微服务常用应用和服务

Consul 部署与使用指南

Consul 部署与使用指南

Consul 是 HashiCorp 公司推出的服务网格解决方案，提供服务发现、健康检查、键值存储和多数据中心支持等功能。以下是 Consul 的完整部署与使用手册。

一、Consul 部署方案

1. 单机模式（开发环境）

```
# 下载 Consul (Linux 示例)
wget https://releases.hashicorp.com/consul/1.15.3/consul_1.15.3_linux_amd64.zip
unzip consul_1.15.3_linux_amd64.zip
sudo mv consul /usr/local/bin/

# 启动开发模式
consul agent -dev -client 0.0.0.0
```

访问 `http://<IP>:8500` 进入 Web UI

2. 集群模式（生产环境）

服务端节点（至少 3台）

```
# 启动第一个服务端
consul agent -server -bootstrap-expect=3 \
  -data-dir=/tmp/consul \
  -node=server1 \
  -bind=192.168.1.101 \
  -ui \
  -client=0.0.0.0
```

```
# 其他服务端加入集群
consul agent -server -data-dir=/tmp/consul \
  -node=server2 \
  -bind=192.168.1.102 \
  -join=192.168.1.101
```

客户端节点

```
consul agent -data-dir=/tmp/consul \
  -node=client1 \
  -bind=192.168.1.201 \
  -join=192.168.1.101
```

3. Docker 部署

```
docker run -d --name=consul \
  -p 8500:8500 \
  -p 8600:8600/udp \
  consul agent -server -ui -bootstrap-expect=1 -client=0.0.0.0
```

二、核心功能使用

1. 服务注册

通过配置文件注册（推荐）

创建 `/etc/consul.d/web.json`：

```
{
  "service": {
    "name": "web",
    "port": 8080,
    "checks": [
      {
        "http": "http://localhost:8080/health",
```

```
      "interval": "10s"
    }
   ]
  }
}
```

重新加载配置:

```
consul reload
```

通过 API 注册

```
curl --request PUT \
  --data @web.json \
  http://localhost:8500/v1/agent/service/register
```

2. 服务发现

DNS 方式

```
dig @127.0.0.1 -p 8600 web.service.consul
```

HTTP API 方式

```
curl http://localhost:8500/v1/catalog/service/web
```

3. 键值存储

```
# 写入
consul kv put config/web/max_conns 25


# 读取
consul kv get config/web/max_conns


# 导出所有 KV
consul kv export > backup.json
```

4. 健康检查

查看节点健康状态:

```
consul monitor
```

三、生产环境最佳实践

1. 多数据中心部署:

   ```
   consul agent -server -datacenter=dc1 ...
   consul agent -server -datacenter=dc2 ...
   ```

2. **ACL** 安全控制:

   ```
   # 启用 ACL
   consul acl bootstrap
   ```

3. 自动加密通信:

   ```
   consul tls ca create
   consul tls cert create -server
   ```

4. 监控集成:
   - 对接 Prometheus
   - 配置 Grafana 仪表盘

四、常见问题排查

1. 节点无法加入集群:

   ```
   consul members
   consul operator raft list-peers
   ```

2. 服务注册失败:

   ```
   consul catalog services
   consul monitor -log-level=debug
   ```

3. 性能问题:

```
consul operator raft stats
```

五、版本升级

```
# 滚动升级步骤
consul leave
# 替换新版本二进制
consul agent -config-dir=/etc/consul.d
```

通过以上指南，您可以快速部署和管理 Consul 集群。Consul 的强大功能可以帮助您构建可靠的微服务架构，实现服务自动发现和健康管理。

Nacos 部署与使用指南

Nacos 部署与使用指南

Nacos 是阿里巴巴开源的服务发现和配置管理平台，支持服务注册、配置管理、服务健康监测等功能。以下是 Nacos 的完整部署与使用手册。

一、 Nacos 部署方案

1. 单机模式（开发环境）

1.1 下载与启动

```
# 下载最新稳定版
wget https://github.com/alibaba/nacos/releases/download/2.2.3/nacos-server-2.2.3.tar.gz
tar -zxvf nacos-server-2.2.3.tar.gz
cd nacos/bin

# 启动（Linux/Mac）
sh startup.sh -m standalone

# Windows
startup.cmd -m standalone
```

访问 `http://localhost:8848/nacos` (默认账号 nacos/nacos)

1.2 Docker 方式

```
docker run --name nacos-standalone -e MODE=standalone -p 8848:8848 -d nacos/nacos-server:v2.2.3
```

2. 集群模式（生产环境）

2.1 准备 3台服务器

```
# 每台服务器修改配置文件 conf/cluster.conf
192.168.1.101:8848
192.168.1.102:8848
192.168.1.103:8848
```

2.2 配置数据库（MySQL）

```
# 创建数据库
CREATE DATABASE nacos_config CHARACTER SET utf8mb4;

# 初始化表结构
USE nacos_config;
SOURCE conf/nacos-mysql.sql
```

2.3 修改配置文件

```
# conf/application.properties
spring.datasource.platform=mysql
db.num=1
db.url.0=jdbc:mysql://127.0.0.1:3306/nacos_config?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true
db.user=root
db.password=yourpassword
```

2.4 启动集群

```
# 每台服务器执行
sh startup.sh
```

二、核心功能使用

1. 服务注册与发现

1.1 Spring Cloud 集成

```xml
<!-- pom.xml -->
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>2022.0.0.0</version>
</dependency>
```

```yaml
# application.yml
spring:
  cloud:
    nacos:
      discovery:
        server-addr: 127.0.0.1:8848
```

1.2 手动注册服务

```
curl -X POST 'http://127.0.0.1:8848/nacos/v1/ns/instance?serviceName=example-service&ip=192.168.1.100&port=8080'
```

2. 配置管理

2.1 创建配置

1. 登录控制台 → 配置管理 → 配置列表
2. 新建配置:
   - Data ID: example.properties
   - Group: DEFAULT_GROUP
   - 配置格式 : Properties
   - 内容 :

```
server.port=8080
spring.datasource.url=jdbc:mysql://localhost:3306/test
```

2.2 Spring Boot 获取配置

```
@RefreshScope
@RestController
public class ConfigController {
```

```
    @Value("${server.port}")
    private String port;

    @GetMapping("/config")
    public String getConfig() {
        return port;
    }
}
```

3. 命名空间与分组

3.1 创建命名空间

1. 控制台 → 命名空间 → 新建
2. 填写命名空间 ID和名称

3.2 使用命名空间

```
spring:
  cloud:
    nacos:
      config:
        namespace: your-namespace-id
        group: YOUR_GROUP
```

三、生产环境最佳实践

1. 高可用架构 :
   - 至少 3节点集群
   - 使用独立 MySQL集群
2. 安全配置 :

```
# conf/application.properties
nacos.core.auth.enabled=true
nacos.core.auth.system.type=nacos
nacos.core.auth.server.identity.key=yourkey
nacos.core.auth.server.identity.value=yourvalue
```

3. 监控告警：
   - 集成Prometheus
   - 配置健康检查告警规则
4. 备份策略：
   - 定期备份MySQL数据
   - 导出重要配置

四、常见问题排查

1. 服务注册失败：

```
# 检查Nacos服务状态
curl http://127.0.0.1:8848/nacos/v1/ns/service/list


# 查看日志
tail -f logs/nacos.log
```

2. 配置不生效：
   - 检查namespace和group是否匹配
   - 确认配置已发布
   - 检查客户端是否有@RefreshScope注解
3. 性能优化：

```
# 调整JVM参数
JAVA_OPT="${JAVA_OPT} -Xms2g -Xmx2g -Xmn1g"
```

五、版本升级

```
# 1. 备份数据和配置
# 2. 停止旧版本
sh shutdown.sh
# 3. 替换新版本
# 4. 启动新版本
sh startup.sh
```

通过本指南，您可以快速部署Nacos服务并应用于生产环境。Nacos作为微服务架构的核心组件，能够有效管理服务发现和配置中心，提高系统可靠性和可维护性。

Kong 安装和部署指南

Kong 安装和部署指南

Kong 是一个云原生、快速、可扩展的微服务抽象层 (API 网关)。以下是 Kong 的安装和部署方法:

1. 安装 Kong

在 Linux 上安装

Ubuntu/Debian

```
# 添加 Kong 仓库
sudo apt-get update
sudo apt-get install -y apt-transport-https curl
echo "deb https://kong.bintray.com/kong-deb `lsb_release -sc` main" | sudo tee -a /etc/apt/sources.list
curl -o bintray.key https://bintray.com/user/downloadSubjectPublicKey?username=bintray
sudo apt-key add bintray.key
sudo apt-get update

# 安装 Kong
sudo apt-get install -y kong
```

CentOS/RHEL

```
# 添加 Kong 仓库
sudo yum install -y wget
wget https://bintray.com/kong/kong-rpm/rpm -O bintray-kong-kong-rpm.repo
sudo mv bintray-kong-kong-rpm.repo /etc/yum.repos.d/
sudo yum update -y

# 安装 Kong
```

```
sudo yum install -y kong
```

□□ Docker

```
# □□ Docker □□
docker network create kong-net

# □□ PostgreSQL □□
docker run -d --name kong-database \
  --network=kong-net \
  -p 5432:5432 \
  -e POSTGRES_USER=kong \
  -e POSTGRES_DB=kong \
  -e POSTGRES_PASSWORD=kong \
  postgres:9.6

# □□□□□
docker run --rm \
  --network=kong-net \
  -e "KONG_DATABASE=postgres" \
  -e "KONG_PG_HOST=kong-database" \
  -e "KONG_PG_PASSWORD=kong" \
  kong:latest kong migrations bootstrap

# □□ Kong
docker run -d --name kong \
  --network=kong-net \
  -e "KONG_DATABASE=postgres" \
  -e "KONG_PG_HOST=kong-database" \
  -e "KONG_PG_PASSWORD=kong" \
  -e "KONG_PROXY_ACCESS_LOG=/dev/stdout" \
  -e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" \
  -e "KONG_PROXY_ERROR_LOG=/dev/stderr" \
  -e "KONG_ADMIN_ERROR_LOG=/dev/stderr" \
  -e "KONG_ADMIN_LISTEN=0.0.0.0:8001, 0.0.0.0:8444 ssl" \
  -p 8000:8000 \
  -p 8443:8443 \
  -p 8001:8001 \
```

```
  -p 8444:8444 \
  kong:latest
```

2. 配置 Kong

基本配置

编辑 `/etc/kong/kong.conf` 或设置环境变量:

```
# 数据库配置
KONG_DATABASE=postgres  # 或 cassandra
KONG_PG_HOST=localhost
KONG_PG_PORT=5432
KONG_PG_USER=kong
KONG_PG_PASSWORD=kong
KONG_PG_DATABASE=kong

# 代理配置
KONG_PROXY_LISTEN=0.0.0.0:8000, 0.0.0.0:8443 ssl
KONG_ADMIN_LISTEN=0.0.0.0:8001, 0.0.0.0:8444 ssl
```

初始化数据库

```
kong migrations bootstrap [-c /path/to/kong.conf]
```

启动 Kong

```
kong start [-c /path/to/kong.conf]
```

3. 验证安装

检查 Kong 是否运行:

```
curl -i http://localhost:8001/
```

应该看到 Kong 的欢迎信息。

4. 基本使用

添加服务

```
curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=example-service' \
  --data 'url=http://mockbin.org'
```

添加路由

```
curl -i -X POST \
  --url http://localhost:8001/services/example-service/routes \
  --data 'hosts[]=example.com'
```

测试代理

```
curl -i -X GET \
  --url http://localhost:8000/ \
  --header 'Host: example.com'
```

5. 生产环境部署建议

1. 使用 **PostgreSQL 集群**：确保数据库高可用
2. 配置 **Kong 集群**：多个 Kong 节点共享同一个数据库
3. 启用 **TLS**：配置 SSL 证书
4. 设置监控：使用 Prometheus 或 Datadog 监控 Kong
5. 配置日志：设置适当的日志级别和日志轮转
6. 使用负载均衡器：在 Kong 节点前放置负载均衡器

6. 常见插件

Kong 的强大之处在于其插件系统，常用插件包括:

- 认证： Keycloak, JWT, OAuth2, Basic Auth
- 安全： CORS, IP Restriction, Bot Detection
- 流量控制： Rate Limiting, Request Size Limiting
- 监控： Prometheus, Datadog
- 日志： File Log, HTTP Log, TCP Log

希望这个指南能帮助您顺利安装和部署 Kong！如需更详细的配置，请参考 Kong 官方文档。

Kong 图形管理界面解决方案

Kong 图形管理界面解决方案

Kong 本身提供了一个强大的 RESTful Admin API，但默认不包含图形用户界面 (GUI)。以下是几种为 Kong 添加图形管理界面的方法：

1. Konga - 最流行的 Kong GUI

安装 Konga

Konga 是最受欢迎的 Kong 图形管理工具，提供完整的 API 网关管理功能。

使用 Docker 安装 Konga

```
# 创建 PostgreSQL 容器用于 Konga (可选，也可以使用现有数据库)
docker run -d --name konga-database \
  --network=kong-net \
  -p 5433:5432 \
  -e POSTGRES_USER=konga \
  -e POSTGRES_DB=konga \
  -e POSTGRES_PASSWORD=konga \
  postgres:13

# 初始化 Konga 数据库
docker run --rm \
  --network=kong-net \
  pantsel/konga:latest -c prepare -a postgres -u postgres://konga:konga@konga-database:5432/konga

# 启动 Konga
docker run -d -p 1337:1337 \
  --network=kong-net \
  -e "DB_ADAPTER=postgres" \
```

```
-e "DB_URI=postgres://konga:konga@konga-database:5432/konga" \
-e "KONGA_HOOK_TIMEOUT=120000" \
-e "NODE_ENV=production" \
--name konga \
pantsel/konga:latest
```

访问 Konga

访问 `http://localhost:1337`，首次访问需要注册管理员账户。

连接 Konga 到 Kong

1. 登录后点击 "CONNECTIONS" → "NEW CONNECTION"
2. 填写 Kong Admin URL (通常是 `http://kong:8001` 或 `http://localhost:8001`)
3. 点击 "CONNECT"

2. Kong Manager (企业版)

Kong 企业版提供了官方图形界面 Kong Manager，功能包括:

- 可视化 API 管理
- 用户和权限管理
- 插件配置
- 监控和数据分析

安装 Kong 企业版

```
# 下载企业版包后
sudo apt-get install ./kong-enterprise-edition-*.deb
```

3. 其他开源替代方案

Kong Dashboard

```
# 使用 npm 安装
npm install -g kong-dashboard


# 启动
kong-dashboard start --kong-url http://localhost:8001


# 高级启动选项
kong-dashboard start \
  --kong-url http://kong:8001 \
  --port 8080 \
  --basic-auth user1=password1 user2=password2
```

4. 图形界面功能对比

功能	**Konga**	**Kong Manager**	**Kong Dashboard**
服务管理	✓	✓	✓
路由管理	✓	✓	✓
插件配置	✓	✓	✓
消费者管理	✓	✓	✓
证书管理	✓	✓	✗
集群管理	✓	✓	✗
监控仪表板	✓	✓	✗
多环境支持	✓	✓	✗
用户权限管理	✓	✓	✗
插件市场	✗	✓	✗

5. 生产环境建议

1. 启用认证：为 GUI 配置基本认证或 OAuth
2. 使用 **HTTPS**：通过 Nginx 或 Kong 本身为 GUI 启用 SSL
3. 限制访问：只允许内部网络或 VPN 访问管理界面
4. 定期备份：特别是 Konga 的数据库
5. 监控：监控 GUI 应用的健康状态

6. 常见问题解决

问题 1：Konga 无法连接 Kong

- 检查 Kong 的 `admin_listen` 地址是否正确
- 确认网络连通性（特别是 Docker 网络）
- 检查 Kong 是否启用了 Admin API (`KONG_ADMIN_LISTEN`)

问题 2：界面加载缓慢

- 增加 Konga 的 `KONGA_HOOK_TIMEOUT` 环境变量
- 优化数据库性能（为 PostgreSQL 添加索引）
- 考虑增加 Konga 实例的资源

问题 3：权限不足

- 确保连接 Kong 时使用了正确的 Admin API 密钥（如果配置了）
- 检查 Kong 的 RBAC 配置（企业版）

选择适合您需求的 GUI 方案可以大大简化 Kong 的管理工作，特别是对于不熟悉 REST API 的团队成员。

Kong 认证与鉴权配置指南

Kong 提供了多种认证和鉴权机制来保护您的 API 服务。以下是 Kong 的主要认证鉴权方式及其配置方法。

1. 基本认证 (Basic Authentication)

启用基本认证插件

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=basic-auth" \
  --data "config.hide_credentials=true"
```

创建消费者

```
curl -X POST http://localhost:8001/consumers \
  --data "username=johndoe"
```

添加凭证

```
curl -X POST http://localhost:8001/consumers/johndoe/basic-auth \
  --data "username=john" \
  --data "password=doe"
```

测试请求

```
curl -i -X GET http://localhost:8000/{route-path} \
  -H "Authorization: Basic $(echo -n 'john:doe' | base64)"
```

2. JWT 认证 (JSON Web Tokens)

启用 JWT 插件

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=jwt" \
  --data "config.claims_to_verify=exp" \
  --data "config.key_claim_name=kid"
```

创建消费者

```
curl -X POST http://localhost:8001/consumers \
  --data "username=jwt-user"
```

生成 JWT 凭证

```
curl -X POST http://localhost:8001/consumers/jwt-user/jwt
```

使用 JWT 访问

```
curl -i -X GET http://localhost:8000/{route-path} \
  -H "Authorization: Bearer <JWT_TOKEN>"
```

3. OAuth2 认证

启用 OAuth2 插件

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=oauth2" \
  --data "config.enable_authorization_code=true" \
  --data "config.enable_client_credentials=true" \
  --data "config.mandatory_scope=true" \
  --data "config.scopes=email,phone,address" \
  --data "config.token_expiration=7200"
```

□□□□

```
curl -X POST http://localhost:8001/consumers \
  --data "username=oauth2-user"
```

□□ OAuth2 □□

```
curl -X POST http://localhost:8001/consumers/oauth2-user/oauth2 \
  --data "name=MyApp" \
  --data "redirect_uris=http://myapp.com/callback" \
  --data "client_id=CLIENT_ID" \
  --data "client_secret=CLIENT_SECRET"
```

□□□□□

```
curl -X POST http://localhost:8000/{route-path}/oauth2/token \
  --data "grant_type=client_credentials" \
  --data "client_id=CLIENT_ID" \
  --data "client_secret=CLIENT_SECRET" \
  --data "scope=email"
```

4. Keycloak □□ (OpenID Connect)

□□ OIDC □□

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=openid-connect" \
  --data "config.issuer=https://keycloak.example.com/auth/realms/myrealm" \
  --data "config.client_id=CLIENT_ID" \
  --data "config.client_secret=CLIENT_SECRET" \
  --data "config.auth_methods=authorization_code,bearer" \
  --data "config.scopes=openid,profile,email" \
  --data "config.redirect_uri=https://myapp.com/callback"
```

5. ACL 访问控制

启用 ACL 插件

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=acl" \
  --data "config.allow=group1,group2" \
  --data "config.hide_groups_header=true"
```

为消费者添加 ACL 组

```
curl -X POST http://localhost:8001/consumers/{consumer}/acls \
  --data "group=group1"
```

6. 组合认证策略

Kong 支持多种认证方式的组合使用:

```
# 启用基本认证
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=basic-auth"

# 启用 JWT 认证
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=jwt" \
  --data "config.anonymous=<anonymous-consumer-id>"
```

7. 高级配置技巧

匿名访问控制

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=key-auth" \
  --data "config.anonymous=<anonymous-consumer-id>"
```

跨域认证 (CORS)

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=cors" \
  --data "config.origins=*" \
  --data "config.methods=GET,POST" \
  --data "config.headers=Accept,Authorization" \
  --data "config.credentials=true"
```

速率限制保护认证端点

```
curl -X POST http://localhost:8001/services/{service}/plugins \
  --data "name=rate-limiting" \
  --data "config.minute=5" \
  --data "config.policy=local"
```

8. 生产环境最佳实践

1. 使用 **HTTPS**: 所有认证流量必须加密
2. 定期轮换密钥: 特别是 JWT 签名密钥和 OAuth2 密钥
3. 最小权限原则: 只授予必要的权限
4. 监控认证尝试: 检测暴力破解行为
5. 结合 **WAF**: 使用 Web 应用防火墙增强保护
6. 日志审计: 记录所有认证相关活动

9. 故障排除

常见问题 **1**: 认证失败

- 检查消费者是否已创建
- 验证凭证是否正确
- 检查插件配置是否正确

- 查看 Kong 错误日志

常见问题 **2**：**JWT** 过期

- 检查令牌有效期
- 验证签名算法是否匹配
- 确认密钥是否正确

常见问题 **3**：**OAuth2** 回调问题

- 检查 redirect_uri 是否匹配
- 验证客户端 ID 和密钥
- 确保授权范围正确

通过合理配置这些认证机制，您可以构建一个安全可靠的 API 网关系统。