

OSC 2026 秋

プライベートクラウド Marmotのご紹介



私が、開発を進めているパソコン 1 台から始められるオープンソースのプライベート・クラウド・プラットフォームについて、デモを含めて、何ができるか、何を
目指しているかを紹介させていただきます。

2026年10月3日

高良 真穂

お話しする人

- 経歴

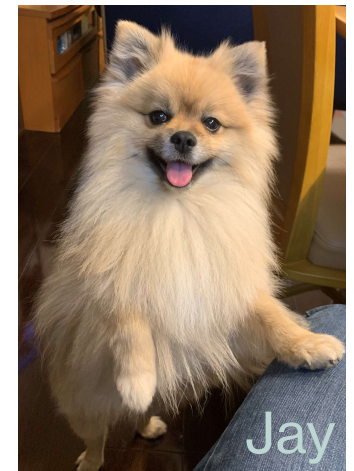
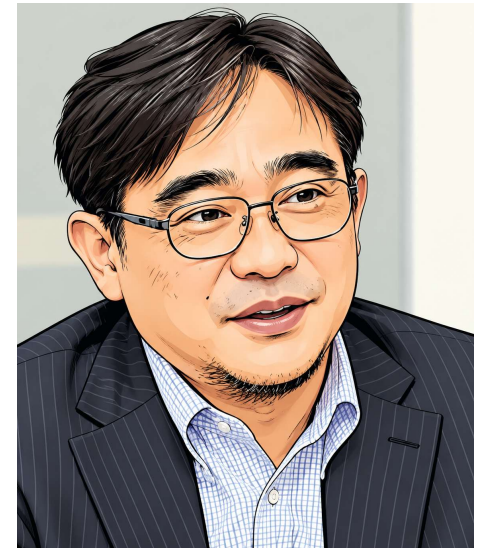
- もとIBMクラウド中の人
- もとKintoneサービス中の人

- 所属 株式会社ジール

本プレゼンは所属する会社を代表するものではなく個人の取り組みです

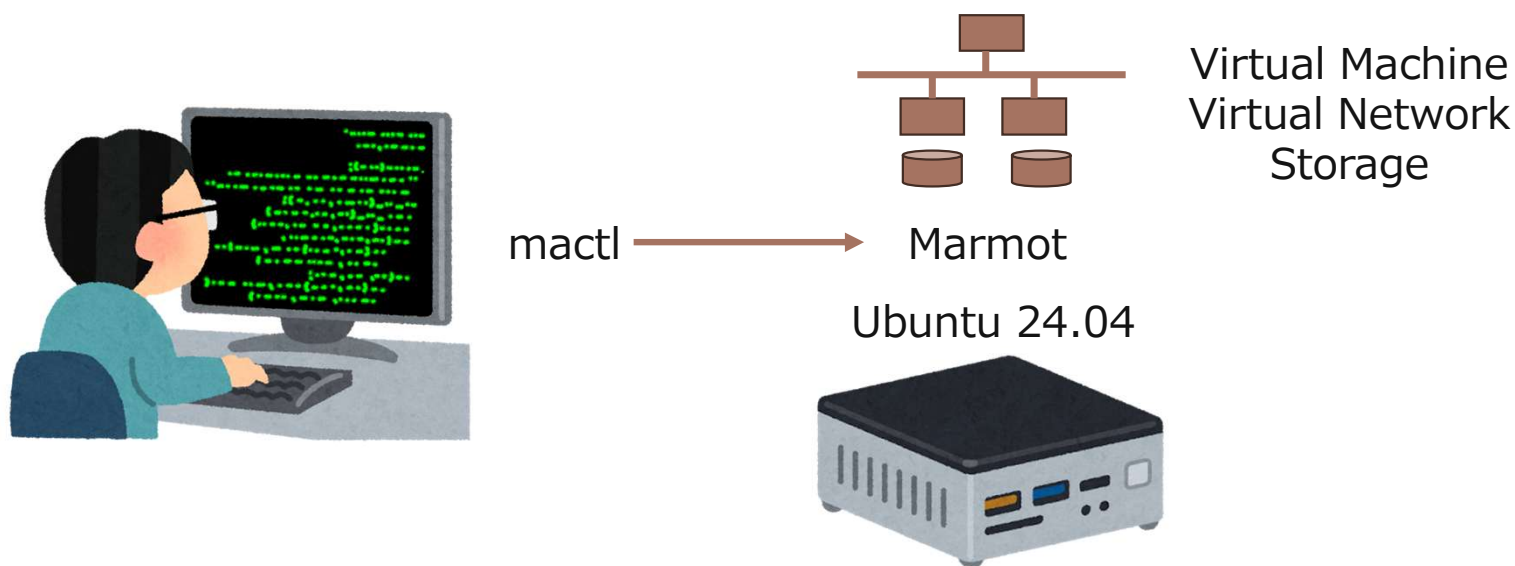
- 愛犬と散歩が楽しい人

- 知見を活かしてMarmotを開発



Marmotとは何か

Marmotは、小規模な環境でも簡単にプライベートクラウドを構築できるOSSです。



なぜMarmotを作ったのか？

- クラウドは便利だけど、すべてをクラウドに置く必要があるだろうか？
- もっと小さく、もっと安価に、もっと自由にコントロールできるクラウド環境があってもいい。



- 開発環境／Ansible開発環境
- 検証環境
- 学習環境
- ホームラボ
- CI/CDランナー
- エッジ環境
- スモールシステム
- 小規模サービス
- Kubernetesのアプリ開発環境

Marmotのコンセプト

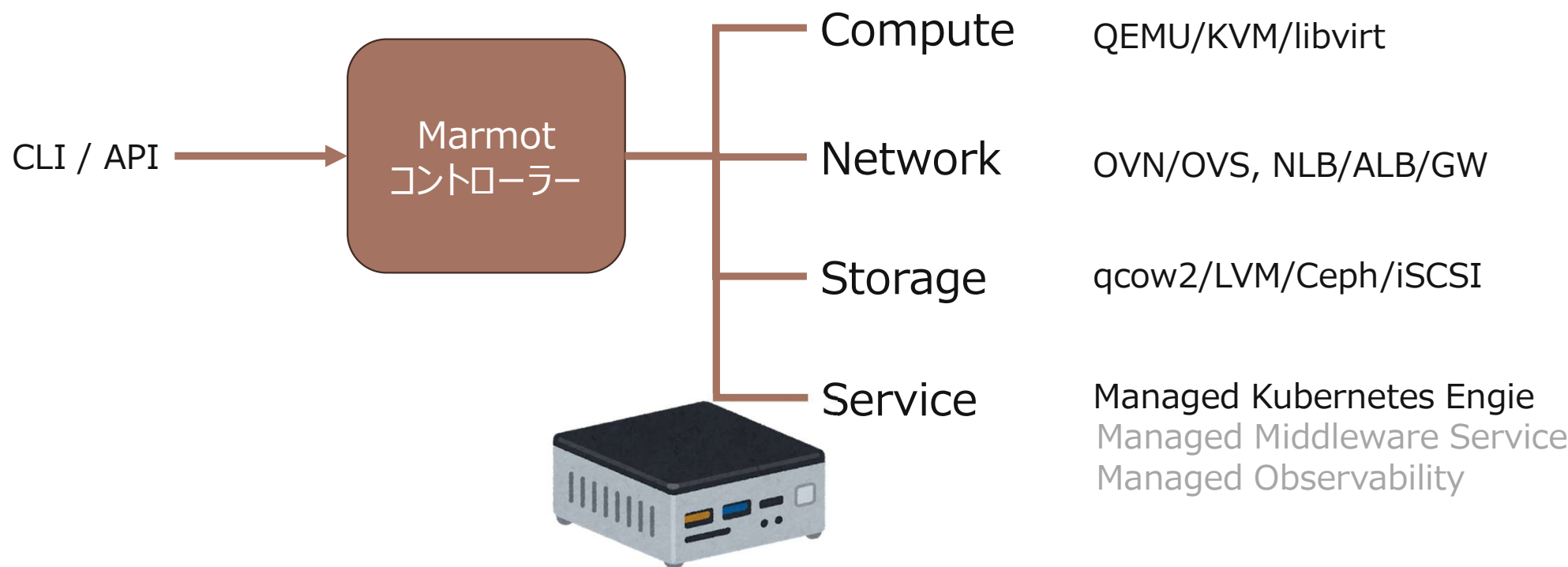
Simple 宣言型で簡単に作れる

Small 1 台のPCから始められる

Open 自分で使い、自分で改良できる

Marmotのアーキテクチャ

- Kubernetesのコントローラーのリコンサイル ループ に倣って、クラウドのオブジェクトもコントロールしたら面白そう！





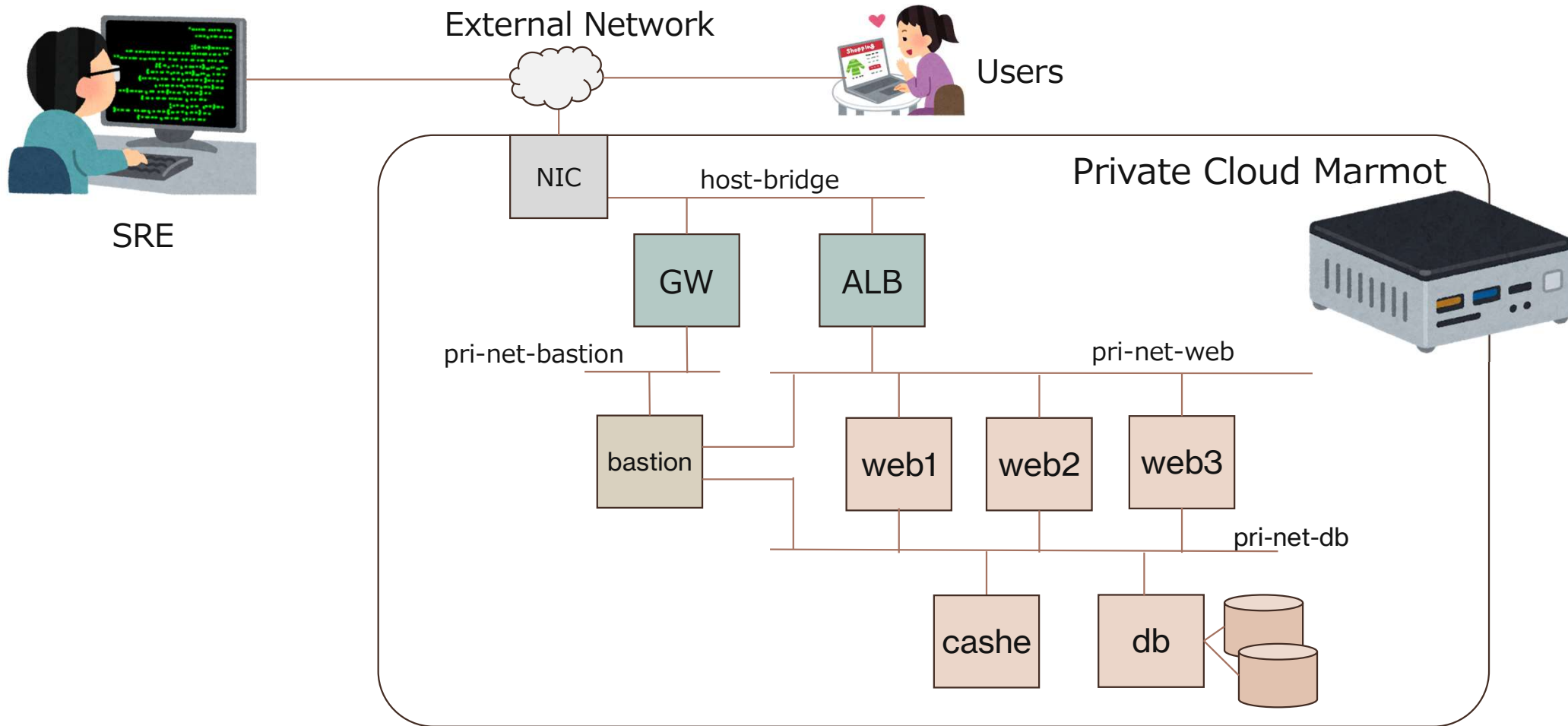
Marmotで何ができるのか？

デモのおしながき

- 1 アプリ開発・実行基盤構築
- 2 踏み台サーバー経由で仮想サーバーに入る方法
- 3 カスタムVMイメージの作成
- 4 Marmot Kubernetes Engine

1 アプリ開発・実行基盤構築

次の構成を3つのYAMLファイルを適用するだけで構築できます。



1 アプリ開発・実行基盤構築

- 次の3ステップで、前述のシステム構成が構築できます。

ネットワークのデプロイ

```
$ mactl create -f pri-networks.yaml
```

サーバーのデプロイ

```
$ mactl create -f servers.yaml
```

ロードバランサーとゲートウェイのデプロイ

```
$ mactl create -f gateways
```

※ 1本のYAMLにまとめる事もできますが、分かり易さと保守性を兼ねて3本にしました。

YAMLの例

ネットワークの YAML

```
1  apiVersion: v1
2  kind: VirtualNetwork
3  metadata:
4    name: pri-net-bastion
5    comment: 踏み台サーバー用のプライベートネットのネットワーク
6    labels:
7      case: 30
8      role: bastion
9  spec:
10   ipNetworkAddress: 172.16.90.0/26
11  ---
12  apiVersion: v1
13  kind: VirtualNetwork
14  metadata:
15    name: pri-net-web
16    comment: Webサーバー用のプライベートネットのネットワーク
17    labels:
18      case: 30
19      role: web
20  spec:
21   ipNetworkAddress: 10.120.1.0/24
22  ---
23  apiVersion: v1
24  kind: VirtualNetwork
25  metadata:
26    name: pri-net-db
27    comment: DBサーバー用のプライベートネットのネットワーク
28    labels:
29      case: 30
30      role: db
31  spec:
32   ipNetworkAddress: 10.120.10.0/24
```

サーバーのYAML (抜粋)

```
1  apiVersion: v1
2  kind: Server
3  metadata:
4    name: srv30-bastion
5    comment: Privateネットワークの踏み台サーバー
6    labels:
7      case: 30
8      role: bastion
9  spec:
10   cpu: 1
11   memory: 1024
12   mmImage: ubuntu24.04
13   auth:
14     publicKey: "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP6CPvNH
15   networkInterface:
16     - networkname: pri-net-bastion
17     - networkname: pri-net-web
18     - networkname: pri-net-db
19  ---
20  apiVersion: v1
21  kind: Server
22  metadata:
23    name: srv30-cache
24    comment: Privateネットワークのキャッシュサーバー
25    labels:
26      case: 30
27      role: cache
28  spec:
29   cpu: 1
30   memory: 1024
31   mmImage: ubuntu24.04
32   auth:
33     publicKey: "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP6CPvNH
34   networkInterface:
35     - networkname: pri-net-db
36  ---
```

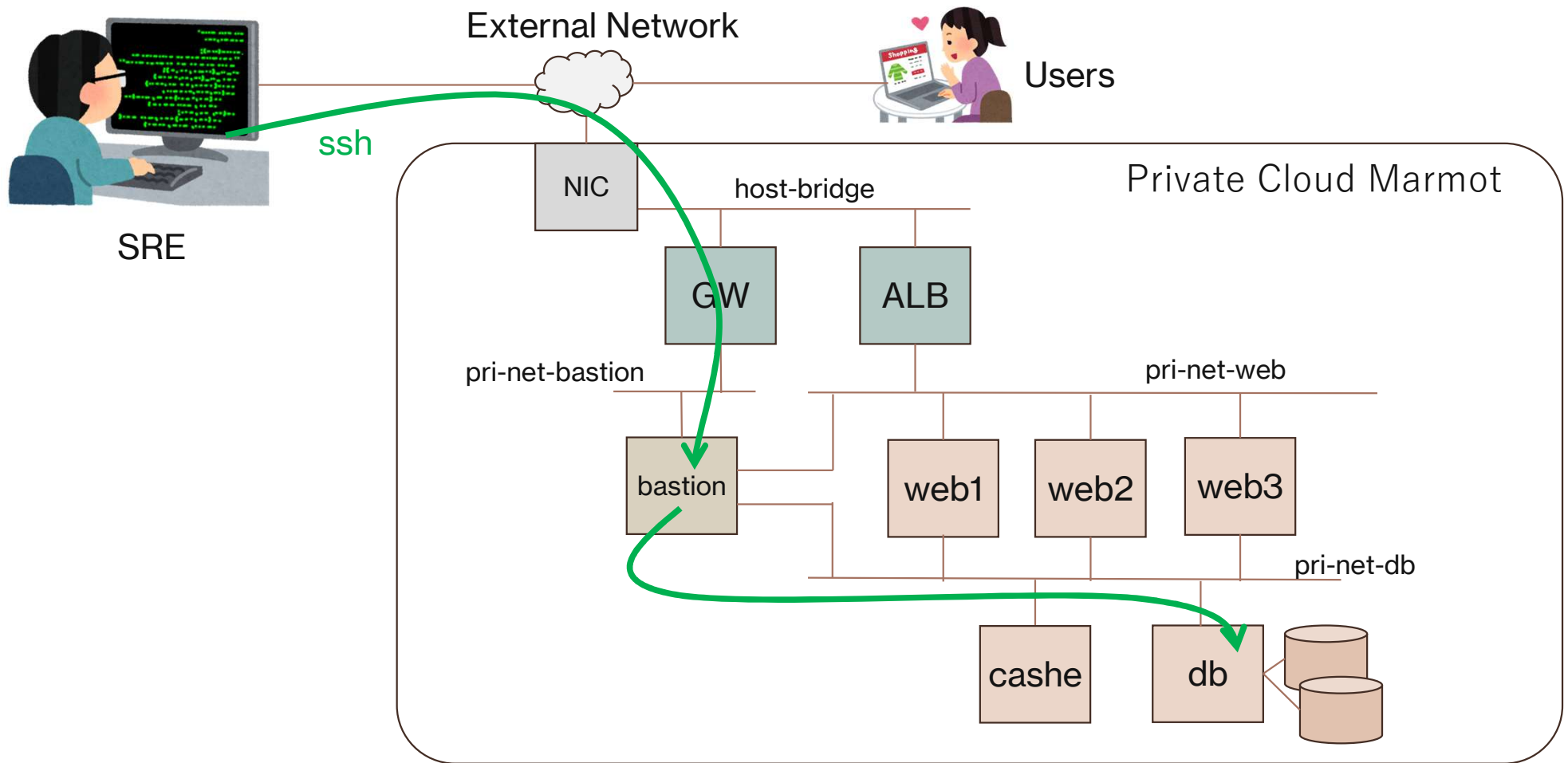
アプリ開発 & 実行環境の構築

動画はサイズの関係で削除しています。
ご覧になりたい時は、発表のビデオをご覧ください

1 アプリ開発・実行基盤構築（YAMLのリンク）

https://github.com/takara9/marmot-manifests/tree/main/30_server-on-private-net

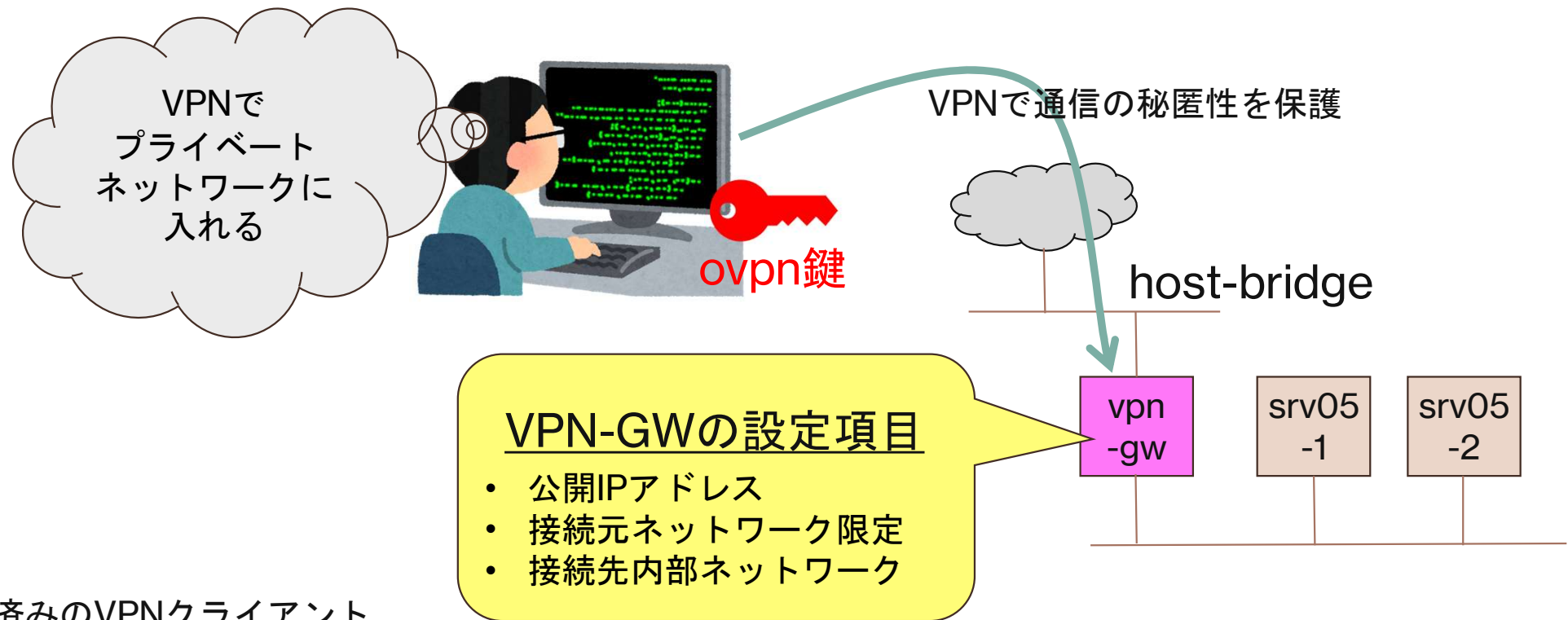
2 踏み台サーバー経由で仮想サーバーに入る方法



踏み台から プライベート内の VMへ入る方法

動画はサイズの関係で削除しています。
ご覧になりたい時は、発表のビデオをご覧ください

他にOpen VPNクライアントで接続する方法もあります



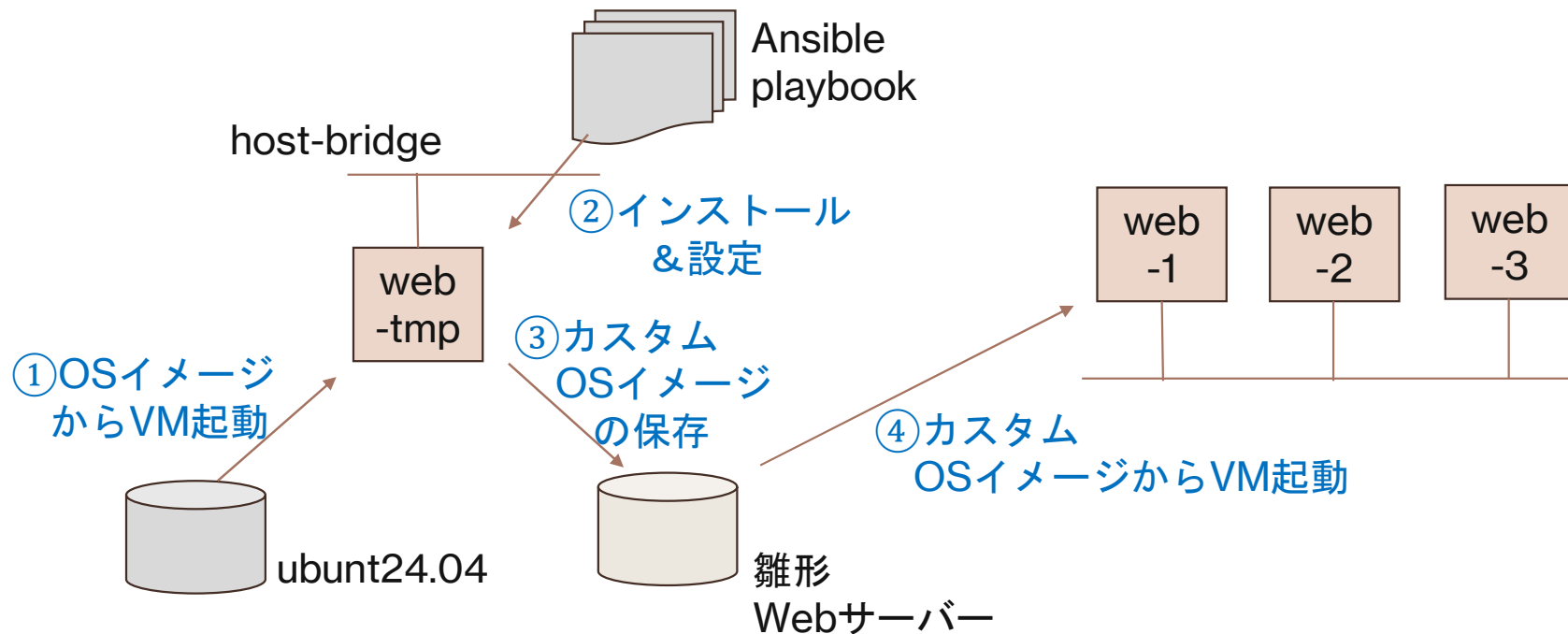
検証済みのVPNクライアント

MacOSの場合 TunnelBlick

Windowsの場合 OpenVPN Connect

3 カスタム・VMイメージの作り方

- ・ カスタマイズしたOSをテンプレートとして保存して、利用することができます。



カスタムVM イメージの作り方

動画はサイズの関係で削除しています。
ご覧になりたい時は、発表のビデオをご覧ください

MarmotとAnsibleの連携

Marmot の YAMLに AnsibleのPlaybookを記述すると
仮想マシンの起動後に、セットアップまで実行

```
1  apiVersion: v1
2  kind: Server
3  metadata:
4    name: web-tmp
5    comment: プライベートネットワークに接続するサーバー
6    labels:
7      app: web1
8  spec:
9    cpu: 1
10   memory: 1024
11   mmImage: ubuntu24.04
12   auth:
13     publicKey: "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAAT6CPvNHHeF8rHScC7vrq7HMqTPQJcl08atMSqHK/gSy For marmot VMS"
14     users:
15       - ubuntu
16   ansible:
17     playbook: playbook/install.yaml
18     inventory: playbook/host-tmp.yaml
19   networkInterface:
20     - networkname: host-bridge
21     - networkname: app-net
```

GitHubのディレクトリ

```
mini:06_application-loadbalancer takara$ tree
.
├── alb.yaml
├── app-net.yaml
├── image
│   ├── pic06-1.png
│   └── pic06-2.png
├── playbook
│   ├── host-tmp.yaml
│   ├── install.yaml
│   └── roles
│       └── webapl
│           ├── files
│           │   ├── package.json
│           │   ├── public
│           │   │   └── index.html
│           │   ├── server.js
│           │   ├── test.sh
│           │   └── webapl.service
│           └── tasks
│               └── main.yaml
├── README.md
├── web-tmp.yaml
└── webs.yaml

8 directories, 15 files
```

Marmot の YAML
と AnsibleのPlaybookを
GitHubで一緒に管理できる



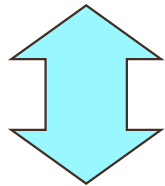
簡単にK8sクラスターを構築できる
Marmot Kubernetes Engine

MKE

Marmot Kubernetes Engineとは

- これまでのMinikubeやKINDでは

- VMやコンテナで起動
- ロードバランサーなし、PVCはローカルストレージ

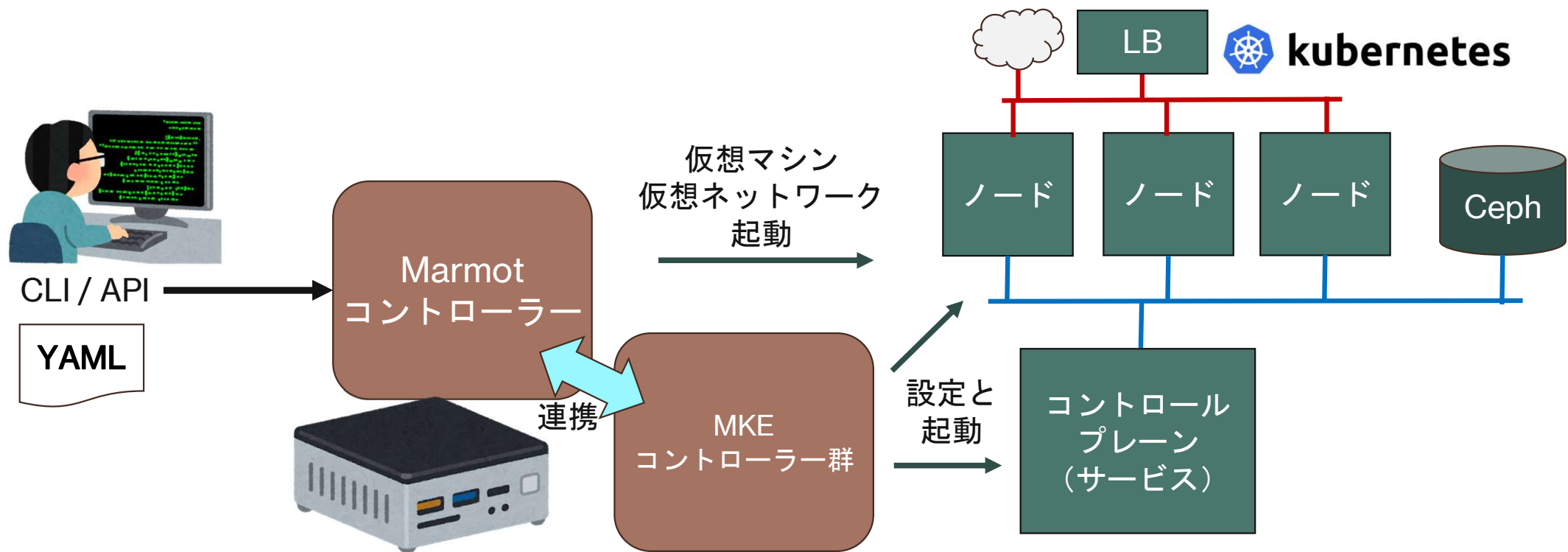


- MKEはクラウドのK8sサービスと同様

- MarmotがK8sクラスタを構築、バージョンアップ、ノード増減
- サービスとロードバランサーが連携
- 永続ストレージとしてCephを自動組み込み
- マルチノードのK8sクラスタが起動できる

MKEのアーキテクチャー

マーモットのコントローラーは、MKEのAPIが記述されたYAMLを受け取ると、MKEのコントローラー群と連動して、K8sコントロールプレーンを起動し、ノードのVMを起動することで、Kubernetesクラスタを構成します。



MIKEのYAMLの例

```
Code Blame 16 lines (16 loc) · 335 Bytes

1  apiVersion: v1
2  kind: KubernetesEngine
3  metadata:
4    name: mke2
5    comment: 検証用K8sクラスタ
6  spec:
7    version: "1.36"
8    nodes: 3
9    nodeSpec:
10      cpu: 2
11      memory: 4096
12      network:
13        external: host-bridge
14        cni-plugin: cilium
15      auth:
16        url: https://github.com/takara9.keys
```

K8sバージョン

ノード数

ノードのスペック

CNIプラグイン

Marmot Kubernetes Engineのデモ

動画はサイズの関係で削除しています。
ご覧になりたい時は、発表のビデオをご覧ください

YAMLファイルのリンク

- <https://github.com/takara9/marmot-manifests/tree/main/70-mke-simple>



Marmotによって何が変わるか？

Marmotは何が違うのか？

宣言型 or 命令型

Kubernetesライクな宣言型APIを採用, Proxmox/OpenStackは命令型

状態管理の所在

etcdがクラスター全体の状態を管理し、コントローラーがその状態に向けて調整

抽象化の範囲

ALB,NLB,MKEなど広範囲

操作体系

mactl は create/apply/get/delete という kubectl に近い体系

Marmotは、どんな状況に適しているか？

状況	適した選択肢	理由
小規模チームで VM を安定運用したい	Proxmox	成熟度・GUI・HA・バックアップが揃っている
多数の利用者に IaaS を提供する	OpenStack	マルチテナントと API 互換性が必須になるため
検証・学習環境を素早く作っては壊す	Marmot	宣言型で再現性が高く、LB や K8s まで一括で出せる
VMware からの移行先	Proxmox 大規模はOpenStack	移行ツールと事例の蓄積

どんな人に使って欲しいか

ターゲット	抱えている課題	Marmot が提供する価値	刺さる一言
① インフラを学ぶエンジニア (SES 若手、転職希望者)	パブリッククラウドは課金が怖い、 VirtualBox では本格的なネットワーク構成を学べない	3層構成・LB・踏み台・プライベートネットワークを自宅のミニPC 1台で再現	「課金を気にせず、本物のクラウド構成を何度でも壊せる」
② SES 企業の教育・研修担当	受講者ごとの環境準備と復旧に 手間がかかる、研修費用が高い	同じ YAML で全員に同一環境を配布、壊れても数分で再作成	「研修環境の準備を YAML 1枚で」
③ 大学・専門学校の教員	演習用のクラウド予算が取れない、 学生ごとの環境管理が大変	学内の余剰サーバーで IaaS 演習環境を構築、RBAC で権限分離	「教室にプライベートクラウドを」
④ アプリ開発者・SRE の検証	本番に近い多層構成をローカルで 試せない、検証のたびにクラウド費用が発生	本番類似構成を宣言的に再現し、 障害試験や構成変更の事前検証が可能	「本番と同じ形の検証環境を、手元で」
⑤ Kubernetes 学習者	kind や minikube では LB や ストレージ連携が本番と違う	MKE で VM ベースの K8s クラスタを LB・Ceph 連携込みで構築	「kind の次に進む、本物に近い K8s ラボ」
⑥ 小規模チーム・研究室	パブリッククラウド費用の増大、 VMware ライセンス高騰、 OpenStack は重すぎる	手持ちのサーバーで IaC 前提の 社内クラウドを運用	「OpenStack ほど重くない、 YAML で動くプライベートクラウド」

Marmotのインストール方法

- Marmotは、開発中のソフトウェアです。
- そのため、出来るだけ最新バージョンのリリースを利用してください。
- 理由は、不具合があると修正して、バージョンが上がるので、最新バージョンほど安定しています。
- インストールの様子を撮ったビデオをご覧ください。（約 4 分 3 4 秒）



Marmotの インストール

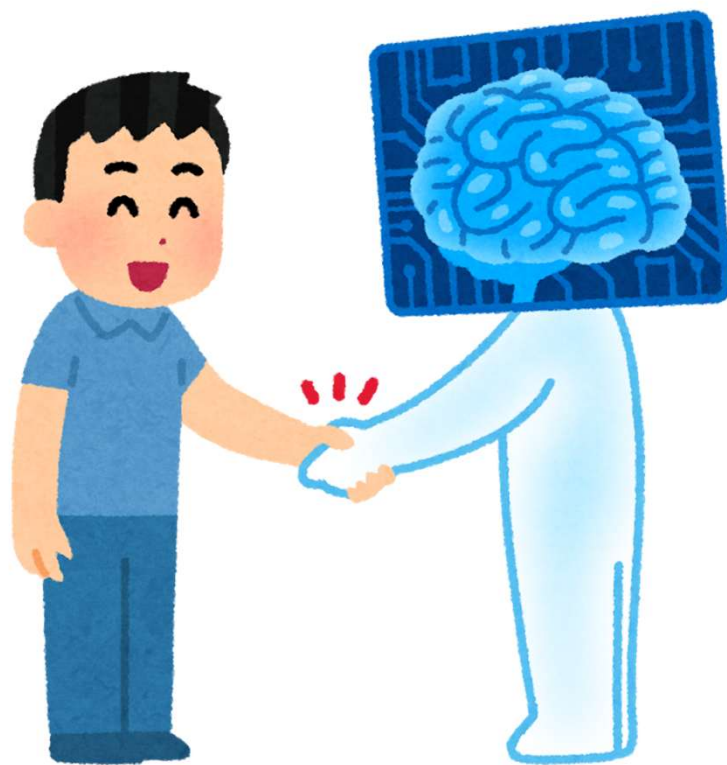
動画はサイズの関係で削除しています。
ご覧になりたい時は、発表のビデオをご覧ください

Marmotを作って分かったこと



生成AIは大切なパートナー

- ・構想策定
- ・設計
- ・実装依頼
- ・判断／承認



- ・設計支援
- ・コーディング
- ・テスト
- ・問題原因調査
と対策

CIのテストは 超重要

- 機能を増やす時、テストも追加
- AIが問題判別する時もテストを使用
- 機能を追加すると、不安定系が発生してしまうので、最適化の繰り返し
- テスト時間が長くなるので、ランナーを増やして並列化
- CIのランナーもMarmotで動かして安定稼働をテスト

The screenshot displays the GitHub Actions interface for a repository named 'marmot'. The workflow 'marmot CI on self-hosted hv2' is shown, with a specific run titled 'Fix NIC omission validation on apply #2864' by user 'takara9'. The run status is 'Success' and it took '20m 27s' to complete. The workflow is triggered by a push event on the 'main' branch.

The workflow diagram shows the following steps:

- lint-check-start (35s)
- module-tests (14s)
- system-tests (8s)
- package-config (10s)
- package-etcd-access (4m 17s)
- package-qcow2-manipulation (10s)
- package-util (4m 40s)
- mactl-command-tests (10s)
- package-lvm (1m 11s)
- package-ceph (11s)
- package-virsh (3m 27s)
- package-marmotd-volume (2m 0s)
- package-marmotd-server (4m 0s)
- package-marmotd-etc (20s)
- package-marmotd-osimage (5m 20s)
- package-marmotd-osimage (5m 40s)
- mactl-vm-deploy-basic (5m 40s)
- mactl-vm-deploy-1 (5m 30s)
- mactl-vm-deploy-2 (5m 34s)
- mactl-vm-deploy-3 (5m 0s)
- module-tests-end (3s)
- mte-vm-e2e (4m 0s)
- system-tests-end (8s)
- final-cleanup (10s)

クラウドの基礎技術が俯瞰できた

Application Platform Kubernetes / Container		Security IAM Secure Store Encryption
Managed Service DB / Cache / Queue		
Control Plane API / Controller / Reconcile	Observability Metrics / Log / Tracing	
Infrastructure Compute / Network / Storage		
Hardware / Linux KVM / OVN / LVM		

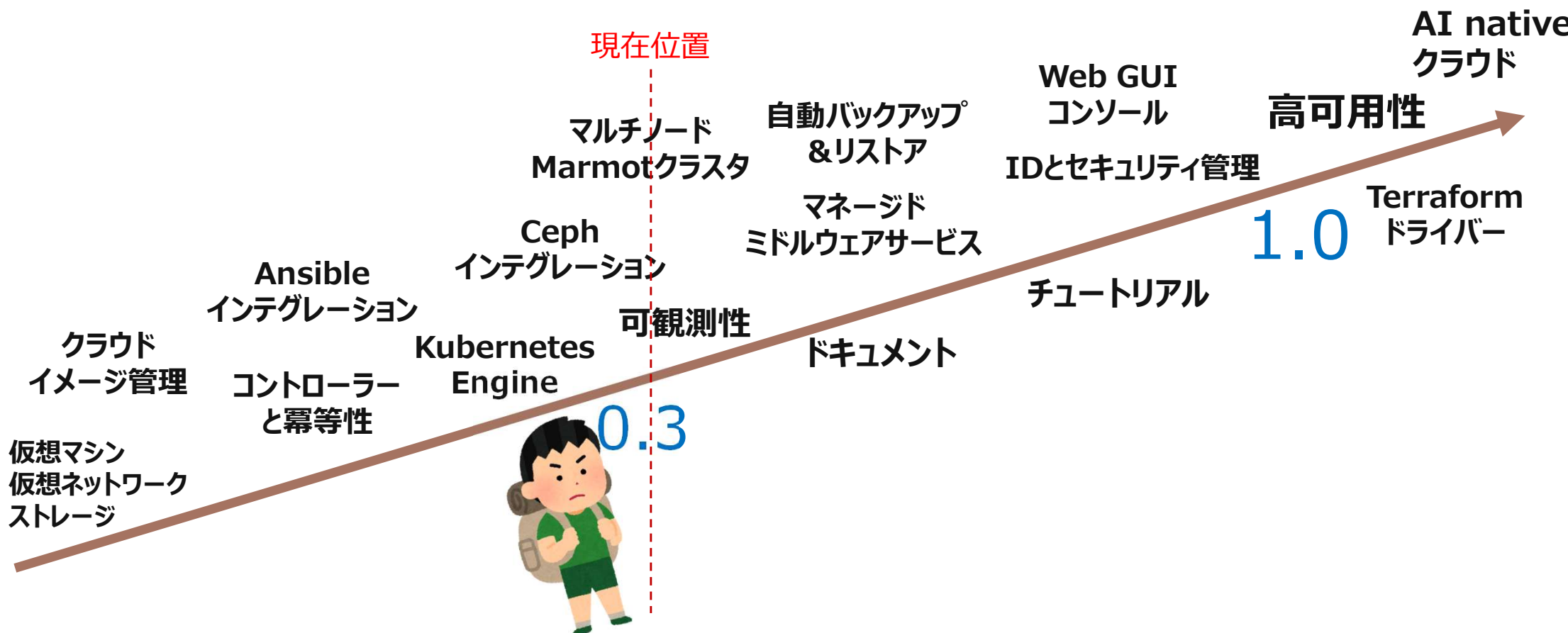


開発を進めて
一つ達成すると
次の目標が見える



Marmotの未来と現在

Marmotのロードマップ



Marmotのリポジトリと情報源

GitHubリポジトリ

<https://github.com/takara9/marmot>

YAMLマニフェスト集

<https://github.com/takara9/marmot-manifests>

解説記事

<https://qiita.com/tags/marmot>

まとめ

- Marmotは、開発途上のソフトウェアです。v1.0までには、色々やるべき事が沢山あります。皆様に使って頂いて、フィードバックを頂くことで、v1.0に成長できると思っています。
- 是非、Marmotを試して、皆様の勉強、研究活動、検証に活かして頂いて、たくさんのフィードバックを頂けますよう、お願いします
- Marmotプロジェクトに、開発 & メンテナーの参加も募集しています。



プライベートクラウドMarmotを よろしくお願いします



10月17日に出展します！
ブースに遊びに来てね

