# 云原生博客项目各阶段操作代码集合

## 第一阶段：搭建 LNMP 环境及安装 WordPress

### 规范说明

1. **服务启动命令格式**：systemctl start nginx mariadb php-fpm（去除括号，使用空格分隔多服务）
2. **路径一致性**：WordPress 根目录为/var/www/wordpress，后续配置文件修改路径需统一
3. **安全提示强化**：数据库密码建议通过环境变量或独立配置文件管理，避免明文写入文档

### 操作代码

1. **安装 Nginx 服务**

|  |
| --- |
| yum install -y nginx |

1. **安装 Remi 仓库（适用于 CentOS 7）**

|  |
| --- |
| yum install -y https://rpms.remirepo.net/enterprise/remi-release-7.rpm |

1. **启用 PHP 7.4 模块**

|  |
| --- |
| yum-config-manager --enable remi-php74 |

1. **安装 PHP 及必要扩展**

|  |
| --- |
| yum install -y php php-fpm php-mysqlnd php-mbstring php-gd php-xml php-json php-zip |

1. **安装 MariaDB 数据库**

|  |
| --- |
| yum install -y mariadb-server mariadb |

1. **初始化数据库安全设置**

|  |
| --- |
| mysql\_secure\_installation  # 按提示设置root密码（初始为空，直接回车后设置新密码），后续选项全部输入Y |

1. **管理服务状态**

|  |
| --- |
| systemctl start nginx mariadb php-fpm # 启动服务  systemctl enable nginx mariadb php-fpm # 设置开机自启  systemctl status nginx mariadb php-fpm # 检查服务状态（返回active表示启动成功） |

1. **配置 Nginx 支持 PHP**

|  |
| --- |
| mkdir -p /var/www/wordpress # 创建网站根目录  chown -R nginx:nginx /var/www/wordpress # 授权Nginx用户访问 |

1. **创建 Nginx 配置文件**

|  |
| --- |
| vim /etc/nginx/conf.d/wordpress.conf |

粘贴以下内容：

|  |
| --- |
| server {  listen 80;  server\_name your-domain.com; # 请替换为实际域名（本地测试可保留localhost）  root /var/www/wordpress;  index index.php index.html index.htm;  location / {  try\_files $uri $uri/ /index.php?$args;  }  location ~ \.php$ {  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  include fastcgi\_params;  }  } |

1. **验证并重启 Nginx**

|  |
| --- |
| nginx -t # 检查配置语法  systemctl restart nginx # 应用配置 |

1. **创建 WordPress 数据库**

|  |
| --- |
| mysql -u root -p # 登录MariaDB（输入数据库root密码） |

在数据库终端执行：

|  |
| --- |
| CREATE DATABASE wordpress;  CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'StrongDBPassword123!'; # 强密码示例  GRANT ALL PRIVILEGES ON wordpress.\* TO 'wpuser'@'localhost';  FLUSH PRIVILEGES;  EXIT; |

1. **下载并配置 WordPress**

|  |
| --- |
| cd /var/www/wordpress  wget https://wordpress.org/latest.zip  unzip latest.zip  mv wordpress/\* .  rm -rf wordpress latest.zip  cp wp-config-sample.php wp-config.php  vim wp-config.php # 配置数据库信息（重点修改以下内容） |

关键配置：

|  |
| --- |
| define('DB\_NAME', 'wordpress');  define('DB\_USER', 'wpuser');  define('DB\_PASSWORD', 'StrongDBPassword123!'); # 与数据库用户密码一致  define('DB\_HOST', 'localhost'); |

权限调整：

|  |
| --- |
| chown -R nginx:nginx /var/www/wordpress |

1. **完成安装**

通过浏览器访问http://ECS公网IP，按向导完成初始化设置。

## 第二阶段：数据库上云（以阿里云 RDS 为例）

### 技术优势（高可用性体现）

1. **多可用区容灾**：主备库部署在不同物理机房，避免单点硬件故障（如阿里云 "可用区 A + 可用区 B" 架构）
2. **自动化故障转移**：内置心跳检测，主库异常时 30 秒内自动切换备库，应用无感知（RTO≤30 秒）
3. **数据可靠性保障**：
   1. 定时全量备份（存储于 OSS，保留 1-365 天）
   2. Binlog 增量日志支持任意时间点恢复（如误删数据可恢复至 1 分钟前）
   3. 半同步复制确保主备数据强一致性
4. **免运维架构**：自动处理主从同步、监控告警、版本升级，降低人工运维风险

### 操作代码

1. **备份本地数据库**

|  |
| --- |
| mysqldump -u root -p --database wordpress > /tmp/wordpress\_rds\_migrate.sql  # 提示：执行后输入本地数据库root密码 |

1. **导入 RDS 数据库**

|  |
| --- |
| mysql -h rds-endpoint.mysql.rds.aliyuncs.com -P 3306 -u wpuser -p --database wordpress < /tmp/wordpress\_rds\_migrate.sql  # 提示：输入RDS账号密码（建议通过安全方式存储，避免明文记录） |

1. **验证数据迁移**

|  |
| --- |
| mysql -h rds-endpoint.mysql.rds.aliyuncs.com -P 3306 -u wpuser -p wordpress  # 登录后执行：SELECT \* FROM wp\_posts LIMIT 1; 检查数据是否存在 |

1. **更新 WordPress 数据库配置**

|  |
| --- |
| vim /var/www/wordpress/wp-config.php # 关键配置修改如下 |

替换为 RDS 配置：

|  |
| --- |
| define('DB\_HOST', 'rds-endpoint.mysql.rds.aliyuncs.com:3306'); # RDS私网地址+端口  define('DB\_NAME', 'wordpress'); # 与RDS数据库名一致  define('DB\_USER', 'wpuser'); # RDS专用账号  define('DB\_PASSWORD', 'RDS-Strong-Password!'); # RDS账号密码 |

## 第三阶段：静态资源优化

### 方案解析

#### 方案一：OSS+CDN（推荐）

**前提条件**：

1. 已备案域名（根据《互联网信息服务管理办法》，中国内地服务器必须备案）
2. 基础成本：域名（约 10 元 / 年）+ 3 个月 ECS（约 200 元）+ OSS 存储（按用量计费，1GB 月存储约 0.12 元）

**核心优势**：

1. **静态资源分离**：将图片、CSS、JS 等存储于 OSS，降低 ECS 存储压力
2. **全球加速**：CDN 节点缓存静态资源，用户就近获取，延迟降低 70%+
3. **流量卸载**：OSS 直接处理静态资源请求，ECS 仅处理动态逻辑（CPU 利用率可下降 60%）

#### 方案二：GitHub Pages（适用于海外用户）

**限制**：国内访问延迟较高（建议搭配 Cloudflare CDN 中转），本人已经部署在国外平台，奈何身处内地，访问网站全天超时。

#### 方案三：Nginx 反向代理（需备案）

**配置要点**：

|  |
| --- |
| # 在Nginx配置中添加静态资源重定向规则  location ~\* \.(jpg|jpeg|png|gif|css|js|ico|woff|woff2)$ {  proxy\_pass https://your-oss-bucket.oss-cn-hangzhou.aliyuncs.com; # OSS域名  proxy\_redirect off;  proxy\_set\_header Host $host;  } |

**注意**：未备案域名访问 OSS 会触发强制下载，需通过备案域名绑定 OSS Bucket。

## 第四阶段：负载均衡与高可用架构

### 高可用性实现路径

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 高可用能力提升 | 技术实现 | 故障恢复时间 |
| 第一阶段 | 单点架构（无高可用） | 单 ECS 部署 LNMP | 故障恢复需人工介入 |
| 第二阶段 | 数据库高可用 | RDS 多可用区主备 + 自动故障转移 | ≤30 秒 |
| 第三阶段 | 静态资源高可靠 | OSS 跨区域冗余存储 + CDN 节点备份 | 存储故障无感知 |
| **第四阶段** | **应用层高可用** | 负载均衡 + 多 ECS 实例 + 健康检测 | 秒级故障切换 |

### 操作步骤

1. **创建 ECS 实例镜像**
   1. 基于已配置的 LNMP 环境创建自定义镜像（阿里云控制台 - ECS - 镜像 - 创建镜像）
   2. 使用镜像创建 2 台 ECS（主：可用区 A，备：可用区 B，同一 VPC）
2. **配置负载均衡（SLB）**
   1. 创建传统型负载均衡实例（监听端口 80，协议 HTTP）
   2. 添加后端服务器：主 ECS 权重 90，备 ECS 权重 10（加权轮询算法）
   3. 开启健康检查（HTTP 检查路径/health.php，响应 200 视为正常）
3. **自动化故障切换脚本（示例逻辑）**

|  |
| --- |
| # 检测主服务器状态  while true; do  STATUS=$(curl -s -o /dev/null -w %{http\_code} http://主ECS公网IP)  if [ $STATUS -ne 200 ]; then  # 从负载均衡中移除故障服务器  aliyun slb RemoveBackendServers --LoadBalancerId lb-xxx --BackendServers 主ECS实例ID  # 发送告警通知（邮件/Slack）  curl -X POST https://api.example.com/alert -d "主服务器故障"  fi  sleep 30 # 每30秒检测一次  done |

## 与传统博客系统的核心区别

|  |  |  |
| --- | --- | --- |
| 对比维度 | 传统博客系统 | 云原生博客系统 |
| **数据库架构** | 单实例本地部署（单点风险） | 云端高可用 RDS（多副本 + 自动容灾） |
| **静态资源** | 本地存储（带宽瓶颈） | 分布式 OSS+CDN（全球加速 + 流量卸载） |
| **应用部署** | 单服务器部署 | 负载均衡 + 多 ECS 实例（弹性扩展） |
| **故障恢复** | 人工排查 + 手动恢复（小时级） | 自动化故障转移（秒级切换） |
| **运维成本** | 需手动管理数据库、服务器 | 免运维服务（RDS/SLB/OSS 自动管理） |
| **数据可靠性** | 依赖本地备份（易丢失） | 云端多重备份（全量 + 增量 + 异地冗余） |

## 总结

本方案通过**分阶段架构升级**实现云原生高可用性：

1. **基础架构**：LNMP 环境搭建，实现博客系统基本运行（单点架构）
2. **数据上云**：迁移数据库至 RDS，解决数据层单点风险，获得多可用区容灾能力
3. **资源分离**：静态资源存储至 OSS 并通过 CDN 加速，降低服务器压力，提升用户访问速度
4. **应用扩展**：负载均衡结合多 ECS 实例，实现应用层高可用，支持弹性扩展与故障自动切换

每个阶段均针对传统博客系统的痛点（单点故障、性能瓶颈、高运维成本）提供解决方案，最终构建出**数据可靠、访问高效、故障自愈**的云原生博客系统。