认识 Nginx - 起源与使命

Nginx：从俄罗斯走向世界的顶级Web服务器。

创始人：Igor Sysoev

2004

历史背景：发布于2004年，旨在解决C10K并发连接问题。

```
// C10K: 10,000 concurrent connections
```

核心功能

高性能HTTP服务器

极高的稳定性与低资源消耗。

反向代理服务器

灵活分发与负载均衡。

IMAP/POP3/SMTP代理服务器

可靠的邮件处理。

生产环境最佳实践

强调安全性

结构化参数呈现

第二页：环境准备 – 运行需求

稳健运行的基石：依赖包与系统环境。

环境检查清单

操作系统

- 主流Linux发行版（CentOS，Ubuntu，Debian）

核心依赖库

- GCC编译器（代码编译基础）
- PCRE库（用于伪静态规则）
- zlib库（用于Gzip压缩支持）
- OpenSSL库（用于HTTPS加密通信）

依赖安装命令（模拟终端）

CentOS / RHEL

```
[root@server ~]# yum install -y gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
```

Ubuntu / Debian

```
[root@server ~]# apt-get update
[root@server ~]# apt-get install -y build-essential libpcre3 libpcre3-dev zlib1g zlib1g-dev libssl-dev
```

生产环境提示：务必使用官方源或受信镜像源，定期更新补丁以确保安全性。

部署实战 - 安装方式

灵活选择：从快速部署到深度定制。

快捷安装：软件包管理 (yum/apt-get)

```
yum install nginx
或
apt-get install nginx
```

优点：简单快速，自动处理依赖，适合快速上手。
缺点：版本可能较旧，定制性受限。

深度定制：源码编译

```
./configure --prefix=/usr/local/nginx
make && make install
```

优点：高度灵活，可定制模块和参数，性能优化。
缺点：编译过程复杂，需手动管理依赖，维护成本高。

云原生：容器化部署 (Docker)

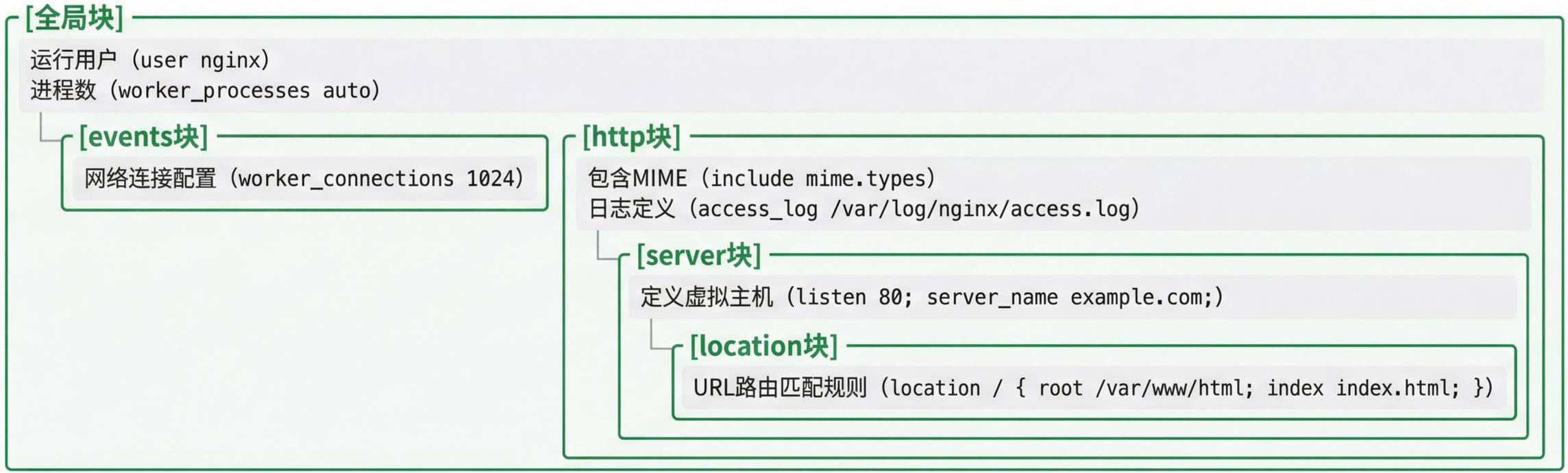
```
docker pull nginx
docker run -d -p 80:80 nginx
```

优点：环境一致性，快速交付，易于扩展和迁移。
缺点：增加了容器技术栈的学习成本，需要理解容器网络和存储。

生产环境建议：始终关注安全性配置与版本更新，结合需求选择最适合的部署模式，确保系统稳定与安全。

第四页：架构初探 - 基础配置

核心配置文件 nginx.conf 的层级奥秘。



CORE POINTS:

- 理论与实践结合，通过代码示例增强可理解性
- 结构化呈现技术参数，确保信息自明性
- 强调生产环境的最佳实践与安全性

服务运维 - 控制指令

Core Message：精准操控：管理 Nginx 进程的艺术。

指令类型	命令示例（代码块）	说明与区别
常用命令		
启动	`nginx`	启动 Nginx 服务。
停止	`nginx -s stop`	立即停止服务。
热加载配置	`nginx -s reload`	平滑重新加载配置文件，不中断现有连接。与重启的主要区别。
优雅退出	`nginx -s quit`	处理完当前请求后停止。
系统管理		
启动/重启	`systemctl start/restart nginx`	使用 systemd 管理服务，适用于现代 Linux 系统。
配置检查	`nginx -t`	**至关重要！**在执行任何重载或重启前，必须先检查配置文件语法是否正确，确保安全性。

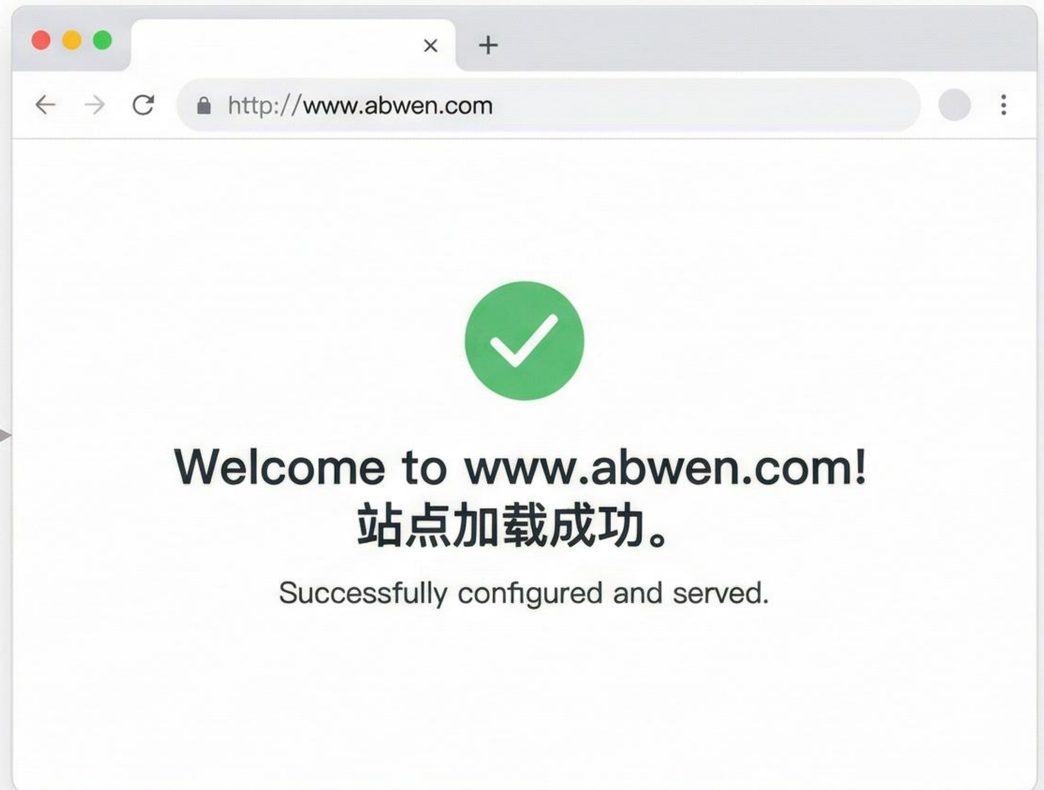
第六页：案例演示 – 配置 Web 站点

Nginx 配置示例（nginx.conf）

```
server {
    # 1. 定义 Server
    listen 80;
    server_name www.abwen.com;

    # 2. 设置根目录
    root /www/abwen;
    index index.html;

    # 3. 日志记录
    access_log logs/abwen.access.log main;
}
```



- **核心要点**：理论与实践结合，通过代码示例增强可理解性。
- 结构化呈现技术参数，确保信息自明性。
- 强调生产环境的最佳实践与安全性。

第七页：进阶之道 - 日常管理

精细化管理，确保服务器长治久安。

日志切割

防止日志文件过大占用磁盘，确保系统稳定。

```
# /etc/logrotate.d/nginx
/var/log/nginx/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 www-data adm
    sharedscripts
    postrotate
        [ ! -f /var/run/nginx.pid ] || kill -USR1 `cat /var/run/nginx.pid`
    endscript
}
```

监控检查

使用 stub_status 模块实时监控并发数与服务状态。

```
location /nginx_status {
    stub_status on;
    access_log off;
    allow 127.0.0.1;
    deny all;
}

# 访问示例：curl http://127.0.0.1/nginx_status
```

性能调优

调整 worker_connections 与缓冲区大小，优化并发处理。

```
worker_processes auto;

events {
    worker_connections 1024; # 适当增加
    use epoll;
    multi_accept on;
}

http {
    client_body_buffer_size 16k;
    client_header_buffer_size 1k;
    large_client_header_buffers 4 8k;
}
```

安全加固

隐藏版本号、限制并发请求，筑牢安全防线。

```
http {
    server_tokens off; # 隐藏版本号
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    server {
        location / {
            limit_conn addr 10; # 限制单IP并发连接数
        }
    }

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
}
```

能力扩展 - 常用模块

无限可能：利用模块化设计扩展功能。

内置模块

Gzip（压缩）

通过压缩响应数据减少传输量，提升加载速度。

```
gzip on;
gzip_types text/plain application/json;
```

Auth_Basic（认证）

提供基本的HTTP身份验证，保护敏感资源。

```
auth_basic "Restricted Area";
auth_basic_user_file /etc/nginx/.htpasswd;
```

Rewrite（URL重写）

灵活修改请求URI，实现URL重定向和伪静态。

```
rewrite ^/old-url$ /new-url permanent;
```

扩展模块

Lua（高性能脚本化）

嵌入Lua脚本，实现复杂的业务逻辑和动态处理。

```
content_by_lua_block {
  ngx.say("Hello, Lua!")
}
```

GeoIP（基于地理位置限流）

根据客户端IP的地理位置信息进行访问控制或限流。

```
geoip_country /usr/share/GeoIP/GeoIP.dat;
if ($geoip_country_code = CN) { return 403;
}
```

ModSecurity（WAF防护）

集成开源Web应用防火墙，提供全面的安全防护。

```
modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/main.conf;
```

CORE POINTS

- 理论与实践结合，通过代码示例增强可理解性
- 结构化呈现技术参数，确保信息自明性
- 强调生产环境的最佳实践与安全性

第九页：前瞻视野 - 未来发展

云原生与现代网络协议的引领者。

HTTP/3（QUIC）支持：

更快的传输速度。

Kubernetes Ingress 控制器：

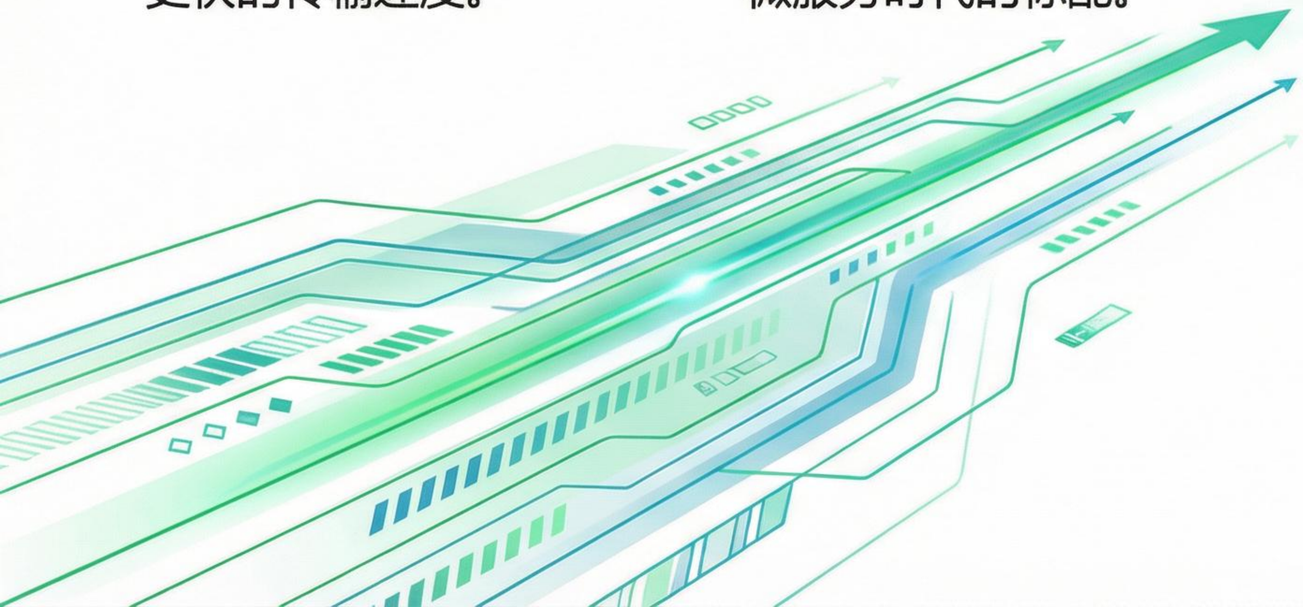
微服务时代的标配。

Nginx Unit：

动态应用服务器架构。

持续演进：

向更轻量化、更智能化的流量网关发展。



CORE POINTS

- 理论与实践结合，通过代码示例增强可理解性
- 结构化呈现技术参数，确保信息自明性
- 强调生产环境的最佳实践与安全性

第十页：课程总结 - 知识回顾

掌握 Nginx，掌控高性能 Web 服务的核心。