ThoughtWorks®

Xian Secure Day

微服务安全思考与实践

yczhang@thoughtworks.com
ThoughtWorks Security Community
Zhang Yuchen

INTRO

- 微服务简介，更大的灵活性伴随着更复杂的安全挑战
- 微服务安全范围介绍——从人到服务
- 从人员到组织，从安全知识培养到组间合作模式
- 尝试治理基础设施——平台、网络、运行环境等
- 资源的离散式分布所带来的管理难题
- 设计、开发、测试、部署、运维——该如何考虑服务的安全
- 微服务的认证与授权
- 强大的开发工具一定是安全的吗?
- Q & A

ThoughtWorks®

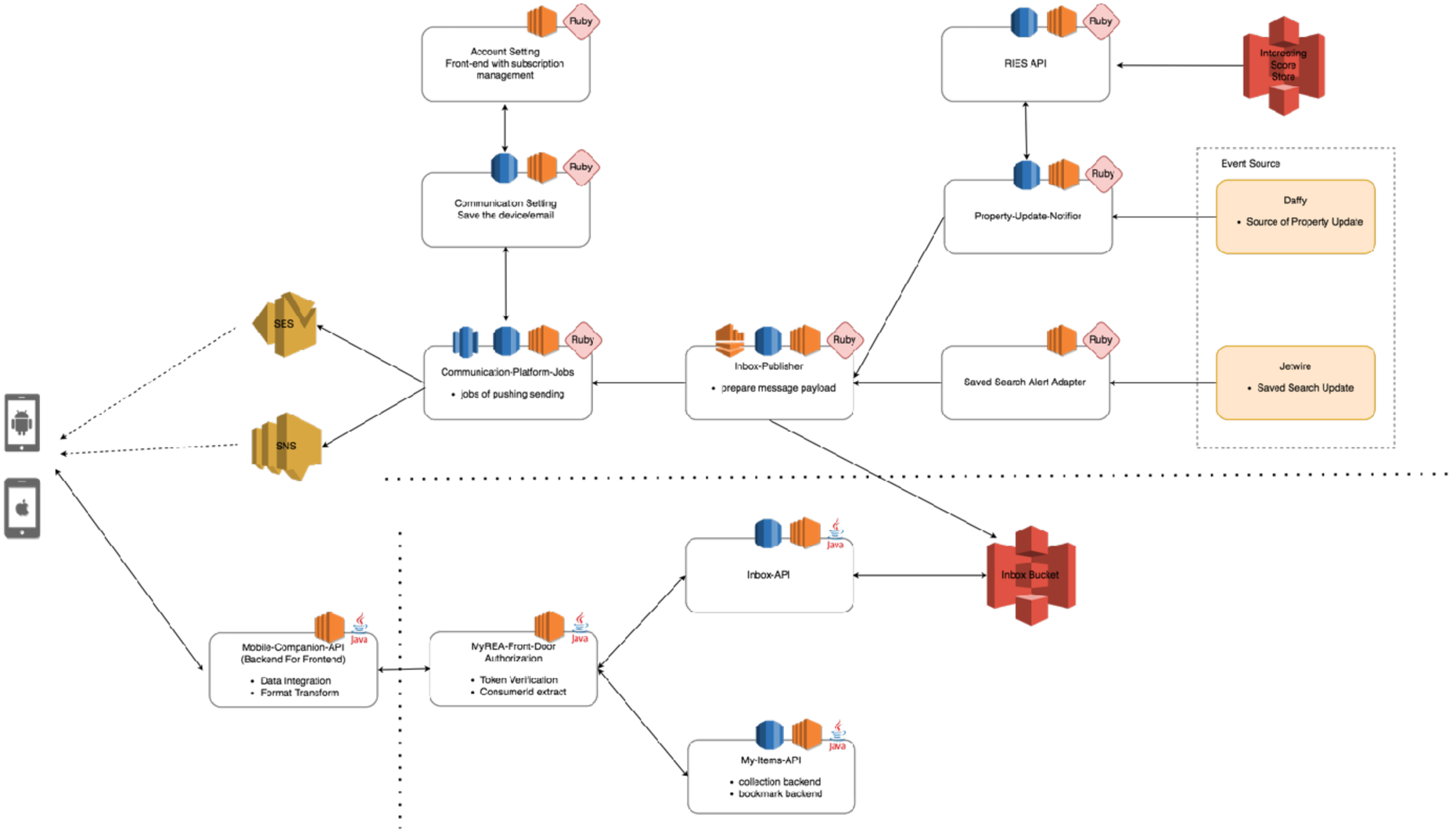
微服务简介，灵活性与治理难题

微服务带来了更小的开发团队，更快的迭代，更少的依赖，更低的交付风险

谁该为服务的安全负责？Dev or Ops？又有什么办法能确保上百个互相调用的服务是安全的呢?

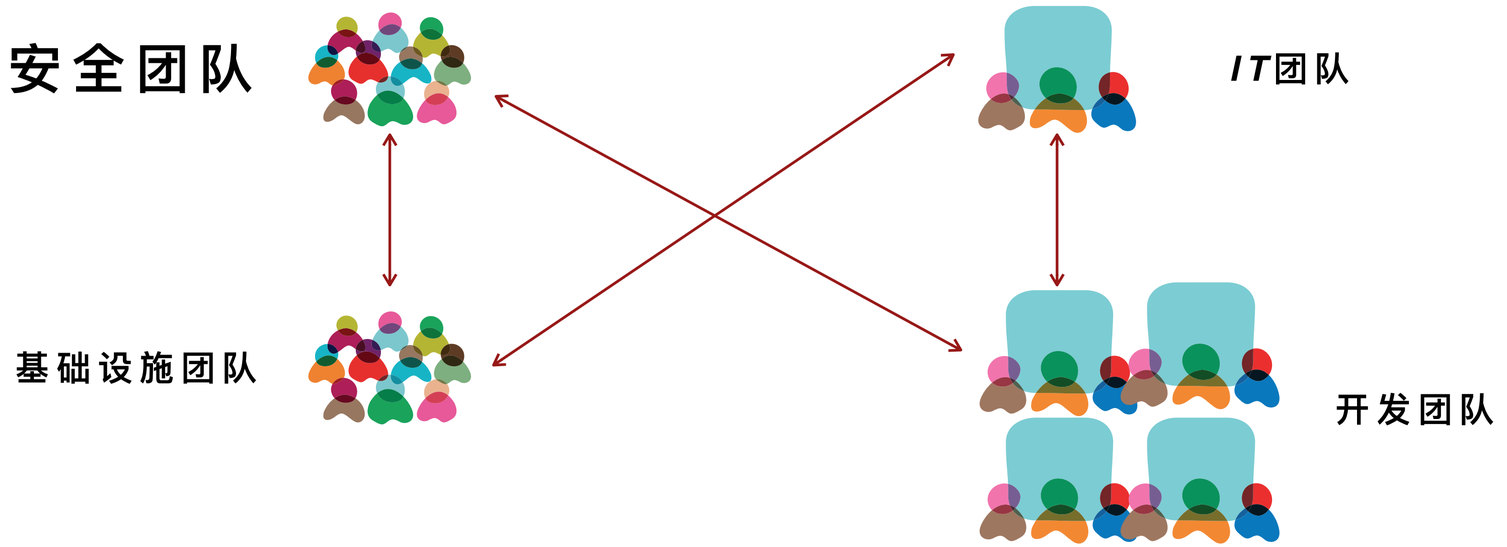
于是我们开始思考……

- 如何保证你的内部服务不被外界访问?
- 如何确保核心网络不被入侵?
- 服务之间的通信会被窃听吗?
- 服务之间的授信该怎么处理?
- 中间件一定是安全的吗?
- 我们在哪里做用户的身份验证?哪里去做授权验证?
- 如果攻击者拿下了一个服务，他可以十分简单的调用下游的服务吗?
- 我们怎么知道请求是没有被篡改的呢?
- 数据库密码，管理员身份，API Key，秘钥等等该怎么办?
- 你现在有多少个服务，多少个容器，多少个数据库，多少个队列?
- 一个小组只有五个人，怎样才能把安全做到位呢?
- 你能说出最新版 OWASP Top 10 的名字吗?
- ……


Ruby
Account Setting
Front-end with subscription
management
Ruby
Communication Setting
Save the device/email
Ruby
Communication-Platform-Jobs
• jobs of pushing sending
SES
SNS
Ruby
Inbox-Publisher
• prepare message payload
Ruby
RIES API
Interesting
Score
Store
Ruby
Property-Update-Notifier
Ruby
Saved Search Alert Adapter
Event Source
Daffy
• Source of Property Update
Jewire
• Saved Search Update
Java
Inbox-API
Inbox Bucket
Java
Mobile-Companion-API
(Backend For Frontend)
• Data Integration
• Format Transform
Java
MyREA-Front-Door
Authorization
• Token Verification
• Consumerid extract
Java
My-Items-API
• collection backend
• bookmark backend

尝试划定微服务安全的范围

安全咨询团队



方法论　最佳实践　教育培训　信息更新

安全评估　威胁漏洞　身份安全　隐私保护

安全评估与安全教育

Control Description
1.1.1 The team understands client security requirements (MSA/SOW/etc.)
1.1.2 The team adheres to client's security requirements
1.2.1 有IRP（事件响应流程）文档，包含事件通知流程（如根据合同/MSA要求，在发生数据泄露等安全事件时及时通知客户）等内容
2.1.1 使用密码管理工具管理共享密码等机密信息
2.1.2 与项目相关的Github账号开启二次认证
2.2.1 正确分配账号权限，应基于least privilege和separation of duties原则（例如DEV、QA、BA权限不同）；无未经许可的账号共享
2.2.3 账号及权限的创建、修改、禁用和删除等操作，有相应流程及记录
2.2.4 至少每半年审计重要账号和访问权限（如production权限账号），确保没有遗留账号、权限累积等情况
3.1.1 成员离开项目组时，应删除一切与项目相关的敏感信息（如客户资料、账号密码记录、源代码等）
3.2.1 敏感数据加密存储和传输（infra/app/CI, etc.）
3.3.1 关键数据应定期备份（在线/离线，防范勒索软件攻击）
3.4.1 如有可能，采用技术手段防止或发现敏感数据泄露（如Google Alert、Gitrob等定期的GitHub扫描）

案例

可能会造成的损失

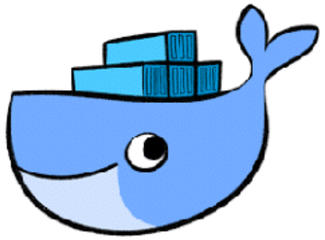
1. 金钱损失
2. 数据丢失
3. 病毒扩散
4. 公司形象受损

ThoughtWorks®

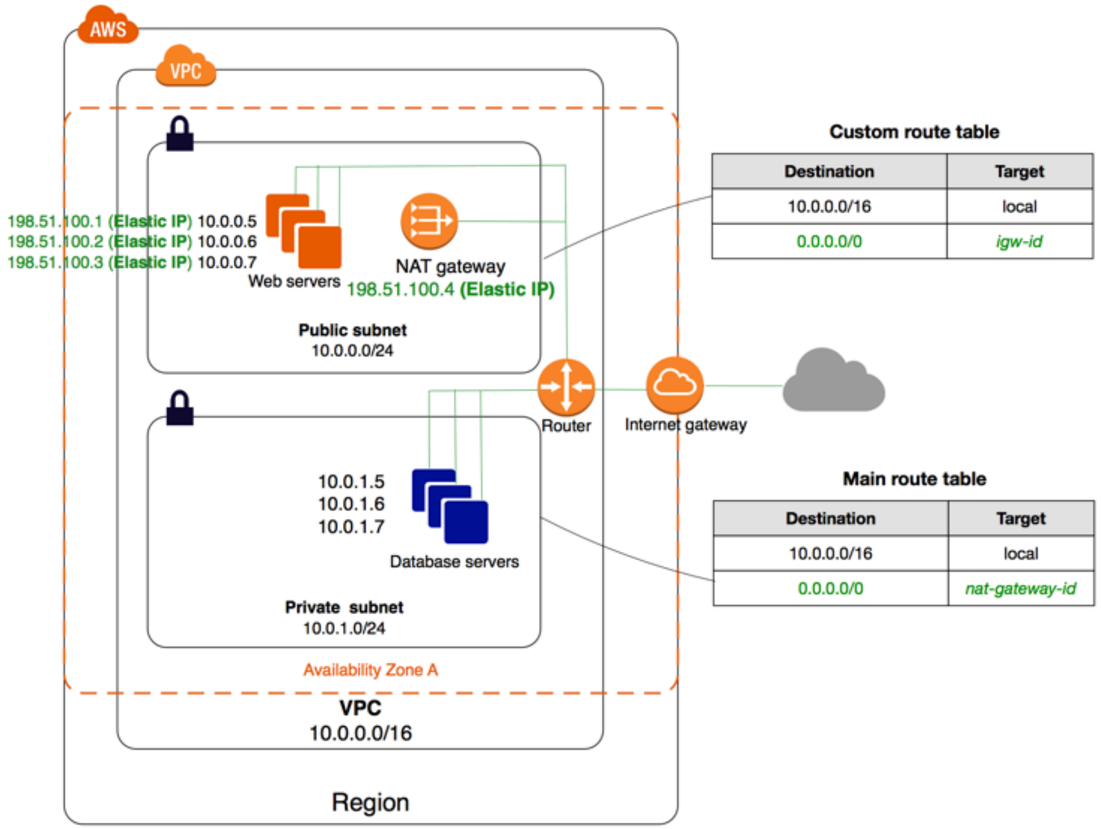
利用强大的基础设施

Philosophy: Automation, Infrastructure as Code, Resource Treating

基础设施方面的思考



基础设施方面的思考 —— NETWORK



- VPN
- Multi AZ
- Cross Region
- Firewall
- CDN
- ACL
- Gateway
- Intrusion-detection

基础设施方面的思考 —— MORE

对等环境

运行环境

灾备

Environment

授权管理

最小权限

身份集成

Access Management

Tool

CI - Registry

Monitoring

Codebase

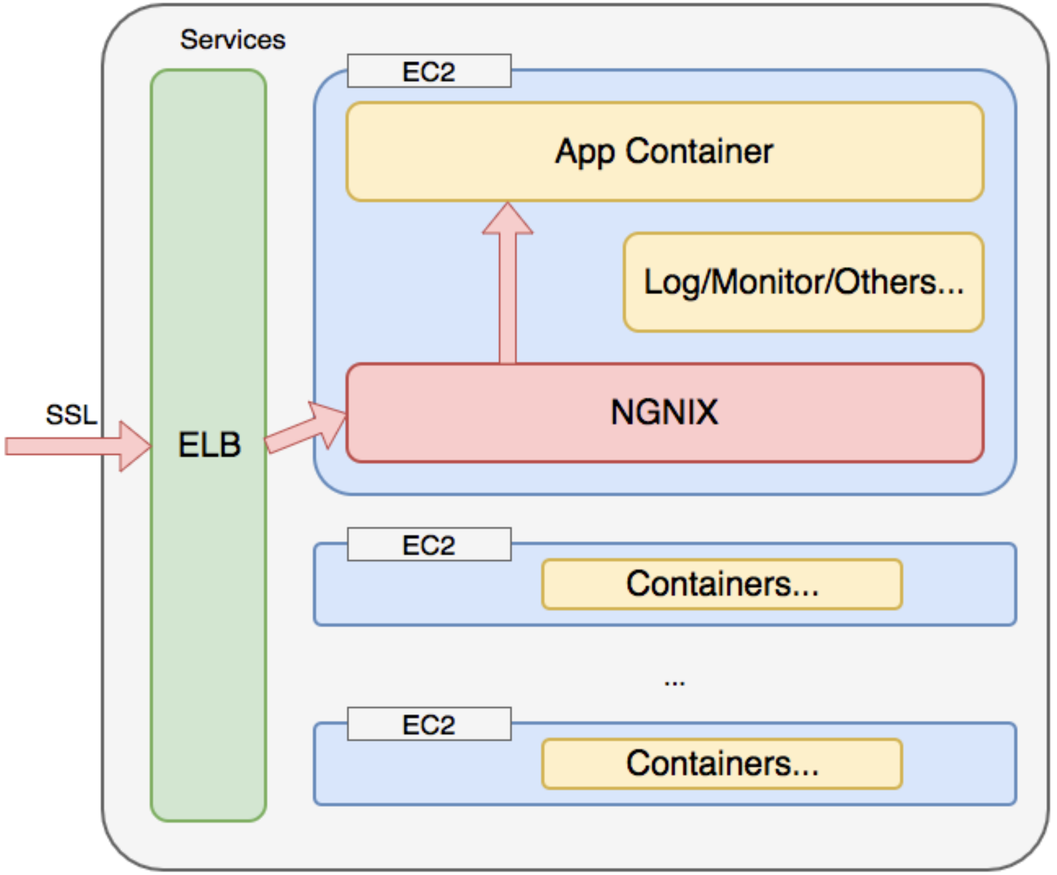
Resource Treating

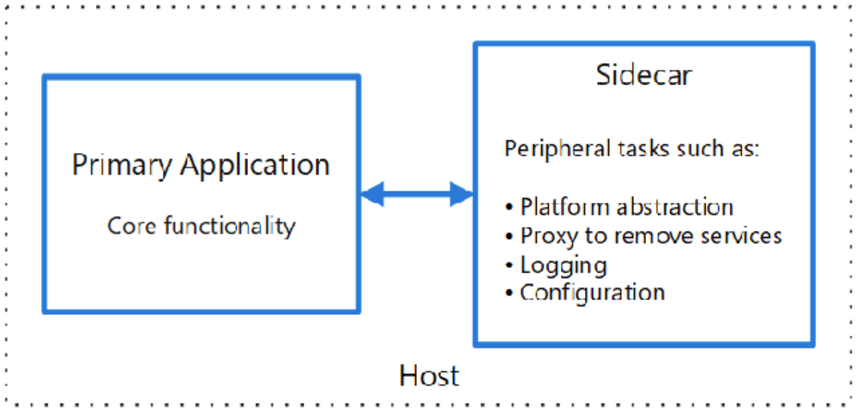
As Code

资源调配

应用部署

基础设施方面的思考 —— SIDECAR





- One sidecar for all services
- SSL, CORS, Slow Connection
- Token Validation
- Request Logging
- Monitoring

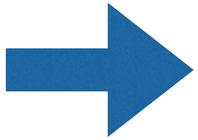
ThoughtWorks®

应用程序与资源

Resources are backing services, services are artifacts.

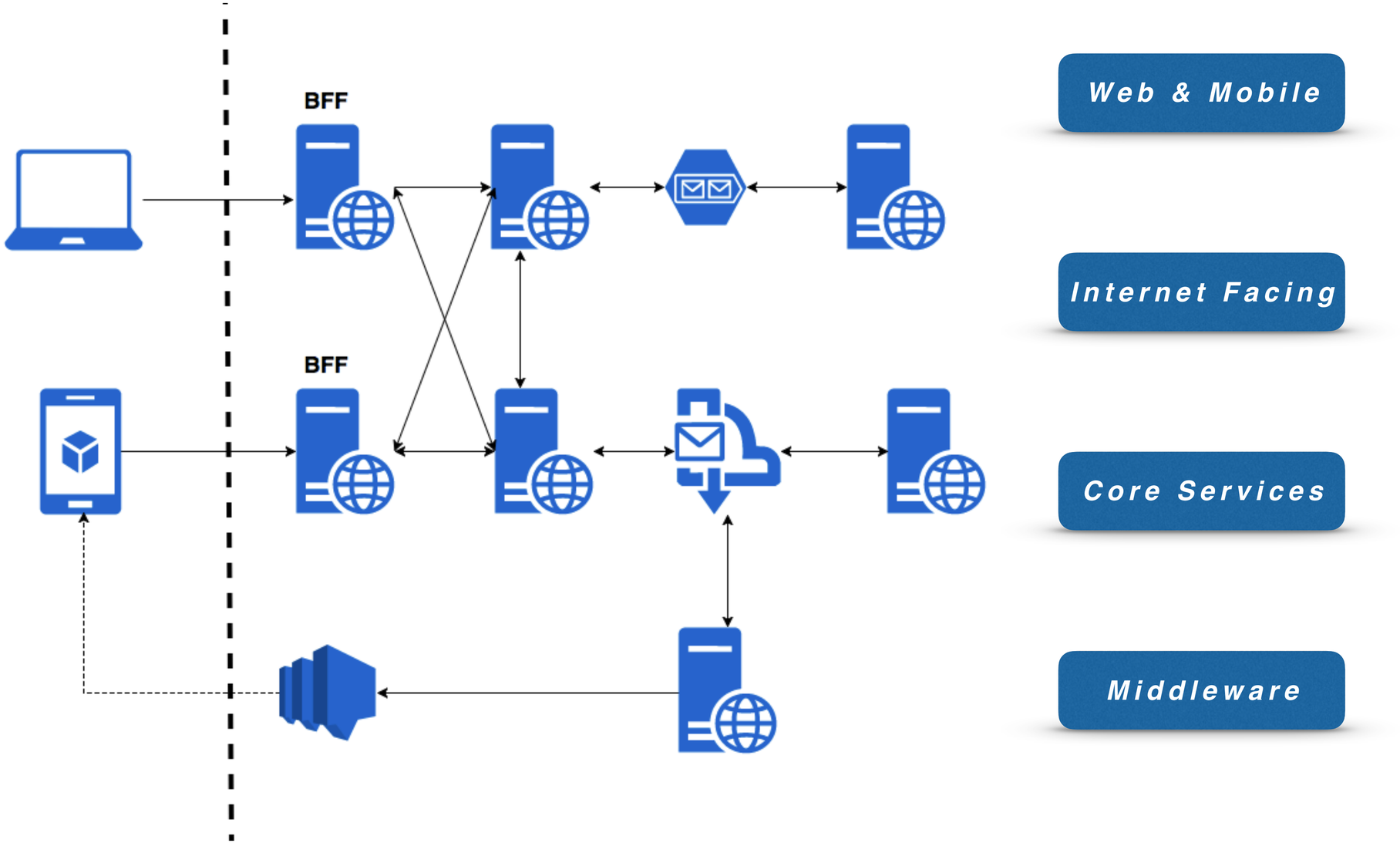
服务所依赖的资源

- 资源也是一种服务
- 三方服务也是一种资源
- 存储服务需要更强的保护策略
- 资源应应用最小权限原则



- root user
- encryption
- snapshot
- breach protection
- access management
- sensitive data
- data aggregation
- …

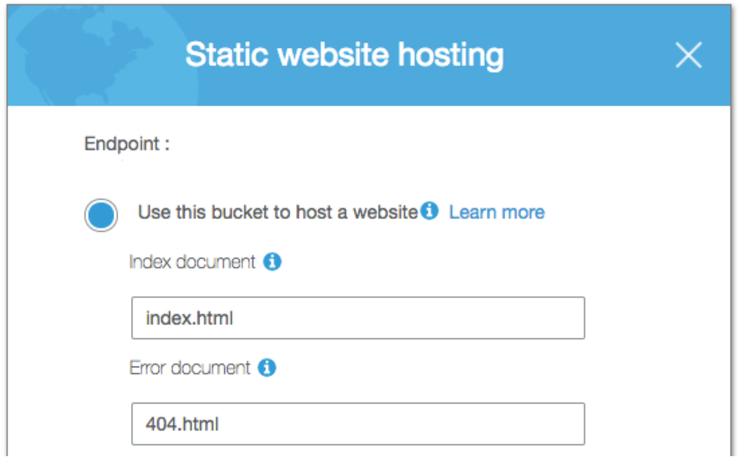
服务 – FUNCTIONALITY



服务 – FRONTEND & INTERNET FACING

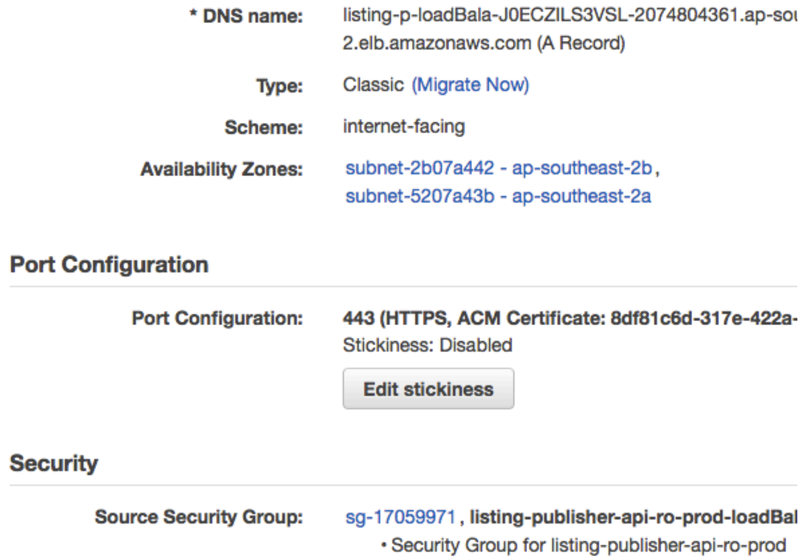
Web & Mobile

- 前后端分离
- 微前端
- 静态内容部署
- SSL & HTTPS
- 前端应用安全



Internet Facing

- SSL & HTTPS
- 认证与授权
- 负载均衡
- BFF
- 服务发现
- 可用性



服务 – CORE SERVICES & MIDDLEWARE

Core Services

- 授权验证
- SSL & HTTPS
- 监控治理
- 服务授信
- 资源访问
- 请求验证

“Microservices architecture, with a large number of services exposing their assets and capabilities through APIs and an increased attack surface, demand a zero trust security architecture — ‘never trust, always verify’. ” —— Tech Radar v17

Middleware

- 数据安全
- 访问限制
- 身份问题

- 攻击者是否有能力通过中间件控制下游系统呢?
- 攻击者是否可以读取到队列中的数据呢?

服务 – IN LIFE-CYCLE

- 设计开发
  - 集成优先：确定服务的边界与资源
  - 扫描工具是一个不错的选择
  - 标准化的认证与授权检查
  - 安全编程指南 & OWSAP TOP 10
  - 运行环境、资源的配置
  - Code Review
  - 日志、警报、ACM 等治理工具
  - 寻求咨询团队的帮助
  - Input & Schema Validation
- 质量保证与维护
  - 自动化的单元测试
  - 特定的安全测试
  - 代码质量检查
  - 回归测试与集成测试
  - 性能测试
  - 预发布环境
  - 灾难恢复演练

服务 – CONFIGURATION

哪些 Config 是非常重要的

- 数据库密码
- 证书
- API Key 等等

静态存放，根据环境读取不同的配置 —— 12 Factor App

- Bind with codebase
- 方便建立对等环境
- 加密，Rotation等

Configuration Service

放在 Configuration Service

- 中心化
- 热修改

ThoughtWorks®

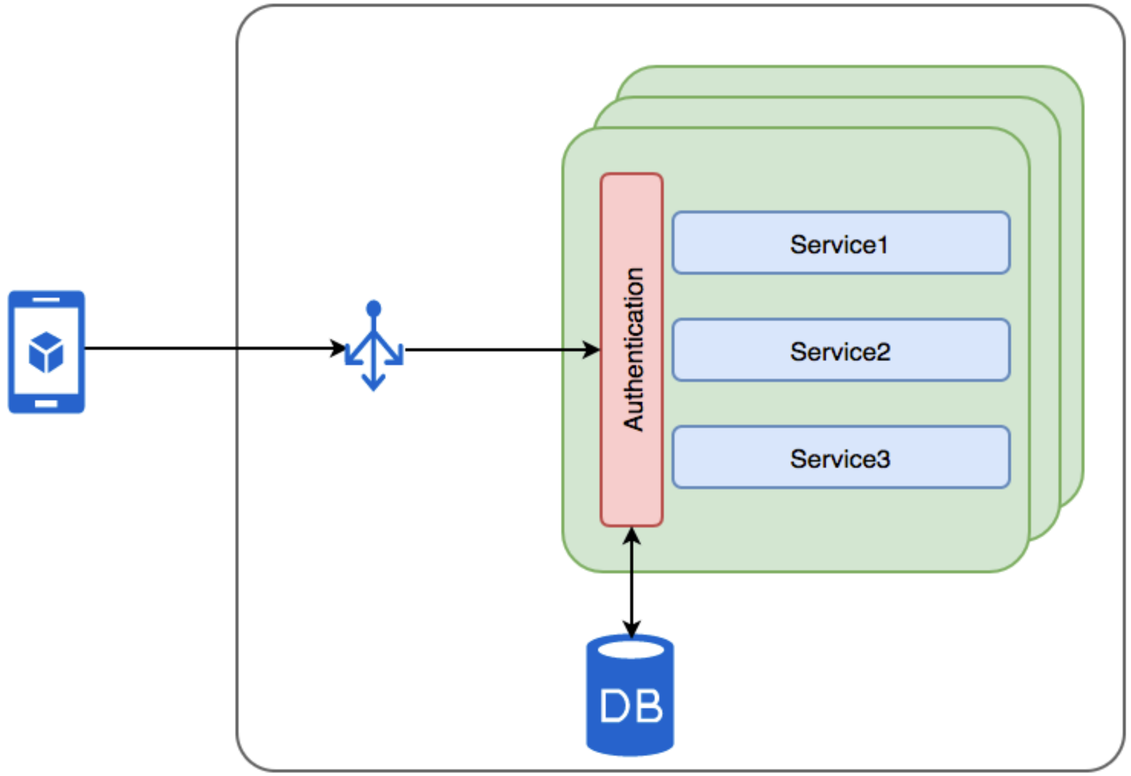
基本的认证与授权

In microserivce, authentication & authorization is a problem

认证与授权 —— MONOLITHIC APP

传统的应用实现认证、授权、验证稍微简单一些

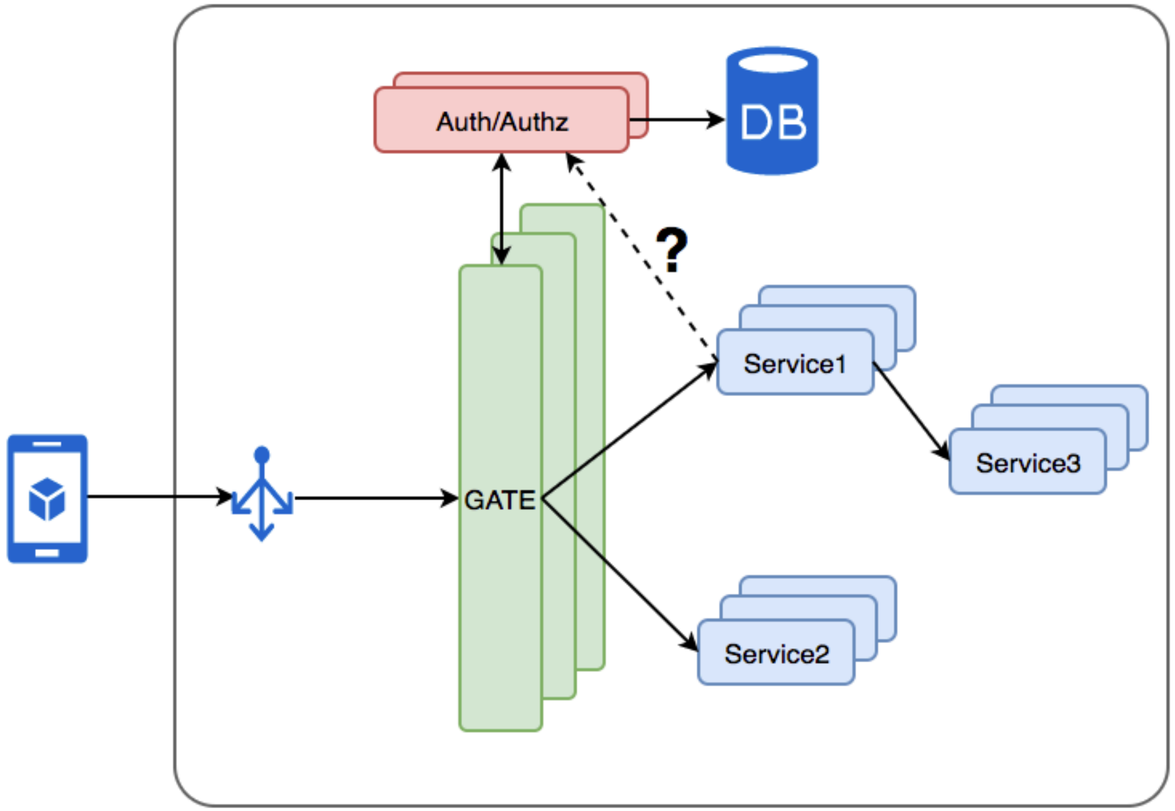
- Session Based 可以很简单的实现
- 方法调用是安全的
- 实现登录也相对简单
- 攻击面也是比较小的



认证与授权 —— USE GATEWAY

目前比较常见的微服务认证授权模式

- 对于客户端来说，不需要太大的改动
- 如何传递身份?
- 中心化的身份方便处理
- 服务之间如何相互信任?
- 内部安全威胁如何应对?



ThoughtWorks®

工具是可靠的吗?

We have a lot of tools to help us, can we trust them?

工具的安全性

自动化是我们的追求，我们使用工具实现自动化，但是往往忽视掉了工具的安全性

- XcodeGhost 事件
- Putty 记录密码事件
- TW 同事发现 CI 漏洞，可以用来挖比特币
- Ruby Gem Vulnerability

我们尝试了并实践了

- 依赖库的安全配置
- 托管 Docker Registry 的身份认证信息
- 使用密码管理工具管理私钥、密码等
- Codebase、PaaS 、日志中心、ACM 等与 SSO 集成
- CI 的安全配置升级
- 使用 KMS 等加密敏感信息

谢谢

Q & A

张羽辰
yczhang@thoughtworks.com

ThoughtWorks®