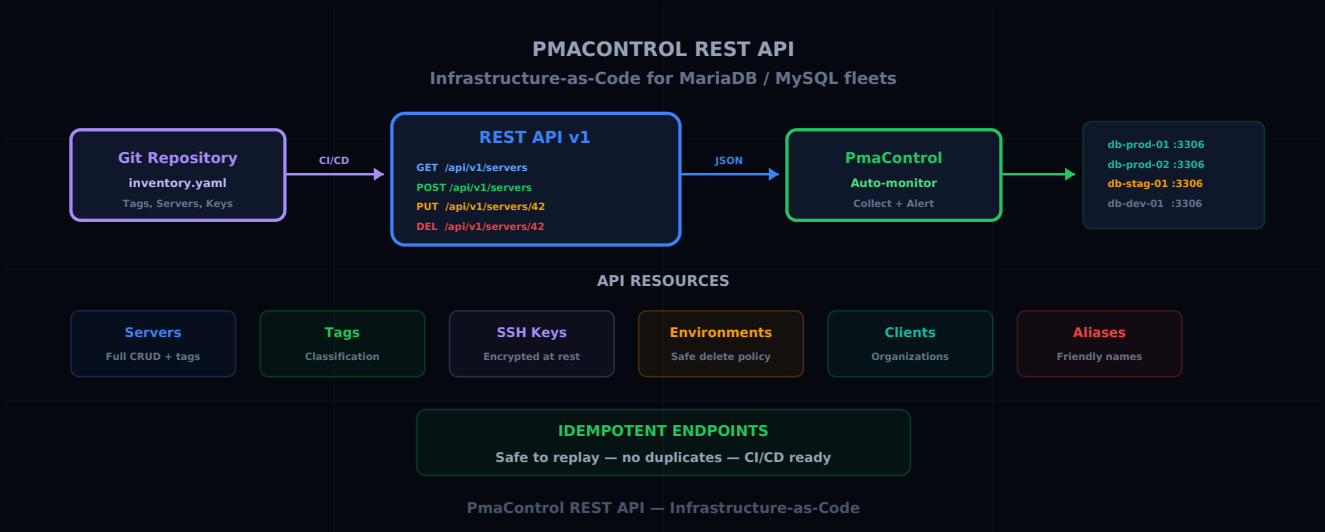
PmaControl REST API : Infrastructure-as-Code pour MariaDB / MySQL

Aurélien LEQUOY · 15 mars 2026

PMACONTROL REST-API INFRASTRUCTURE-AS-CODE DEVOPS AUTOMATION



Pourquoi une API REST ?

PmaControl a commencé comme un outil à interface web. On ajoute un serveur MariaDB / MySQL en cliquant sur des boutons, on configure les tags à la souris, on gère les environnements dans des formulaires.

Cela fonctionne pour 10 serveurs. À 50, c'est pénible. À 200, c'est impossible.

L'API REST de PmaControl transforme l'outil en **plateforme programmable**. Chaque action disponible dans l'interface web est accessible via un endpoint JSON. Cela ouvre la porte à l'Infrastructure-as-Code : décrire son inventaire de bases de données dans des fichiers YAML ou JSON, et l'appliquer automatiquement.

Authentification

L'API utilise une authentification par **utilisateur webservice**. C'est un compte PmaControl dédié, sans accès à l'interface web, dont le token sert de clé d'API.

```
curl -H "Authorization: Bearer <token>" \
     -H "Content-Type: application/json" \
     https://pmacontrol.example.com/api/v1/servers
```

L'utilisateur webservice hérite des ACL de son profil. Un profil "read-only" ne pourra que lister les ressources. Un profil "admin" pourra créer, modifier et supprimer.

Les ressources de l'API

Tags

Les tags sont le système de classification de PmaControl. Chaque serveur peut porter un ou plusieurs tags (production, staging, client-acme, dc-paris, etc.).

```
# Lister tous les tags
GET /api/v1/tags

# Créer un tag
POST /api/v1/tags
{"name": "production", "color": "#22c55e"}

# Modifier un tag
PUT /api/v1/tags/42
{"name": "prod", "color": "#22c55e"}

# Supprimer un tag
DELETE /api/v1/tags/42
```

Les tags sont la brique de base de l'organisation. Ils permettent de filtrer les dashboards, de scoper les alertes, et de restreindre les accès par profil.

Clients

Les clients représentent les organisations ou projets qui possèdent des serveurs.

```
# Lister les clients
GET /api/v1/clients

# Créer un client
```

```
POST /api/v1/clients
{"name": "Acme Corp", "contact_email": "dba@acme.com"}

# Modifier un client
PUT /api/v1/clients/7
{"name": "Acme Corporation"}

# Supprimer un client
DELETE /api/v1/clients/7
```

Environnements

Les environnements (production, staging, développement, etc.) ont une politique de suppression spéciale : **quand un environnement est supprimé, ses serveurs ne sont pas détruits mais réaffectés** à un environnement par défaut.

```
# Lister les environnements
GET /api/v1/environments

# Créer un environnement
POST /api/v1/environments
{"name": "staging", "description": "Pre-production environment"}

# Supprimer — les serveurs sont réaffectés
DELETE /api/v1/environments/3
# Réponse : {"reassigned_servers": 12, "target_environment": "default"}
```

Cette politique évite les suppressions accidentelles de serveurs quand on réorganise les environnements.

Alias

Les alias permettent de référencer un serveur par un nom lisible au lieu de son IP ou hostname interne.

```
# Lister les alias
GET /api/v1/aliases

# Créer un alias
POST /api/v1/aliases
```

```
{"alias": "db-writer-prod", "server_id": 42}
```

Storage Areas

Les storage areas représentent les zones de stockage (datacenters, cloud regions, etc.).

```
# Lister les storage areas
GET /api/v1/storage-areas

# Créer une storage area
POST /api/v1/storage-areas
{"name": "dc-paris-1", "provider": "OVH", "location": "Paris, France"}
```

Clés SSH

PmaControl utilise des clés SSH pour se connecter aux serveurs et collecter les métriques. L'API permet de gérer le cycle de vie des clés.

```
# Lister les clés SSH
GET /api/v1/ssh-keys

# Créer une clé SSH
POST /api/v1/ssh-keys
{
  "name": "pmacontrol-collector-2026",
  "public_key": "ssh-ed25519 AAAA...",
  "private_key": "-----BEGIN OPENSSH PRIVATE KEY-----\n..."
}

# Supprimer une clé
DELETE /api/v1/ssh-keys/5
```

Note de sécurité : les clés privées sont chiffrées au repos dans la base PmaControl. L'API ne retourne jamais la clé privée lors d'un GET.

Serveurs

La ressource principale. Le CRUD complet pour les instances MariaDB / MySQL supervisées.

```
# Lister tous les serveurs
GET /api/v1/servers

# Lister avec filtres
GET /api/v1/servers?tag=production&environment=staging

# Détail d'un serveur
GET /api/v1/servers/42

# Créer un serveur
POST /api/v1/servers
{
  "hostname": "db-prod-01.acme.com",
  "ip": "10.0.1.10",
  "port": 3306,
  "ssh_key_id": 5,
  "client_id": 7,
  "environment_id": 1,
  "tags": ["production", "galera", "dc-paris"]
}

# Modifier un serveur
PUT /api/v1/servers/42
{"port": 3307}

# Assigner des tags
POST /api/v1/servers/42/tags
{"tags": ["production", "critical"]}

# Assigner une clé SSH
PUT /api/v1/servers/42/ssh-key
{"ssh_key_id": 5}

# Supprimer un serveur
DELETE /api/v1/servers/42
```

Workflows Infrastructure-as-Code

L'intérêt principal de l'API est de pouvoir décrire l'inventaire complet dans un fichier versionné et l'appliquer automatiquement.

Exemple : fichier d'inventaire YAML

```yaml
# pmacontrol-inventory.yaml
tags:
  - name: production
    color: "#22c55e"
  - name: staging
    color: "#f59e0b"
  - name: galera
    color: "#14b8a6"

environments:
  - name: production
  - name: staging
  - name: development

ssh_keys:
  - name: collector-2026
    public_key_file: ./keys/collector-2026.pub

servers:
  - hostname: db-prod-01.acme.com
    ip: 10.0.1.10
    port: 3306
    ssh_key: collector-2026
    environment: production
    tags: [production, galera]

  - hostname: db-prod-02.acme.com
    ip: 10.0.1.11
    port: 3306
    ssh_key: collector-2026
    environment: production
    tags: [production, galera]

  - hostname: db-staging-01.acme.com
    ip: 10.0.2.10
    port: 3306
    ssh_key: collector-2026
    environment: staging
    tags: [staging]
```

Script d'application

Un script Python ou Bash lit ce fichier et appelle l'API PmaControl pour synchroniser l'état :

```
#!/bin/bash
API="https://pmacontrol.example.com/api/v1"
TOKEN="your-webservice-token"

# Créer les tags
for tag in production staging galera; do
  curl -s -X POST "$API/tags" \
    -H "Authorization: Bearer $TOKEN" \
    -H "Content-Type: application/json" \
    -d "{\"name\": \"$tag\"}"
done

# Créer les serveurs
curl -s -X POST "$API/servers" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d @server-prod-01.json
```

Intégration CI/CD

Le pattern naturel est d'intégrer cela dans un pipeline CI/CD :

1. Le DBA modifie le fichier `pmacontrol-inventory.yaml` dans Git
2. La merge request est reviewée par l'équipe
3. Le pipeline CI valide la syntaxe et les contraintes (pas de doublon IP, clés SSH valides)
4. Le pipeline CD applique les changements via l'API PmaControl
5. PmaControl commence à superviser les nouveaux serveurs automatiquement

Idempotence

Tous les endpoints de création sont **idempotents** : si la ressource existe déjà (même hostname, même IP), l'API retourne la ressource existante au lieu de créer un doublon. Cela permet de rejouer le script d'inventaire sans effet de bord.

```
# Premier appel : crée le serveur, retourne 201
POST /api/v1/servers  → 201 Created

# Deuxième appel identique : retourne le serveur existant, 200
POST /api/v1/servers  → 200 OK (existing)
```

Codes de retour

Code	Signification
200	Succès (lecture ou mise à jour)
201	Ressource créée
204	Suppression réussie
400	Payload invalide
401	Token manquant ou invalide
403	Permissions insuffisantes
404	Ressource non trouvée
409	Conflit (contrainte d'unicité)

Limites actuelles

L'API v1 couvre les opérations CRUD sur l'inventaire. Elle ne couvre pas encore :

- Les **métriques** (lecture des time-series) — prévu pour v2
- Les **alertes** (configuration des seuils) — prévu pour v2
- Les **backups** (déclenchement et statut) — prévu pour v2
- L'**export de configuration** (my.cnf, ProxySQL rules) — en discussion

Conclusion

L'API REST de PmaControl transforme la gestion d'une flotte MariaDB / MySQL d'un processus manuel en un workflow programmable et versionné.

Décrivez votre inventaire dans Git. Appliquez-le via l'API. Laissez PmaControl superviser automatiquement. C'est l'Infrastructure-as-Code appliquée au monitoring de bases de données.