皆様: NaaS の利用・検討状況は？ SP NaaS への期待は？ Edge SP クラウドや、Private 5G/Local 5G などと絡めたユースケースなど
- Cloud SP の皆様: NaaS をどう見ていますか？
- IT 人材の視点: 高度なアプリケーション人材を SP/ネットワーク業界に呼び込み続けるには?



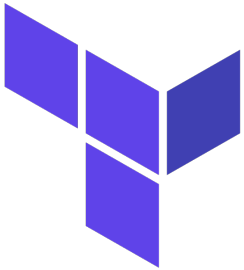
河野さん JANOG 48 発表資料

The bridge to possible

Appendix

Ansible vs Terraform

- Very much focused towards infrastructure automation
- Modules (written in Python)
- Define playbooks in YAML
- Stateless *(by default)* – push out the intent of a playbook on run
- Imperative

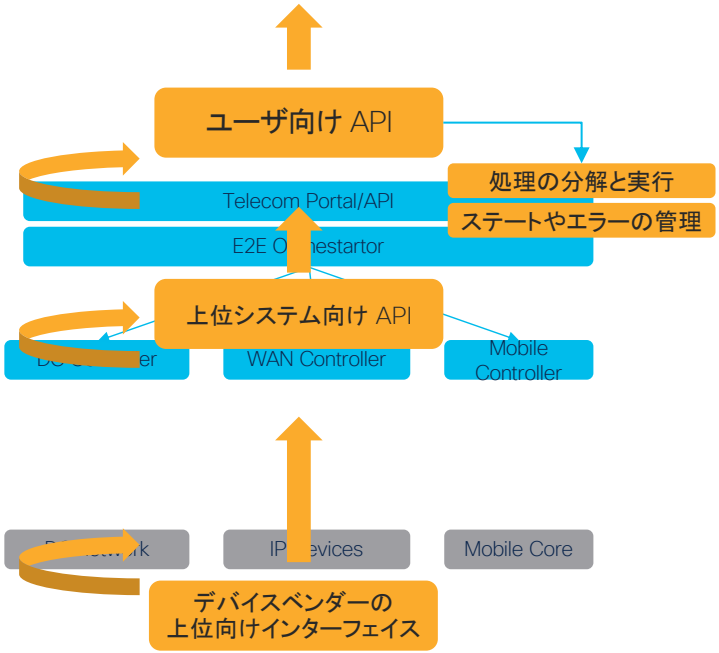


- More focused to cloud automation
- Lends itself better to API first platforms
- Providers (written in Go)
- Define a Terraform config in HCL (HashiCorp configuration language)
- Stateful – keeps a state and looks to ensure the config matches the state
- Declarative

"SP NaaS" オーケストレーション の検討事項

クロスドメイン プロビジョニング

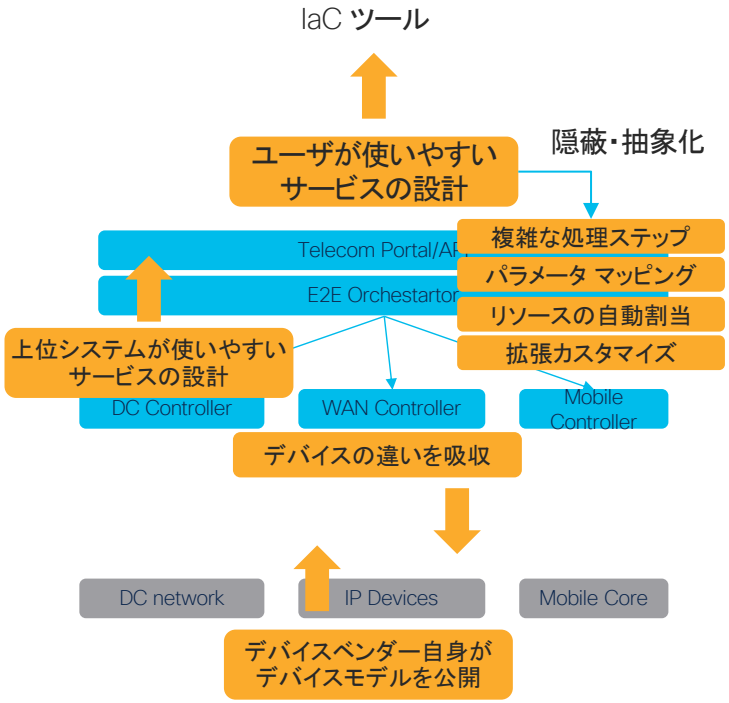
- ドメインコントローラがそのドメインを代表してプロビジョニング リクエストを受け付ける
- ドメイン内部でのクローズド ループ
- 階層的なシステムとなるがコントローラ間のインテグレーションを容易にする仕組みが必要
- 各コントローラで必要なサービスモデルをゼロから開発するのは手間であり、業界標準やコミュニティの成果をマッシュアップ（再利用・組み合わせ）できることが望ましい



"SP NaaS" オーケストレーション の検討事項

サービスの抽象化

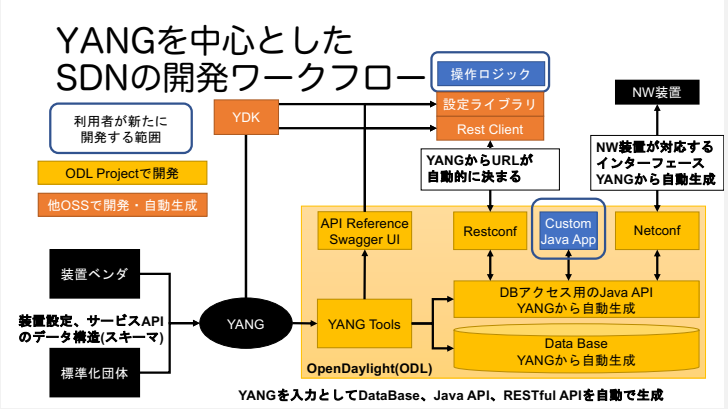
- IaC コードで 「どう設定したいか (what) 」 をシンプルに記述できるようなサービス設計
- ネットワーク機器の設定項目をそのまま API として公開するのではなく抽象化
- ユーザ入力を具体的なデバイス設定にマッピングするためのロジックの開発
- サービス差別化のためにカスタマイズ開発が可能なプラットフォーム
- サービス設計 = ユーザ向けインターフェイス
  設計はセンスが問われサービスの差別化要素となる (Public CSP がそうであるように)



クロスドメイン プロビジョニング　サービスの抽象化

選択肢の一つ: YANG ベースのサービス開発

- YANG (RFC7950): **モデリング言語**
- YANG **モデルは** Github**で公開されている**
  - **ベンダーデバイス** OS **の** YANG:
    - https://github.com/YangModels/yang/tree/master/vendor
  - **ベンダー非依存モデルの例**: OpenConfig
    - https://github.com/openconfig/public
  - Network Slicing YANG
    - IETF **各種ドラフトで定義が進む**
- YANG **モデルから制御用のインターフェイスを自動生成することができる**
- YANG **モデルはモデル間の参照、マウント** (YANG Mount, RFCx)**、追加** (Augment) **といった柔軟な拡張が可能**
- YANG Push **による装置間・コンポーネント間の疎結合かつ有機的な連携**



堀場さん, MPLS Japan 2018
https://mpls.jp/2018/index.html

YANG ベースの実装例として OpenDaylight が取り上げられて紹介されている。他の実装例として商用サポート製品としては Cisco NSO などがある

The bridge to possible