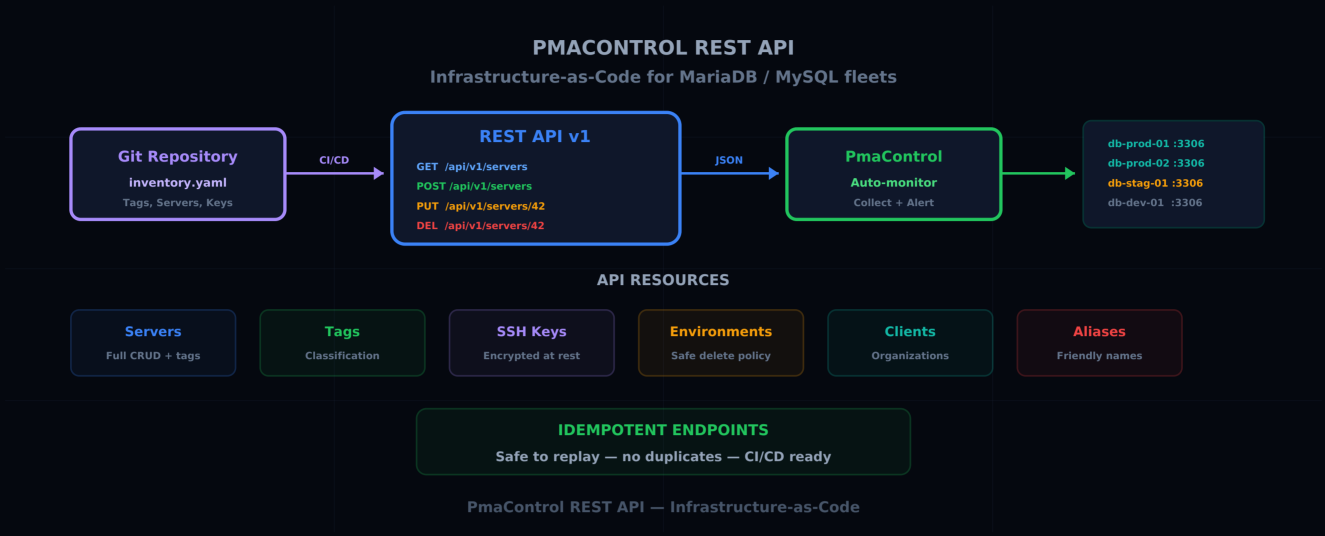
PmaControl REST API：MariaDB / MySQL 的基础设施即代码

Aurélien LEQUOY · 2026年3月15日

PMACONTROL REST-API INFRASTRUCTURE-AS-CODE DEVOPS AUTOMATION



为什么需要 REST API？

PmaControl 最初是一个 Web 界面工具。通过点击按钮添加 MariaDB / MySQL 服务器，用鼠标配置标签，通过表单管理环境。

这对 10 台服务器来说没问题。50 台时就变得繁琐。200 台时则完全不可能。

PmaControl 的 REST API 可以让[illegible]中可用的每个操作都可以通过 JSON 端点访问。这为基础设施即代码打开了大门：用 YAML 或 JSON 文件描述您的数据库清单，并自动应用。

认证

API 使用 **webservice** 用户认证。这是一个专用的 PmaControl 账户，没有 Web 界面访问权限，其令牌充当 API 密钥。

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

webservice 用户继承其配置文件中的 ACL。"只读"配置文件只能列出资源。"管理员"配置文件可以创建、修改和删除。

API 资源

标签

标签是 PmaControl 的分类系统。每个服务器可以携带一个或多个标签（production、staging、client-acme、dc-paris 等）。

```
# 列出所有标签
GET /api/v1/tags

# 创建标签
POST /api/v1/tags
{"name": "production", "color": "#22c55e"}

# 更新标签
PUT /api/v1/tags/42
{"name": "prod", "color": "#22c55e"}

# 删除标签
DELETE /api/v1/tags/42
```

标签是组织的基石。它们允许过滤仪表盘、限定告警范围以及按配置文件限制访问。

客户

客户代表拥有服务器的组织或项目。

```
# 列出客户
GET /api/v1/clients

# 创建客户
POST /api/v1/clients
{"name": "Acme Corp", "contact_email": "dba@acme.com"}

# 更新客户
PUT /api/v1/clients/7
{"name": "Acme Corporation"}

# 删除客户
DELETE /api/v1/clients/7
```

环境

环境（production、staging、development□□□境蓉境府特贫脸删滕策蠡被销毁而是被重新分配

```
# 列出环境
GET /api/v1/environments

# 创建环境
POST /api/v1/environments
{"name": "staging", "description": "Pre-production environment"}

# 删除——服务器被重新分配
DELETE /api/v1/environments/3
# 响应: {"reassigned_servers": 12, "target_environment": "default"}
```

此策略可防止在重组环境时意外删除服务器。

别名

别名允许您使用可读名称而非 IP 或内部主机名来引用服务器。

```
# 列出别名
GET /api/v1/aliases

# 创建别名
POST /api/v1/aliases
{"alias": "db-writer-prod", "server_id": 42}
```

存储区域

存储区域代表存储区域（数据中心、云区域等）。

```
# 列出存储区域
GET /api/v1/storage-areas

# 创建存储区域
POST /api/v1/storage-areas
{"name": "dc-paris-1", "provider": "OVH", "location": "Paris, France"}
```

SSH 密钥

PmaControl 使用 SSH 密钥连接服务器并收集指标。API 管理密钥的生命周期。

```
# 列出 SSH 密钥
GET /api/v1/ssh-keys
```

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

# 删除密钥
DELETE /api/v1/ssh-keys/5
```

支持标准私钥格式。PmaControl 数据库中静态加密存储。API 在 GET 请求中永远不会返回私钥。

服务器

主要资源。受监控的 MariaDB / MySQL 实例的完整 CRUD。

```
# 列出所有服务器
GET /api/v1/servers

# 带过滤条件列出
GET /api/v1/servers?tag=production&environment=staging

# 服务器详情
GET /api/v1/servers/42

# 创建服务器
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"]
}

# 更新服务器
PUT /api/v1/servers/42
```

```
{"port": 3307}

# 分配标签
POST /api/v1/servers/42/tags
{"tags": ["production", "critical"]}

# 分配 SSH 密钥
PUT /api/v1/servers/42/ssh-key
{"ssh_key_id": 5}

# 删除服务器
DELETE /api/v1/servers/42
```

基础设施即代码工作流

API 的主要优势在于能够在版本化文件中描述完整的清单并自动应用。

示例: YAML 清单文件

```
# 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]
```

应用脚本

Python 或 Bash 脚本读取此文件并调用 PmaControl API 来同步状态:

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

# 创建标签
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

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

CI/CD 集成

自然的模式是将其集成到 CI/CD 管道中：

1. DBA 在 Git 中编辑 `control-inventory.yaml` 文件
2. 合并请求由团队审查
3. CI 管道验证语法和约束（无重复 IP、有效的 SSH 密钥）
4. CD 管道通过 PmaControl API 应用变更
5. PmaControl 自动开始监控新服务器

幂等性

所有创建端点都是幂等的：如果资源已存在（相同的主机名、相同的 IP），API 返回现有资源而不是创建重复项。这允许重复运行清单脚本而不会产生副作用。

```
# 第一次调用：创建服务器，返回 201
POST /api/v1/servers  → 201 Created

# 第二次相同调用：返回现有服务器，200
POST /api/v1/servers  → 200 OK (已存在)
```

响应代码

代码	含义
200	成功（读取或更新）
201	资源已创建
204	删除成功
400	无效的请求体
401	缺少或无效的令牌
403	权限不足
404	资源未找到
409	冲突（唯一性约束）

当前限制

v1 API 涵盖清单的 CRUD 操作。尚未涵盖：

- 指标（时间序列读取）——计划在 v2 中实现
- 告警（阈值配置）——计划在 v2 中实现
- 备份（触发和状态）——计划在 v2 中实现
- 配[illegible].cnf、ProxySQL 规则）——讨论中

结论

PmaControl 的 REST API 将 MariaDB / MySQL 集群的管理从手动流程转变为可编程、可版本化的工作流。

在 Git 中描述您的清单。通过 API 应用。让 PmaControl 自动监控。这就是基础设施即代码应用于数据库监控。