

部署文档

1. 系统概述

本系统是一个部署在阿里云云端服务器，利用微服务架构和主从数据库集群，使用 B/S 架构为软件开发团队提供敏捷开发的开发过程流程支持的 web 服务系统。

2. 部署目标

- **可靠性和稳定性**：确保应用程序在生产环境中能够稳定运行，减少故障和宕机的风险。强调监控、错误处理、容错机制和高可用性配置等方面。
- **性能和可扩展性**：提供良好的性能和可扩展性，以满足用户的需求。你可以提及负载均衡、读写分离、数据库优化、并行处理等方面的优化措施。
- **安全性**：强调保护用户数据和应用程序免受安全威胁的目标。使用SSL/TLS加密、防火墙配置、访问控制、数据库加密和安全审计等方面的安全措施。
- **可维护性和可管理性**：使应用程序易于维护和管理。自动化部署、配置管理、日志记录、监控和警报系统等方面的优秀实践。
- **灵活性和可扩展性**：强调你的部署目标是支持将来的增长和变化。如容器化、微服务架构、API设计和集成等方面的灵活性和可扩展性策略。
- **持续集成和持续交付**：实现持续集成和持续交付流程，以快速交付新功能和修复。关于版本控制、自动化测试、持续集成工具和部署流水线等方面的实践。

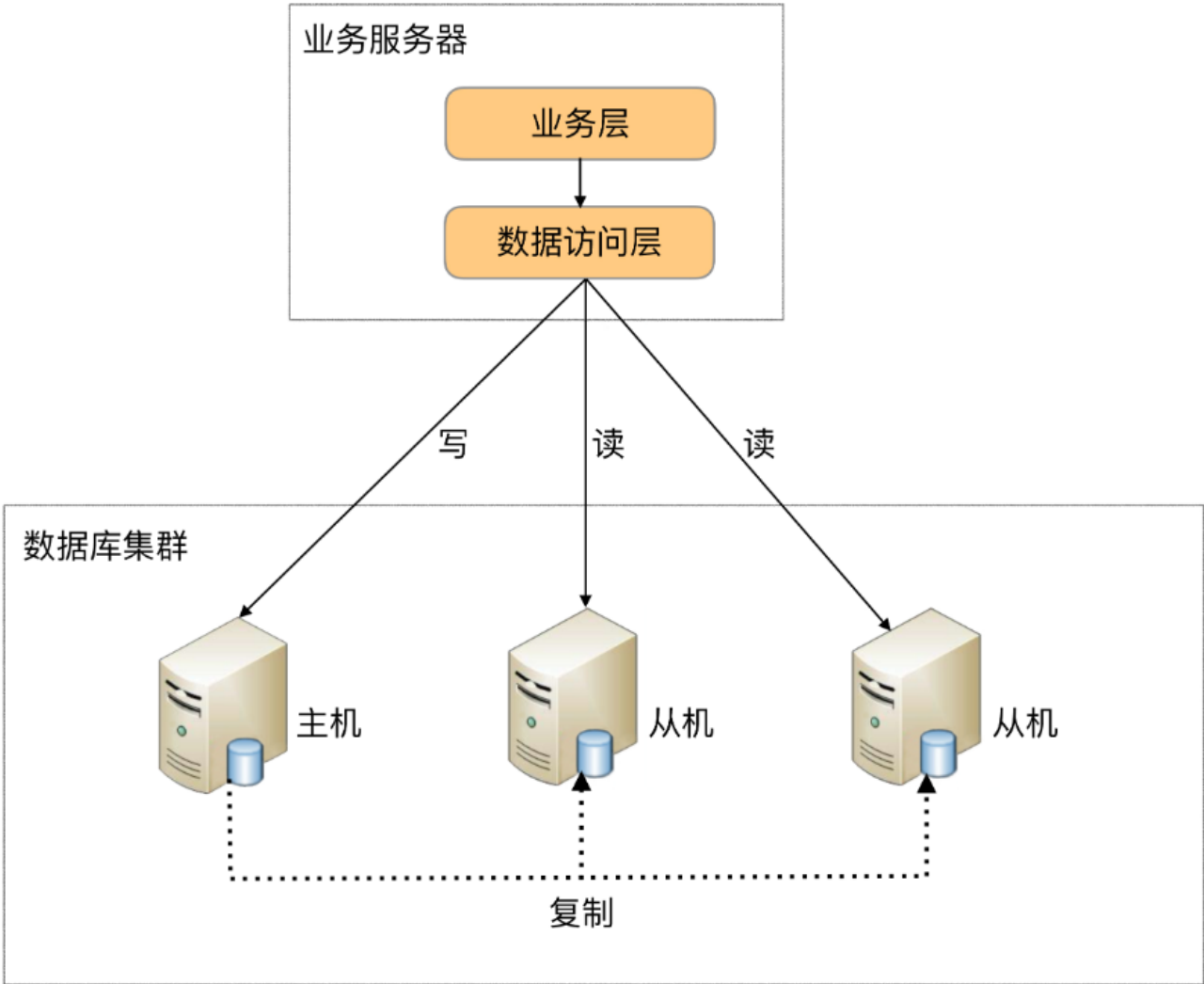
3. 部署配置

两台阿里云服务器：121.41.227.227，101.37.116.97

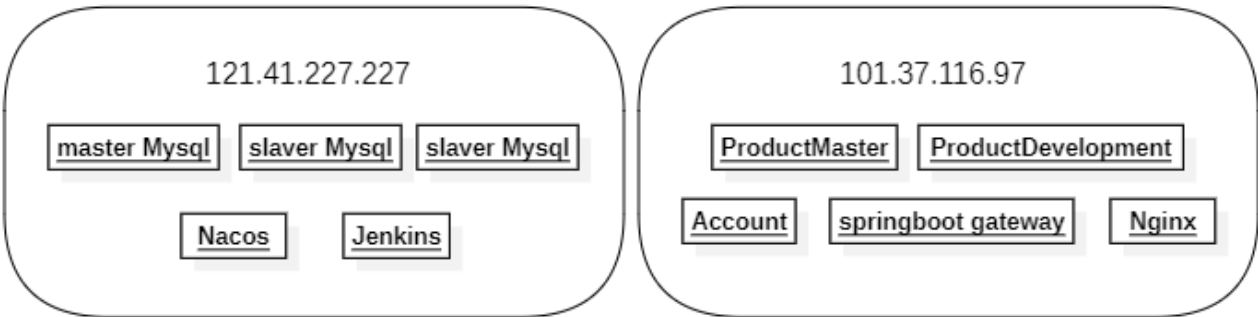
- 所在区域：杭州
- CPU & 内存：2核（vCPU） 4GiB
- 操作系统：Ubuntu 22.04 64位 UEFI版
- 公网带宽：100Mbps（峰值）
- 硬盘空间：ESSD 40GiB（2120 IOPS）
- JDK 17
- Docker

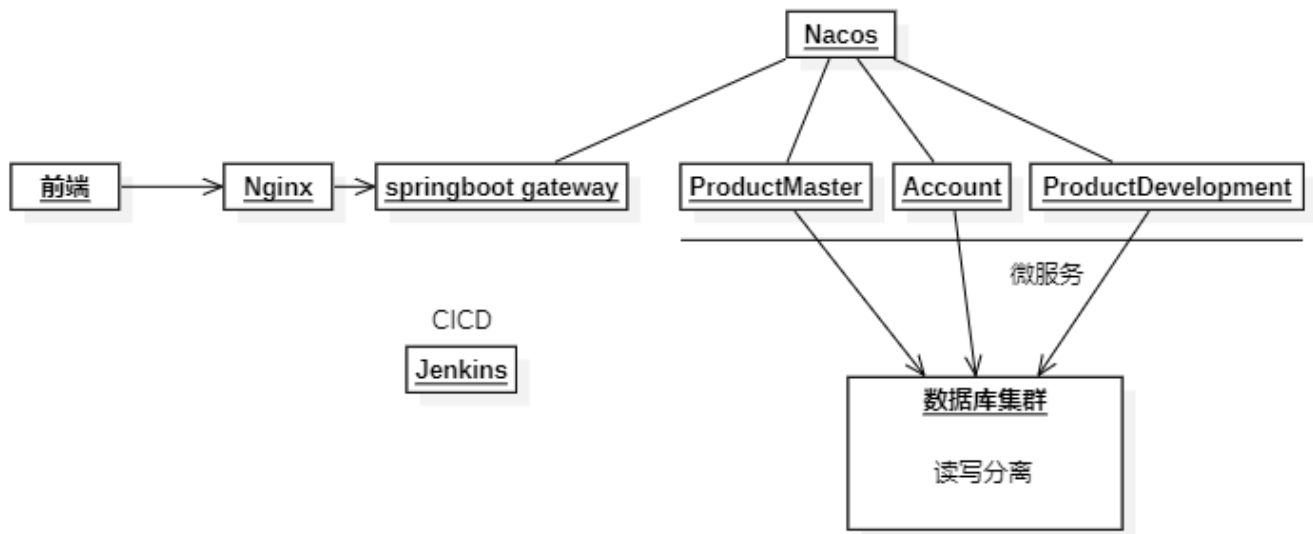
4. 部署架构

数据库主从架构



微服务架构





5. 部署步骤

CI：代码提交阶段

技术：GitHub, git

流程：代码提交阶段也称为版本控制。提交是将开发人员编写的最新代码变更发送到代码存储库的操作。开发人员编写的代码的每个版本都被无限期地存储。在与合作者讨论和审查变更之后，开发人员将编写代码，并在软件需求、特性增强、bug修复或变更请求完成后提交。管理编辑和提交变更的存储库称为源代码管理工具（配置管理工具）。在开发人员提交代码（代码推送请求）后，代码更改被合并到主线代码分支中，这些主线代码分支存储在GitHub这样的中央存储库中。

The screenshot shows the GitHub repository page for **JiCode-backend**. The repository is private and has 7 branches and 1 tag. The main branch is selected. The commit history shows a recent push by **ProductMa** 7 minutes ago. The commit list includes:

- fix: 解决pull冲突 (2 weeks ago)
- Initial commit (3 weeks ago)
- fix: Generator bug (2 weeks ago)
- feat: 新建产品 (新建产品成员) (2 days ago)
- Initial commit (3 weeks ago)
- Update .gitignore (8 minutes ago)
- Initial commit (3 weeks ago)
- README.md (3 weeks ago)
- mvnw (3 weeks ago)
- mvnw.cmd (3 weeks ago)
- pom.xml (last week)


The contributors list includes **wuhang03** and **MomoyamaSawa**.

CI：构建

技术：Maven，jenkins

流程：持续集成过程的目标是提交的代码持续构建为二进制文件或构建产物。通过持续集成来检查添加的新模块是否与现有模块兼容，不仅有助于更快地发现bug，还有助于减少验证新代码更改的时间。构建工具可以根据几乎所有编程语言的源代码创建可执行文件或包（.exe，.dll，.jar等）。

在完成这些检查后，将向流水线中执行UT（**单元测试**），以进一步减少生产中的故障。单元测试可验证开发人员编写的单个单元或组件是否按预期执行。

 Jenkins

Dashboard > ProductMa >

📄 状态

</> 修改记录

📁 工作空间

▶ 立即构建

⚙️ 配置

🗑️ 删除 Maven project

📁 模块

✏️ 重命名

Maven project ProductMa

相关链接

- 最近一次构建(#15),5 天 0 小时之前
- 最近稳定构建(#13),7 天 9 小时之前
- 最近成功的构建(#13),7 天 9 小时之前
- 最近失败的构建(#15),5 天 0 小时之前
- 最近不稳定的构建(#7),7 天 9 小时之前
- 最近未成功的构建(#15),5 天 0 小时之前
- 最近完成的构建(#15),5 天 0 小时之前

☁️ 构建历史 趋势 ▾

🔍 过滤构建... /

❌ #15

2024年1月1日 下午2:30

❌ #14

2024年1月1日 下午2:30

✅ #13

2023年12月30日 上午5:57

❌ #12


2023年12月30日 上午5:51

✅ #11

2023年12月30日 上午5:51

❌ #10

2023年12月30日 上午5:49

 **Jenkins**

🔍 查找 (CTRL+K) ? 🛡️ 1 👤 jicode_user 🚪 注销

Dashboard > ProductMa > #13

📄 状态集

</> 变更记录

🖨️ 控制台输出

📄 编辑编译信息

🗑️ 删除构建 '#13'

🔗 Git Build Data

🔄 Redeploy Artifacts

🔍 查看指纹

⬅️ 上一次构建

➡️ 下一次构建

✅ #13 (2023年12月30日 上午5:57:17)

永久保留这次编译

✎ 添加说明 启动时间: 7 天 20 小时
构建用时: 17 秒

</> No changes.

🕒 启动用户 jicode_user

🔗 git
Revision: 14c1d58f40413d79c77ac7e9c18db4991f8c0bb2
Repository: https://user:ghp_ZHrJopCGfd68ixZgOkRAGCjGiUGpA0ttHu3@github.com/JiCode-TJSE/JiCode-backend.git

- refs/remotes/origin/ProductMa

Module Builds

✅ ProductMa 4.9 秒

✅ main 2.3 秒

CI：测试

技术：apifox, Jmeter

过程：发布构建过程后的一系列自动/人工测试将验证代码的准确性。此阶段可帮助避免生产中的错误。在 api 尺度上进行集成测试。对于由多个团队提交和构建代码的大型组织，这些检查在并行环境中运行，以节省宝贵的时间并尽早将错误通知开发人员。

🏠 主页 JCode

🔍 接口管理

📁 ProductDev (24)

- 📁 project (5)
 - 📁 backlogItems (9)
 - 🗑️ 删除需求
 - ➕ POST 新建需求/工作项 (1)
 - 🔍 GET 我参与的工作项 (1)
 - 🔍 GET 我负责的工作项 (1)
 - 🔍 GET 查询全部需求/工作项 (3)
 - ➕ PUT 更新工作项 (1)
 - ➕ POST 关联工作项 (1)
 - 🔍 GET 解除关联
 - 🔍 GET 查看需求详情
 - 📁 sprint (5)
 - ➕ POST 新建迭代 (1)
 - 🔍 GET 分页查询迭代
 - 🗑️ 删除迭代 (1)
 - ➕ PUT 修改迭代
 - 🔍 GET 获取迭代具体信息
 - 📁 release (4)
 - ➕ POST 新建发布 (1)
 - 🔍 GET 获取所有发布信息 (1)
 - 🗑️ 删除发布 (1)
 - 🔍 GET 获取发布具体信息
 - 📁 schedule (1)
 - ➕ Account (6)
 - 📁 ProductMa (20)
 - 📁 product (5)

📄 文档 修改文档 运行 高级 Mock

GET http://101.37.116.97:8082/api/productdev/backlogitem

Params 1 Body Headers Cookies Auth 前置操作 后置操作 设置

Query 参数

参数名	参数值	类型	说明
id	Microservice-6	string	

添加参数

Body Cookie Header 8 控制台 实际请求

Pretty Raw Preview Visualize JSON utB

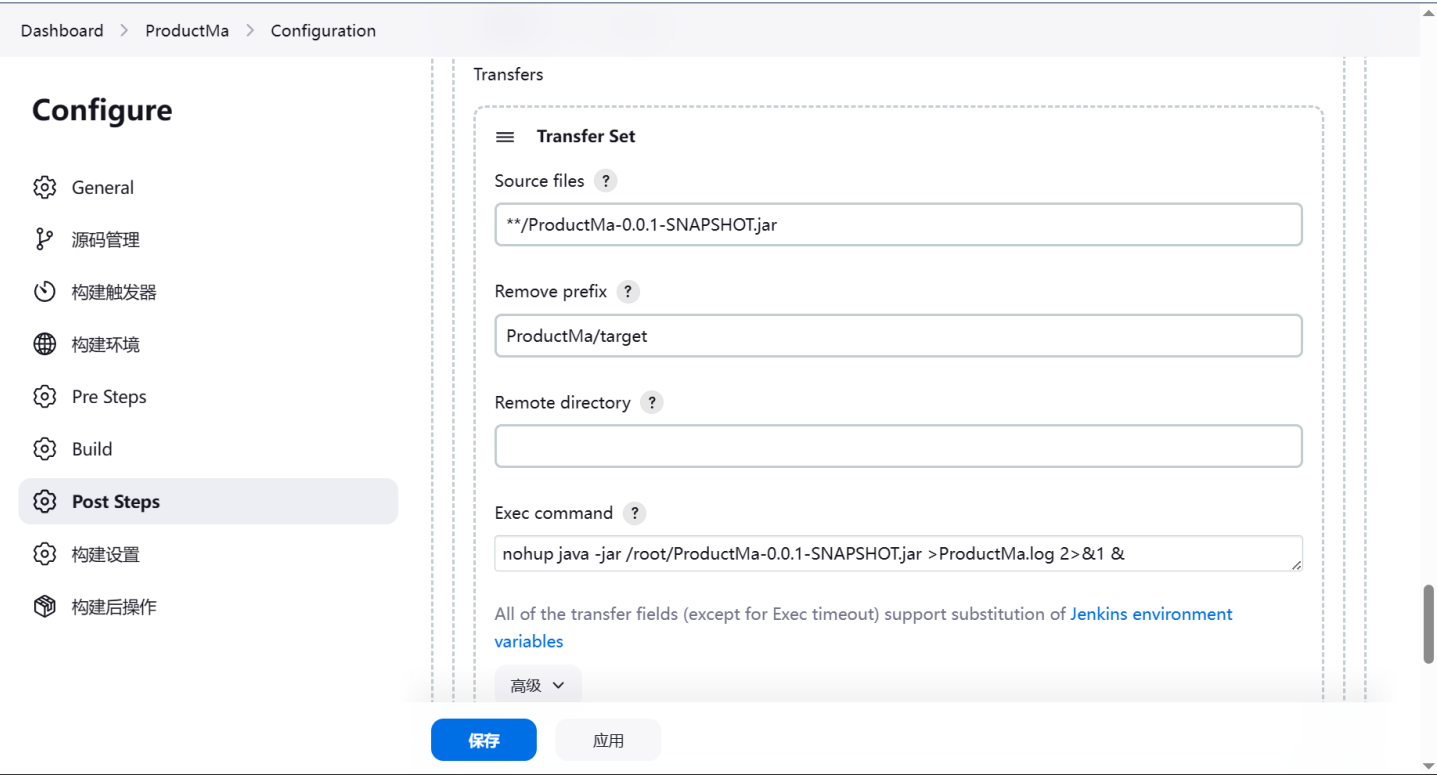
```
1 {
2   "code": 200,
3   "msg": "请求成功",
4   "responseCode": "SUCCESS",
5   "data": {
6     "id": "Microservice-6",
7     "priority": "高",
8     "startTime": "2024-01-02T00:00:00.000+00:00",
9     "endTime": "2024-01-02T00:00:00.000+00:00",
10    "source": "产品规划",
11    "description": "11",
12    "projectId": "6e0326ca-5f75-4104-82fb-b585c6788138",
13    "managerId": "d4cc64a1-ade4-4532-84dc-1ad54ae8df90",
14    "scheduleId": null,
15    "topic": "11",
16    "status": "status",
17    "organizationId": "1",
18    "memberIds": [],
19    "sprintIds": [],
20    "releaseIds": [],
21    "backlogItemIds": [
22      "Microservice-7"
23    ]
24  }
25 }
```

🔍 状态码: 200 耗时: 62 ms 大小: 511 B

CD：部署

技术：Jenkins、Docker

Jenkins 将在测试服务器上构建好的产物和配置自动部署到生产服务器上并自动运行上线。



CD：验证

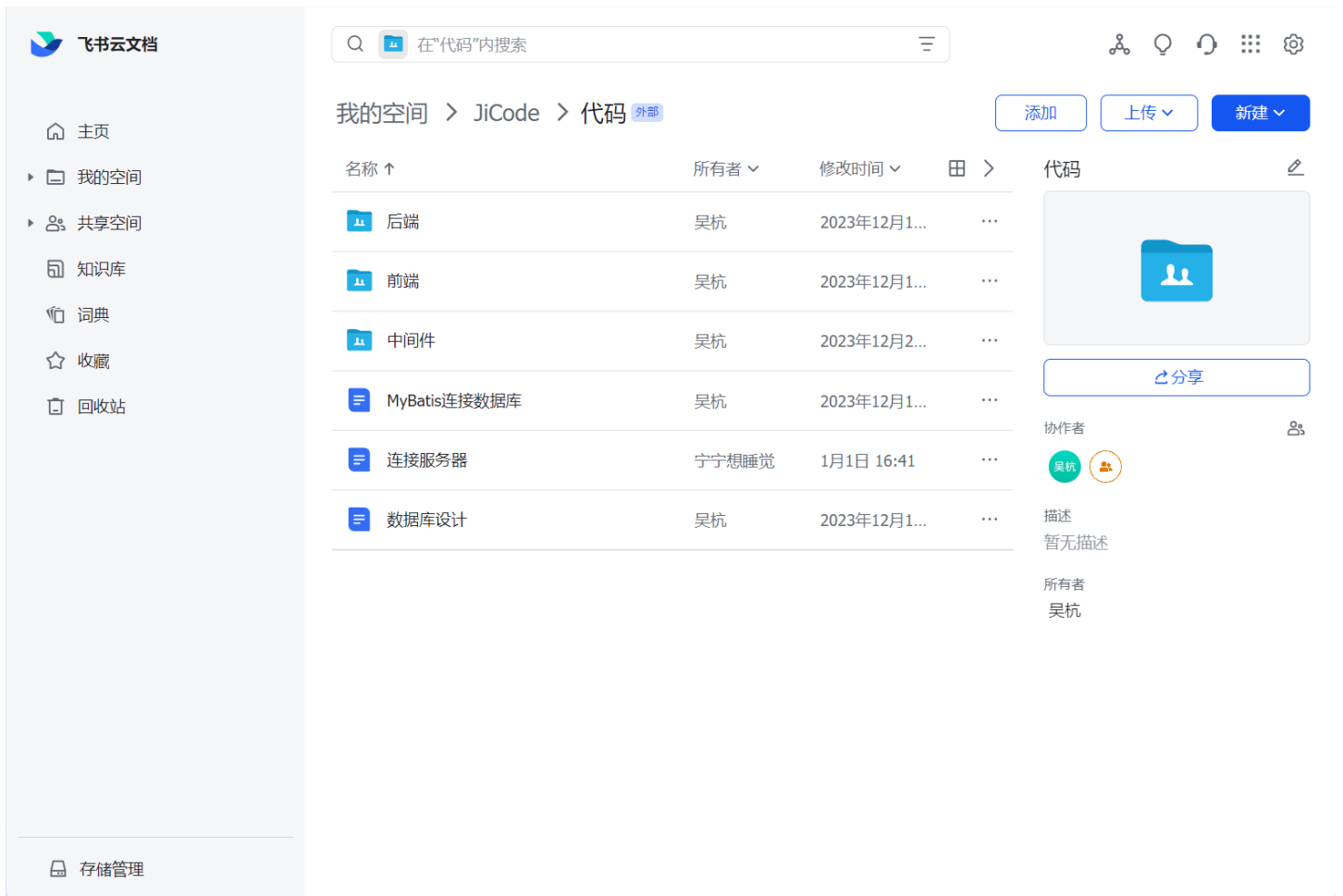
这也是团队优化整个CI/CD流程的关键位置。因为现在已经进行了如此多的测试，所以失败很少见。但是，此时必须尽快解决所有故障，以最大程度地减少对最终客户的影响。这边我们是使用我们的前端网页来验证我们的服务。（详情见前端演示视频）

CD：监控

为了使软件发行版具有故障安全性和健壮性，在生产环境中跟踪发行版的运行状况至关重要。应用程序监视工具将跟踪性能指标，例如CPU利用率和发行版延迟。日志分析器将扫描由底层中间件和操作系统产生的大量日志，以识别行为并跟踪问题的根源。如果生产中出现任何问题，将通知利益相关者以确保生产环境的安全性和可靠性。（详情见监控和日志部分）

CD：反馈与协作

我们利用飞书来进行团队协作，同步记录存在的问题反馈并且团队共享协作实现即时修正。



6. 配置文件/命令行设置

Nacos

- 主要设置了模式为单机启动和设置版本，限制了nacos占用内存的大小，而且让nacos连接上了远程数据库实现nacos配置持久化保存。

```
docker run -d -p 8848:8848 -p 9848:9848 --restart=always -e  
MODE=standalone -e SPRING_DATASOURCE_PLATFORM=mysql -e  
MYSQL_SERVICE_HOST=121.41.227.227 -e MYSQL_SERVICE_PORT=3306 -e  
MYSQL_SERVICE_DB_NAME=nacos -e MYSQL_SERVICE_USER=root -e  
MYSQL_SERVICE_PASSWORD=Elaina1017? --name nacos --memory=1024m  
nacos/nacos-server:latest
```

Jenkins

- 限制jenkins占用内存大小和设置版本。

```
docker run -d -p 8080:8080 -m 720m --memory-swap 720m --name  
jenkins_container jenkins/jenkins:lts-jdk17
```

Ngnix

- 配置https反向代理，配置ssl证书密钥等，连接网关等。

```

1      server {
2          listen 443 ssl;
3          server_name localhost;

4
5          ssl_certificate <证书路径>;
6          ssl_certificate_key <密钥路径>;
7
8          location / {
9              proxy_pass http://127.0.0.1:<网关端口>;
10             proxy_set_header Host $host;
11             proxy_set_header X-Real-IP $remote_addr;
12             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
13             proxy_set_header X-Forwarded-Proto $scheme;
14         }
15     }

```

Springboot gateway

- 主要是配置负载均衡和路由转发，解决跨域问题等。

```

1  server:
2    port: 10010 # 网关端口
3  spring:
4    application:
5      name: gateway # 服务名称
6    cloud:
7      nacos:
8        server-addr: http://121.41.227.227:8848 # nacos地址
9    gateway:
10     routes: # 网关路由配置
11       - id: <route id># 路由id, 自定义, 只要唯一即可
12         # uri: http://127.0.0.1:8081 # 路由的目标地址 http就是固定地址
13         uri: lb://<微服务名># 路由的目标地址 lb就是负载均衡, 后面跟服务名称
14         predicates: # 路由断言, 也就是判断请求是否符合路由规则的条件
15           - Path=/api/<url前缀>/** # 这个是按照路径匹配, 只要以/user/开头就符合要求
16     globalcors: # 全局的跨域处理
17       add-to-simple-url-handler-mapping: true # 解决options请求被拦截问题
18     corsConfigurations:
19       '[/**]':
20         allowedOriginPatterns: # 允许所有网站的跨域请求
21           - "*"
22         allowedMethods: # 允许的跨域ajax的请求方式
23           - "GET"
24           - "POST"

```



```
25         - "DELETE"
26         - "PUT"
27         - "OPTIONS"
28     allowedHeaders: "*" # 允许在请求中携带的头信息
29     allowCredentials: true # 是否允许携带cookie
30     maxAge: 360000 # 这次跨域检测的有效期
```

数据库主从

- master端口：3306；slaver端口：3307，3308
- 运行指令

```
docker run --name <name> -e MYSQL_ROOT_PASSWORD=<数据库密码> -p <暴露端口>:3306 -d mysql:latest
```

- 主库配置:

```
1 server-id=1 // 集群里的唯一id
2 read-only=0 // 只写
3 binlog-do-db=<指定数据库>
```

- 从库配置:

```
1 CHANGE REPLICATION SOURCE TO SOURCE_HOST=<主库host>, SOURCE_USER=<主库用户>,
2 SOURCE_PASSWORD=<用户密码>, SOURCE_LOG_FILE=<主库源log文件>,
3 SOURCE_LOG_POS=<源log文件位置>, get_master_public_key=<集群唯一id>;
```

微服务部署

- ProductMaster端口：8080；Account端口：8081；ProductDevelopment端口：8082

使用Jenkins进行部署，所以以下均是Jenkins中的配置，jenkins装了maven插件和SSHposter插件

- Git

```
https://user:<token>@github.com/JiCode-TJSE/JiCode-backend.git
```

- 准备阶段

`./start.sh <微服务名>-0.0.1-SNAPSHOT`（脚本是将之前运行的进程杀死，准备部署新程序）

```
1 appname=$1
2 pid=`ps -ef | grep $1 | grep 'java -jar' | awk '{printf $2}'`
```

```

3 if [ -z $pid ];
4     then
5         echo "$appname not started"
6     else
7         kill -9 $pid
8         echo "$appname stopping..."
9 fi

```

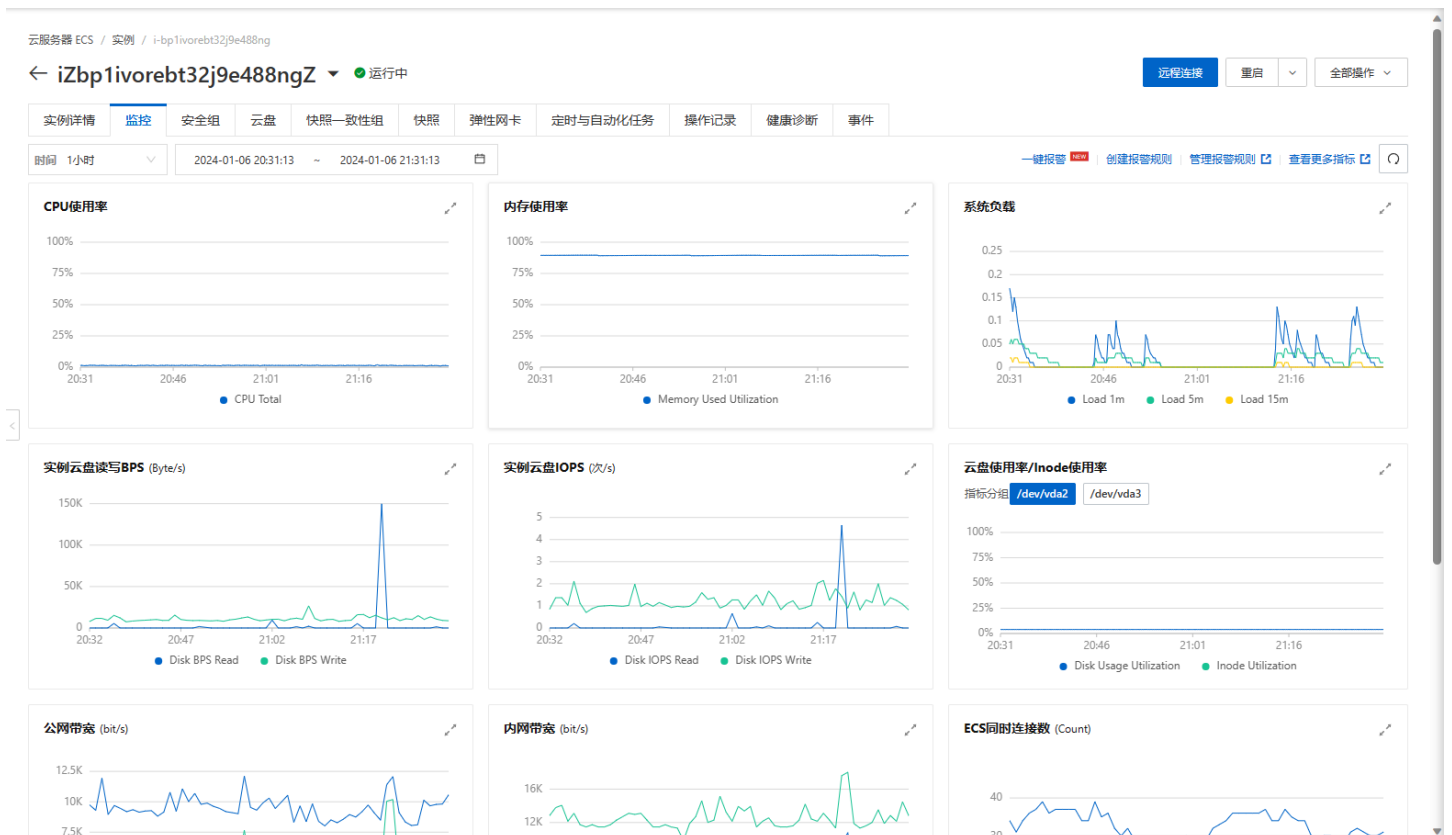
部署阶段

- 1 Source files: **/<微服务名>-0.0.1-SNAPSHOT.jar
- 2 Remove prefix: ProductMa/target
- 3 Exec command: `nohup java -jar -Dserver.port=<指定端口> -Dspring.cloud.nacos.discovery.ip=101.37.116.97 /root/<微服务名>-0.0.1-SNAPSHOT.jar ><微服务名>.log 2>&1 &`

7. 监控和日志

阿里云服务器监视

- 实时监视部署服务器的状态，如CPU/内存使用率，在超过阈值时发送报警短信/邮件。



docker日志和springboot日志系统

- 定期查看docker日志（下图1），检查各中间件的运行状态；而微服务的运行状态由springboot的日志系统记录到文件中（下图2），方便查错与检错。

```
root@i7bplivorebt32j9e488ngZ:~# docker logs jenkins
Running from: /usr/share/jenkins/jenkins.war
webroot: /var/jenkins_home/war
2023-12-29 15:42:25.601+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2023-12-29 15:42:27.166+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2023-12-29 15:42:27.277+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-10.0.18; built: 2023-10-27T01:59:58.245Z; git: 8545fd9bf4cd0d0838f626b405fd49634
41546b7; jvm 17.0.9+9
2023-12-29 15:42:27.605+0000 [id=1] INFO o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2023-12-29 15:42:27.661+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdManager#doStart: Session workerName=node0
2023-12-29 15:42:28.267+0000 [id=1] INFO hudson.WebAppMain$ContextListener#start: Jenkins home directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HO
ME")
2023-12-29 15:42:28.729+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started w.@6778aea6{Jenkins v2.426.2,,file:///var/jenkins_home/war/,AVAILABLE}{/var/je
nkins_home/war}
2023-12-29 15:42:28.742+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started ServerConnector@235a0c16{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2023-12-29 15:42:28.764+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started Server@31e4bb20{STARTING}{10.0.18,sto=0} @3942ms
2023-12-29 15:42:28.774+0000 [id=25] INFO winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2023-12-29 15:42:28.979+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Started initialization
2023-12-29 15:42:28.998+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2023-12-29 15:42:29.924+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2023-12-29 15:42:29.930+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: Started all plugins
2023-12-29 15:42:29.937+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2023-12-29 15:42:30.138+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
2023-12-29 15:42:30.139+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2023-12-29 15:42:30.139+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2023-12-29 15:42:30.141+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2023-12-29 15:42:30.199+0000 [id=47] INFO hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2023-12-29 15:42:30.807+0000 [id=31] INFO jenkins.install.SetupWizard#init:

*****
*****
*****
```

docker 查看日志

```
application.yml Account\... application.yml ProductMa\... 1 application.log X
ProductMa > application.log
1 2024-01-06T21:40:32.352+08:00 WARN 15644 --- [ProductManagementService] [main] c.a.nacos.client.logging.NacosLogging : Load Logback Configuration of Nacos fail,
message: Could not initialize Logback Nacos logging from classpath:nacos-logback.xml
2 2024-01-06T21:40:32.486+08:00 WARN 15644 --- [ProductManagementService] [main] c.a.nacos.client.logging.NacosLogging : Load Logback Configuration of Nacos fail,
message: Could not initialize Logback Nacos logging from classpath:nacos-logback.xml
3 2024-01-06T21:40:32.514+08:00 INFO 15644 --- [ProductManagementService] [main] c.jiCode.ProductMa.ProductMaApplication : Starting ProductMaApplication using Java
21.0.1 with PID 15644 (D:\Study\JiCode\code\JiCode-backend\ProductMa\target\classes started by nene in D:\Study\JiCode\code\JiCode-backend\ProductMa)
4 2024-01-06T21:40:32.517+08:00 INFO 15644 --- [ProductManagementService] [main] c.jiCode.ProductMa.ProductMaApplication : No active profile set, falling back to 1
default profile: "default"
5 2024-01-06T21:40:34.627+08:00 WARN 15644 --- [ProductManagementService] [main] o.s.w.s.m.m.a.RequestMappingHandlerMapping : No MyBatis mapper was found in '[com.
example.myapp.mappers]' package. Please check your configuration.
6 2024-01-06T21:40:36.245+08:00 WARN 15644 --- [ProductManagementService] [main] o.mybatis.spring.SqlSessionFactoryBean : Property 'mapperLocations' was specified
but matching resources are not found.
7 2024-01-06T21:40:37.546+08:00 DEBUG 15644 --- [ProductManagementService] [main] s.w.s.m.m.a.RequestMappingHandlerMapping : 24 mappings in
'requestMappingHandlerMapping'
8 2024-01-06T21:40:37.723+08:00 DEBUG 15644 --- [ProductManagementService] [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Patterns [/webjars/**, /**] in
'resourceHandlerMapping'
9 2024-01-06T21:40:37.769+08:00 DEBUG 15644 --- [ProductManagementService] [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : ControllerAdvice beans: 0
@ModelAttribute, 0 @InitBinder, 1 RequestBodyAdvice, 1 ResponseBodyAdvice
10 2024-01-06T21:40:37.821+08:00 DEBUG 15644 --- [ProductManagementService] [main] .m.m.a.ExceptionHandlerExceptionHandlerResolver : ControllerAdvice beans: 1
@ExceptionHandler, 1 ResponseBodyAdvice
11 2024-01-06T21:40:38.181+08:00 WARN 15644 --- [ProductManagementService] [main] c.a.nacos.client.logging.NacosLogging : Load Logback Configuration of Nacos fail,
message: Could not initialize Logback Nacos logging from classpath:nacos-logback.xml
12 2024-01-06T21:40:39.350+08:00 INFO 15644 --- [ProductManagementService] [main] c.jiCode.ProductMa.ProductMaApplication : Started ProductMaApplication in 7.948
seconds (process running for 10.029)
```

微服务自动生成日志文件

8. 安全考虑

网络传输安全

- 使用SSL/TLS加密通信：SSL通过在通信双方之间建立安全的加密连接，确保传输的数据在传输过程中不被窃听、篡改或伪造。SSL使用公钥加密和私钥解密的技术，通过证书验证和密钥交换等步骤来确保通信的机密性和完整性。（下图展示用https安全协议访问我们的api，使用https需要服务端配置ssl加密，这边暂时使用自签名证书，所以浏览器会报警告）

```
1 {
2   "msg": "请求成功",
3   "code": 200,
4   "data": {
5     "id": "dd91e3f8-2323-4d47-8a9c-648ea64bc5f3",
6     "title": "JiCode",
7     "detail": "JiCode",
8     "mark": "JiCode",
9     "team_name": "JiCode",
10    "team_id": "test_team"
11  }
12 }
```

数据库存储安全

- 敏感数据哈希加密：将敏感数据（如密码、个人信息等）转换为不可逆的散列值的技术。哈希加密算法将输入数据转换为固定长度的字符串，这个过程是单向的，不可逆的，即无法从散列值反推出原始数据。通过将敏感数据进行哈希加密，可以提高数据的安全性，即使数据库被攻击或泄露，攻击者也无法轻易获得原始数据。（下图展示密码存储用的是不可逆的MD5哈希算法，即使得到了这张数据表数据，也破解不了密码，在程序层面也是用同样的算法进行哈希值对比来鉴别用户身份）

account 输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)					
	account_id	organization_id	email	phone_number	password
1	0e92f261-504c-442e-9123-94e8ee7f6c00	1	2154298@tongji.edu.cn	[NULL]	e10adc3949ba59abbe56e057f20f883e
2	0eed0085-e741-45ee-8e72-310f7f3d6618	1	2496803478@qq.com	[NULL]	698d51a19d8a121ce581499d7b701668
3	201f50c5-3b8f-4950-b3b2-68810b3cae82	1	152977702@qq.com	[NULL]	55fc5b709962876903785fd64a6961e5
4	8541700f-850f-4d71-ab3e-c6b985448ed4	1	x.pfqpnod@qq.com	[NULL]	41443b79e93433746487aa4f9fec2c6b
5	d4cc64a1-ade4-4532-a4dc-1ad54ae0df90	1	wuhang1008@163.com	[NULL]	9aa945a84eee4f8c0e9c62e88d675519
6	da7a7313-ae32-4020-9f9f-6567bada26f5	1	1156504938@qq.com	[NULL]	202cb962ac59075b964b07152d234b70

9. 回滚计划

- Github 代码版本回滚