极客时间

构建微服务云原生应用

架构师杨波

极客时间

扫码试看/订阅

《Spring Boot & Kubernetes 云原生微服务实践》

极客时间

CHAPTER 05

安全框架设计和实践

极客时间

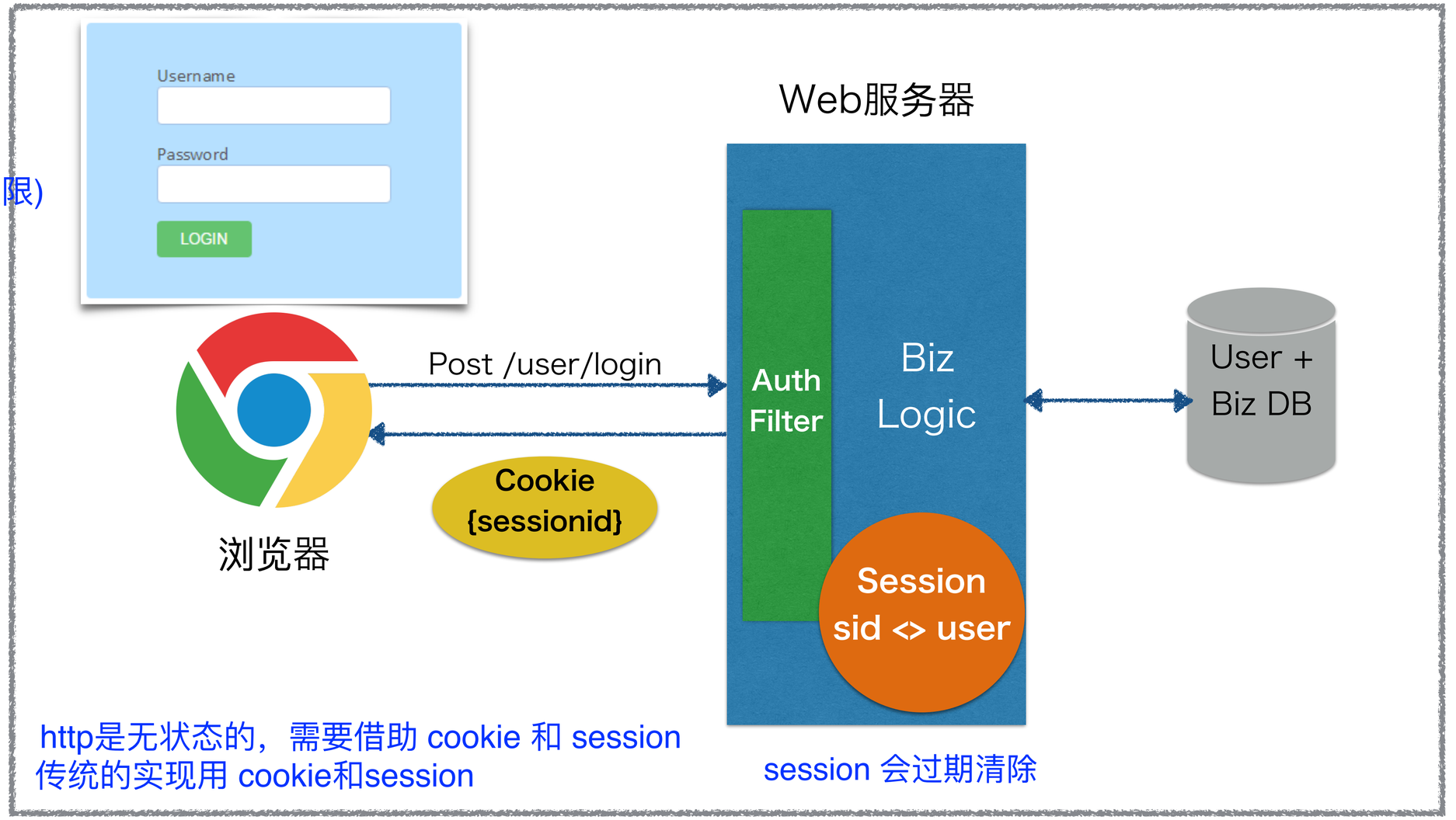
第 1 部分

网站安全认证架构演进 ~ 单块阶段(上)

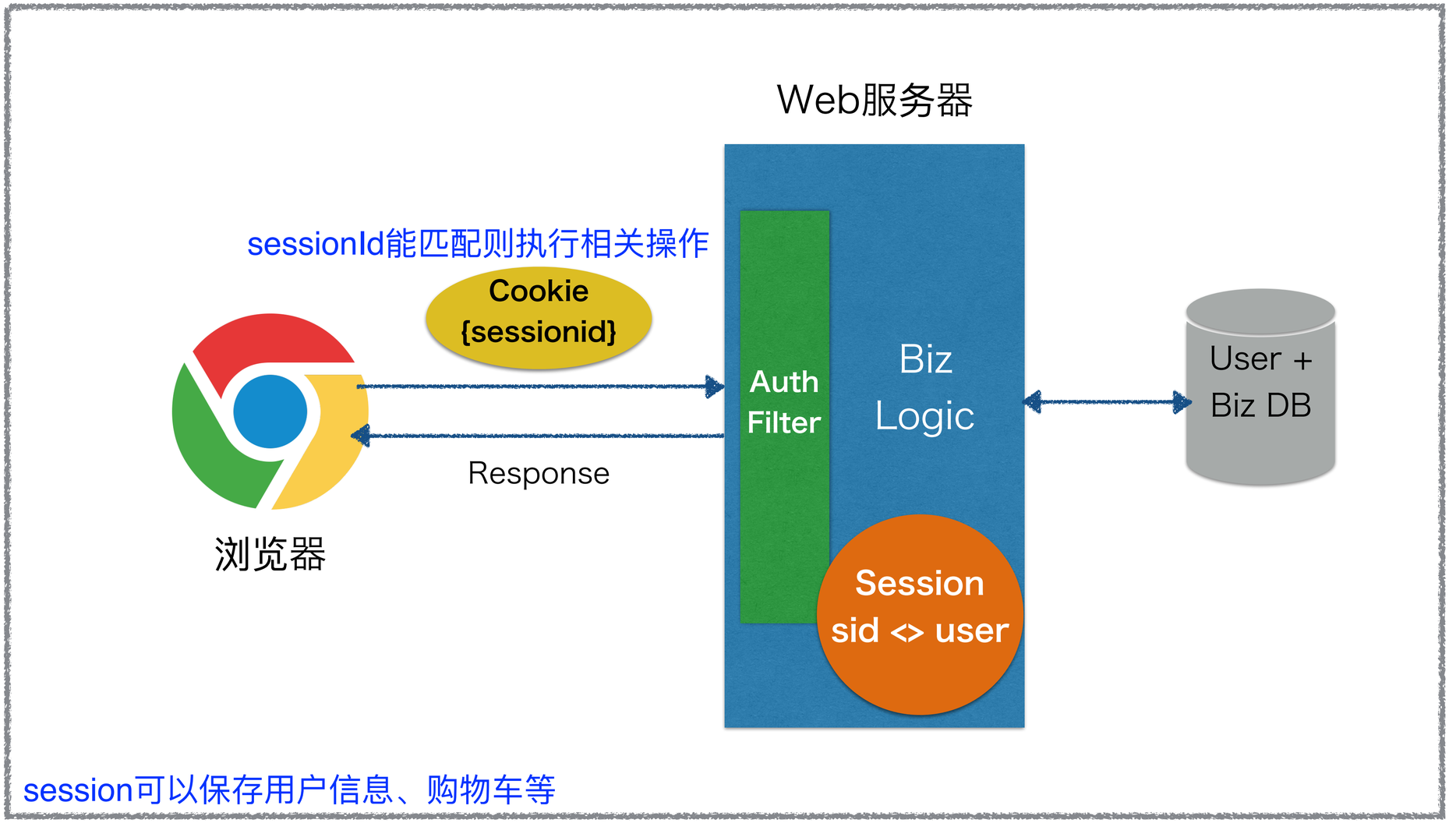
MyStore Auth V1 ~ 认证阶段

认证:你是谁

授权:你能做什么(权限)



MyStore Auth V1 ~ 访问阶段



极客时间

第 2 部分

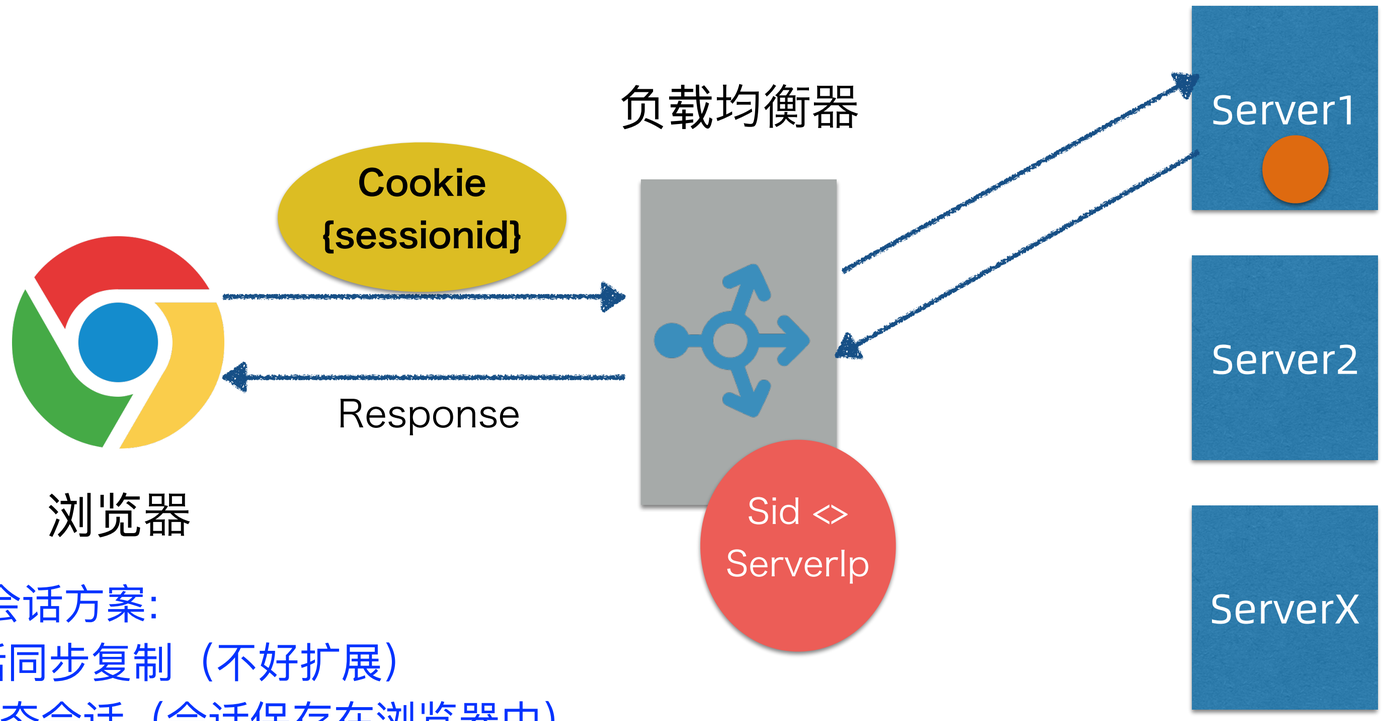
网站安全认证架构演进～单块阶段(下)

Auth V1.1 ~ Sticky Session

黏性会话

解决因为负载均衡原因造成的session尚未过期而退出登录

需要LB支持，记录 sessionId 与 某台服务器绑定



解决黏性会话方案:

方案1.会话同步复制（不好扩展）

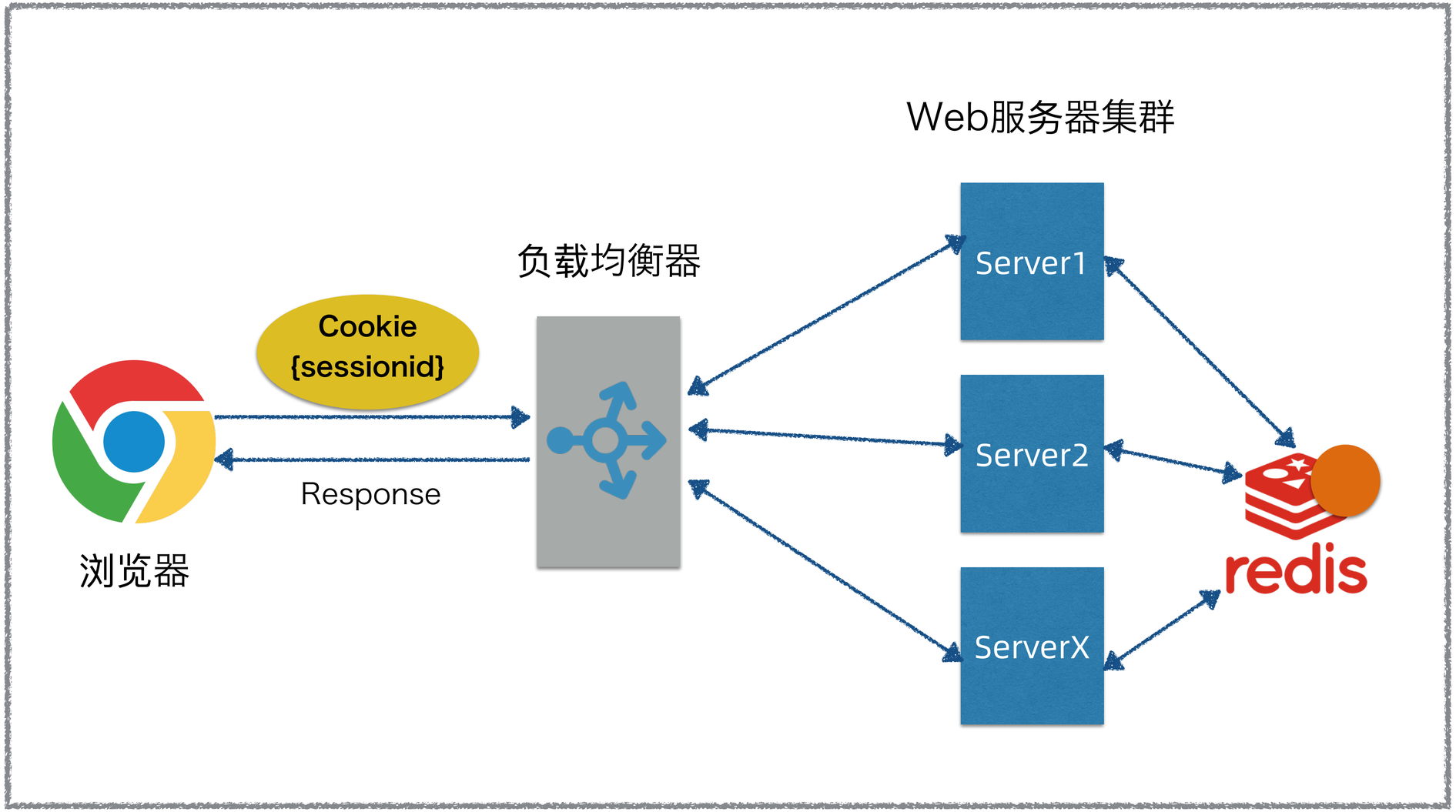
方案2.无状态会话（会话保存在浏览器中）

方案3.集中状态状态会话（如 redis 缓存）

劣势:升级、宕机(稳定性)问题，用户体验差

Auth V1.5 ~ Centralized Session

集中会话



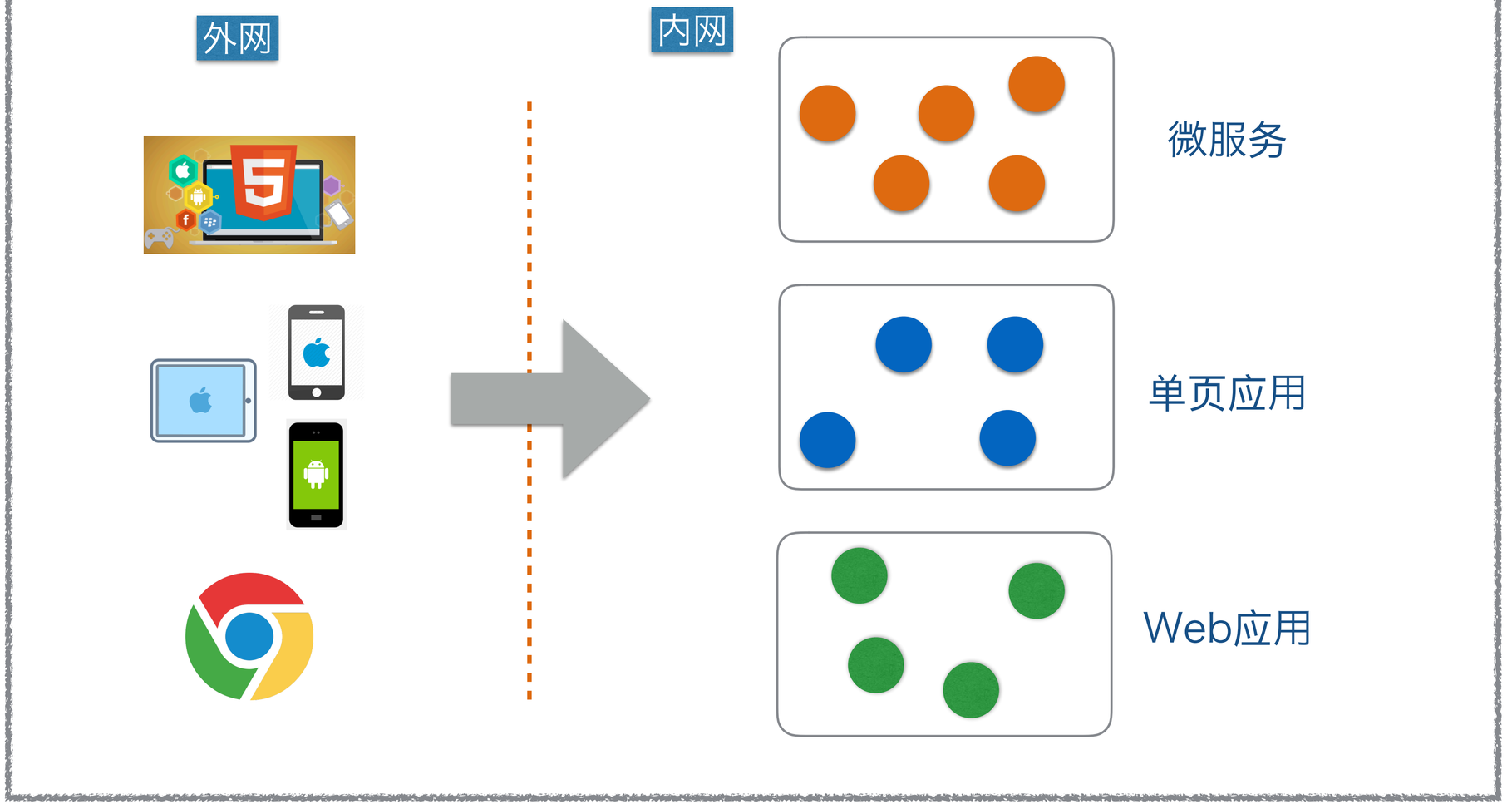
第 3 部分

网站安全认证架构演进～微服务阶段

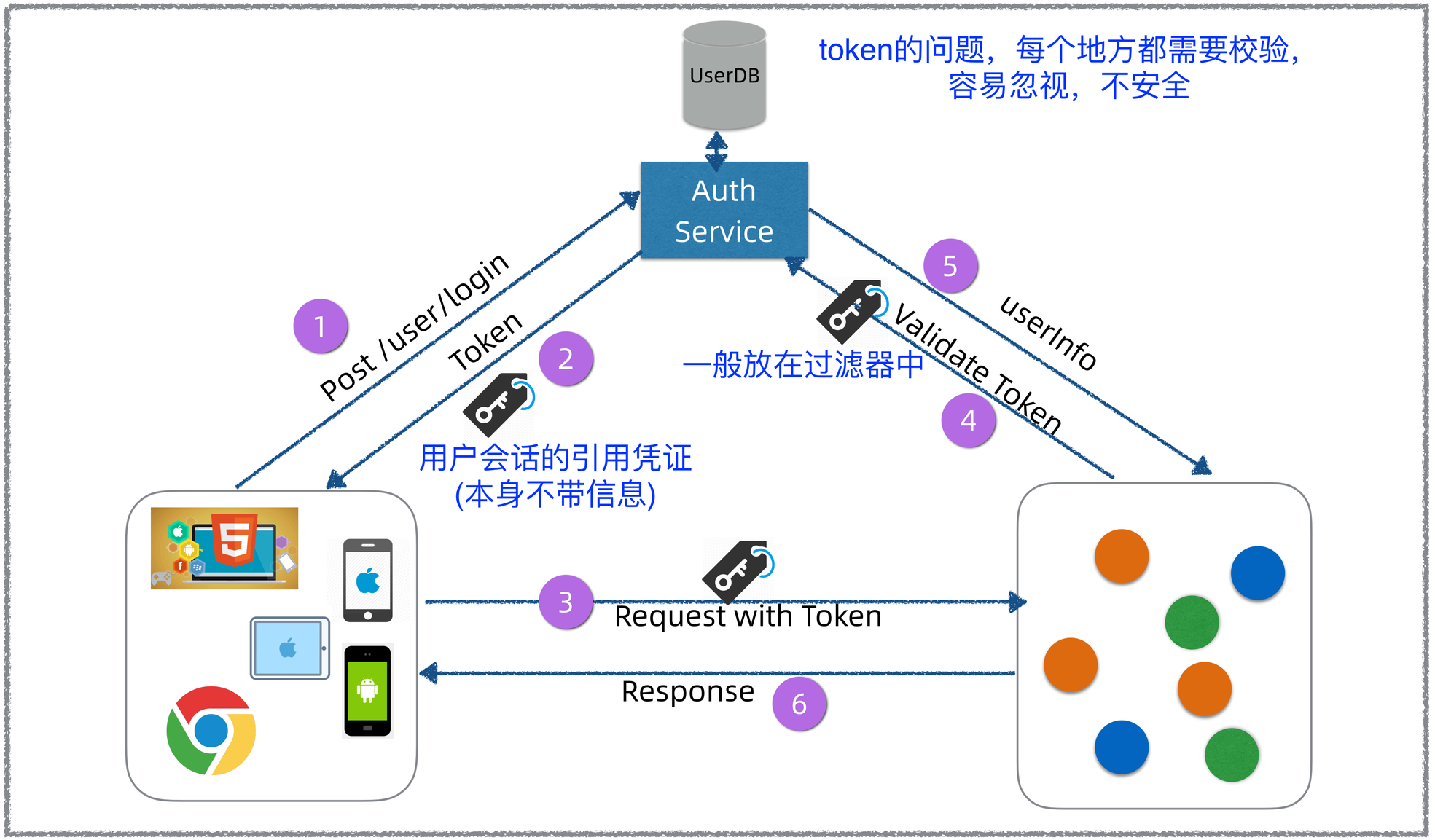
微服务认证授权挑战

对每个服务都要认证

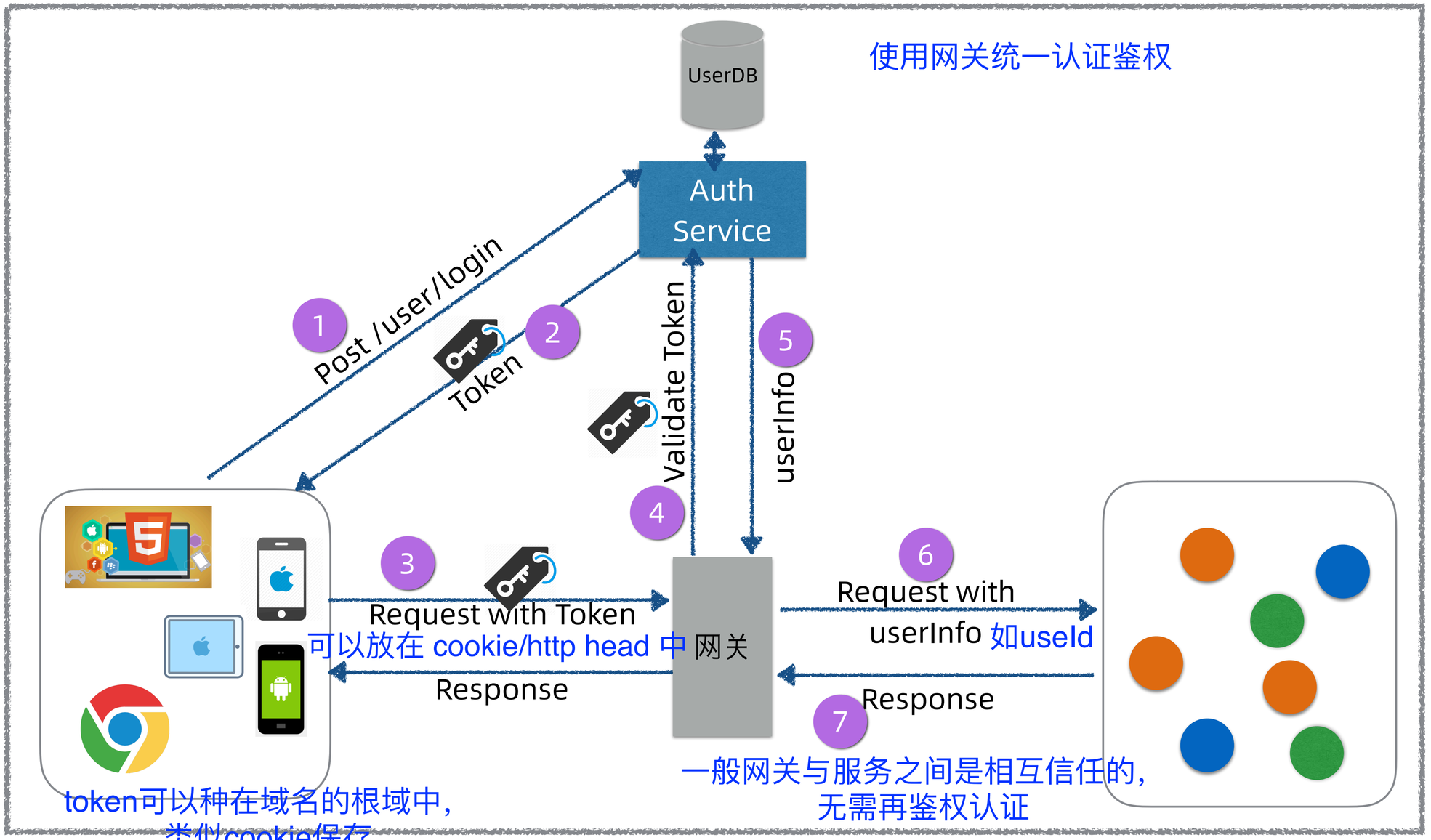
实现SSO（单点登录）



Auth 3.0 ~ Auth Service + Token 令牌(认证的凭证)



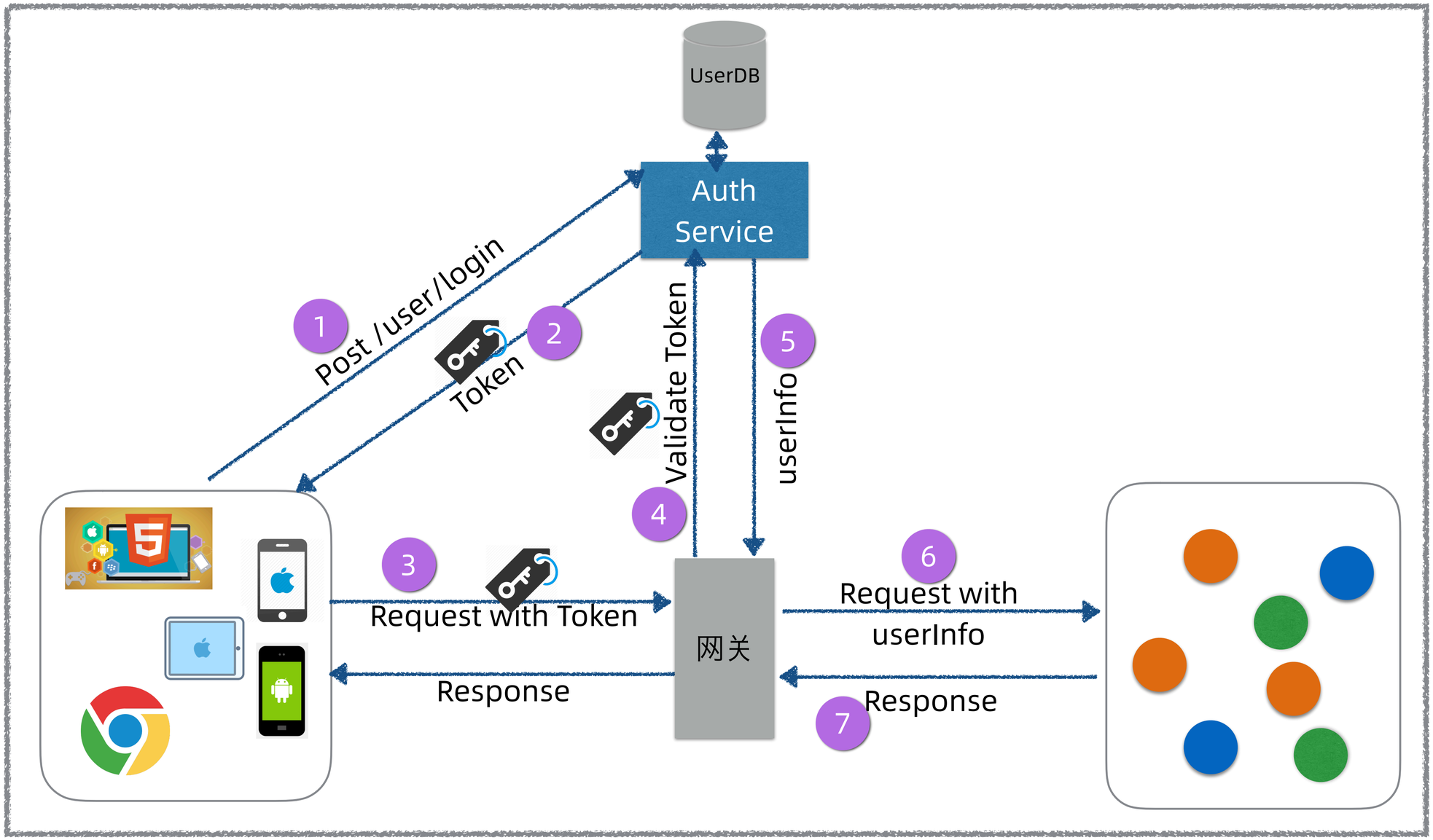
Auth 3.5 ~ Token + Gateway



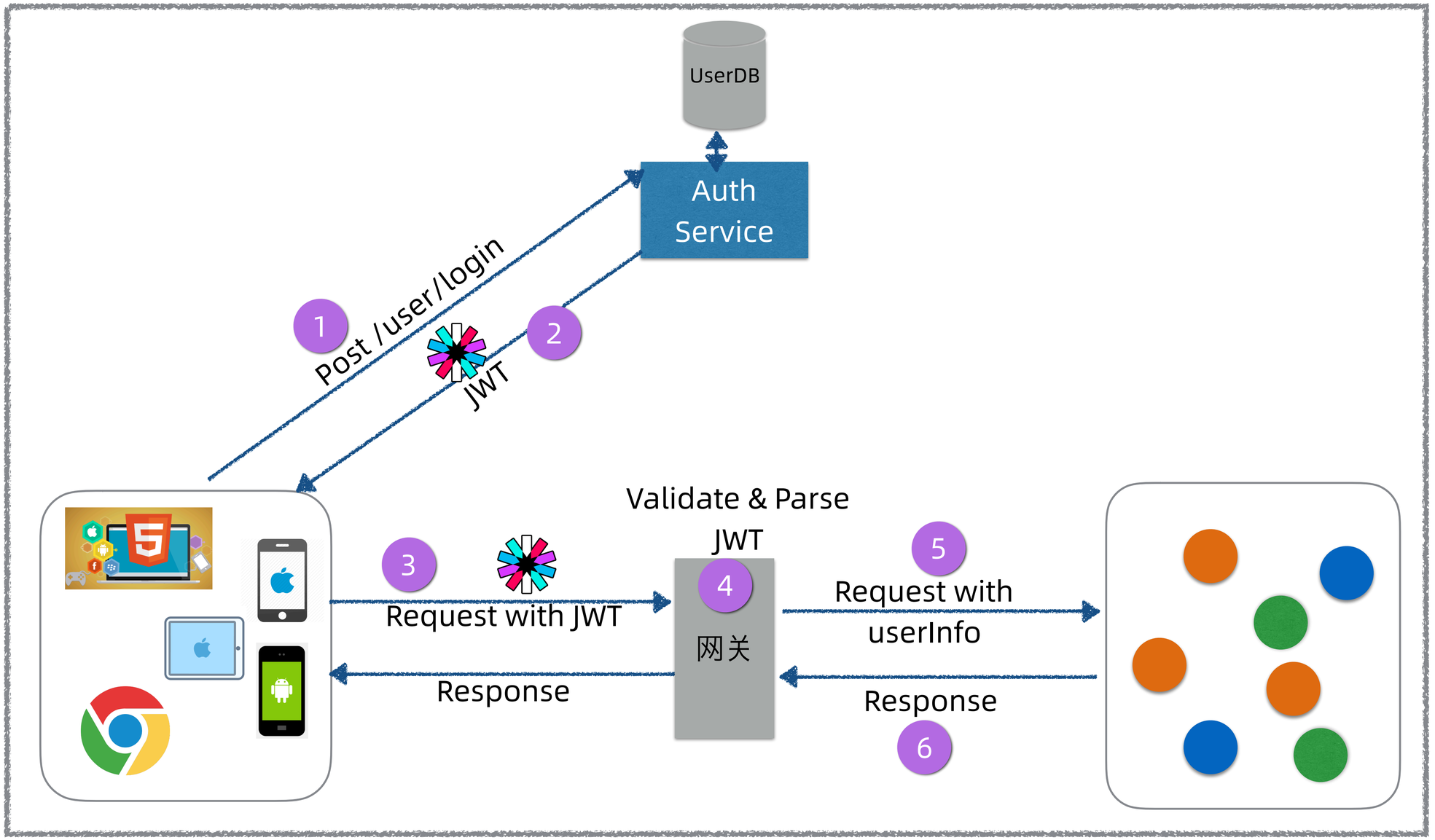
第 4 部分

基于JWT令牌的安全认证架构

Auth 3.5 ~ Token + Gateway



Auth 3.6 ~ JWT + Gateway

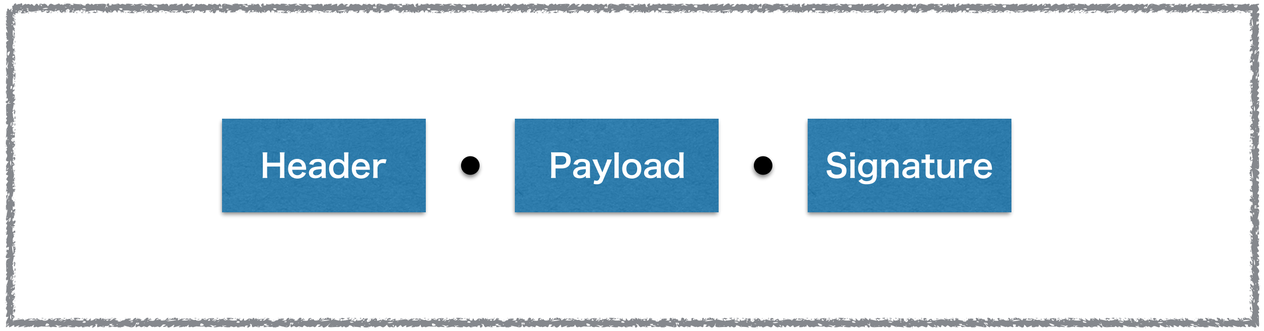


极客时间

第 5 部分

JWT令牌原理

JWT令牌结构



JWT令牌示例

Header

Payload

Signature

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni
J9.eyJpc3MiOiJodHRwczovL215LnNw
cmluZzJnby5jb20iLCJpYXQiOjE0MzU
xNzk2MDMsImV4cCI6MTQzNTE4MTQ
yMSwiYXVkIjoid3d3LnNwcmluZzJnb
y5jb20iLCJzdWIiOiJ3aWxsaWFtQGdtY
WlsLmNvbSIsIlJvbGUiOlsiYXBwcm92
ZXIiLCJ2aWV3ZXIiXX0.pOetxt5vmIyI
wu35-4UJUlR_R24R3Ks46nS1b-xiiME
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

类型和签名方式

PAYLOAD: DATA

```
{
  "iss": "https://my.jskillcloud.com",
  "iat": 1435179603,
  "exp": 1435181421,
  "aud": "www.jskillcloud.com",
  "sub": "bobo@jskillcloud.com",
  "Role": [
    "approver",
    "viewer"
  ]
}
```

颁发的网址
令牌颁发时间
到期时间
目标站点
目标用户

可定制payload

VERIFY SIGNATURE

jwt.io 在线解码

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

签名公式

base64Url(Header) + “.” + base64Url(Payload) + “.” + base64Url(Signature)

基于base64url编码

JWT令牌类比签名支票

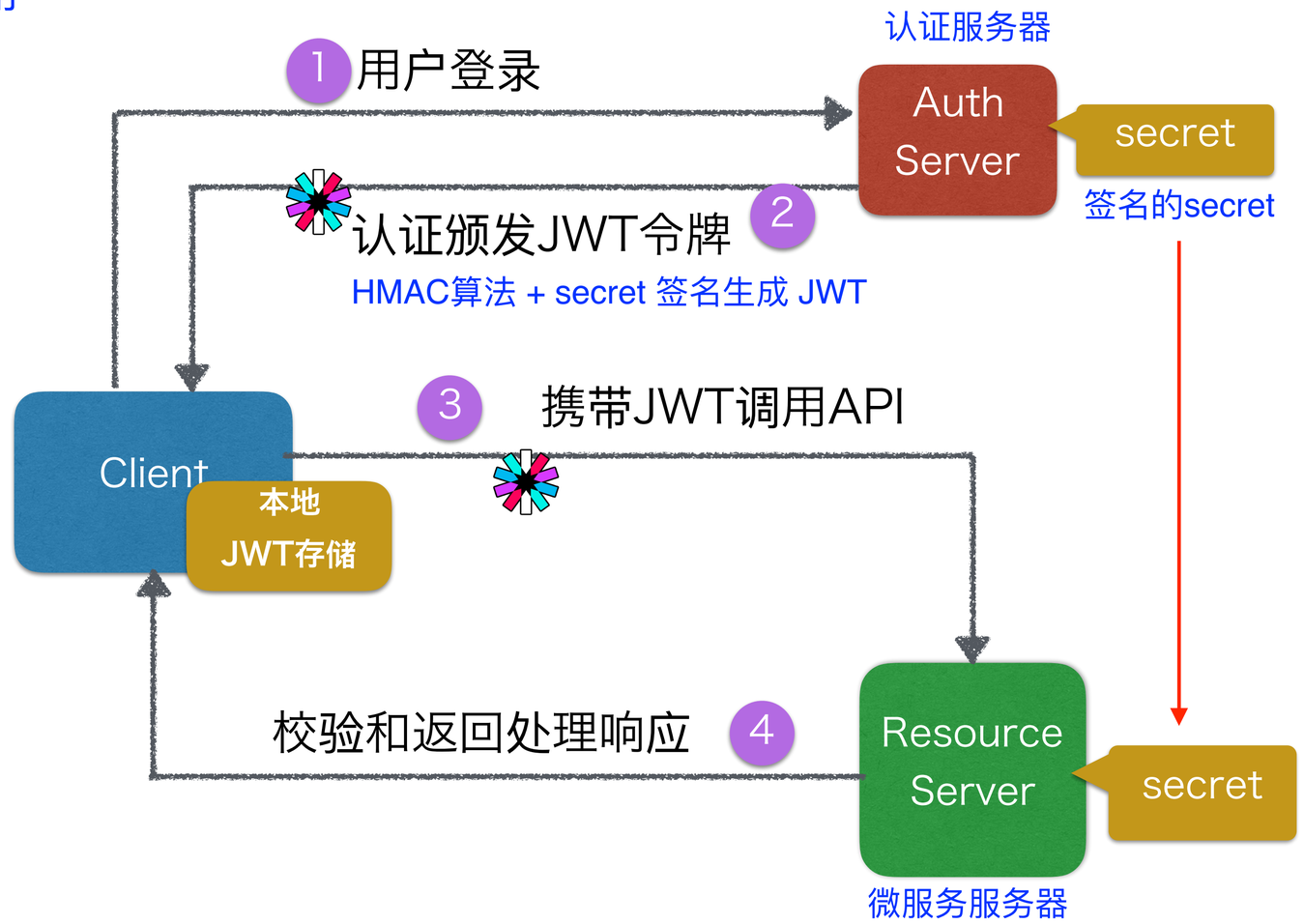
极客时间

第 6 部分

两种 JWT 流程

HMAC 流程

本项目使用

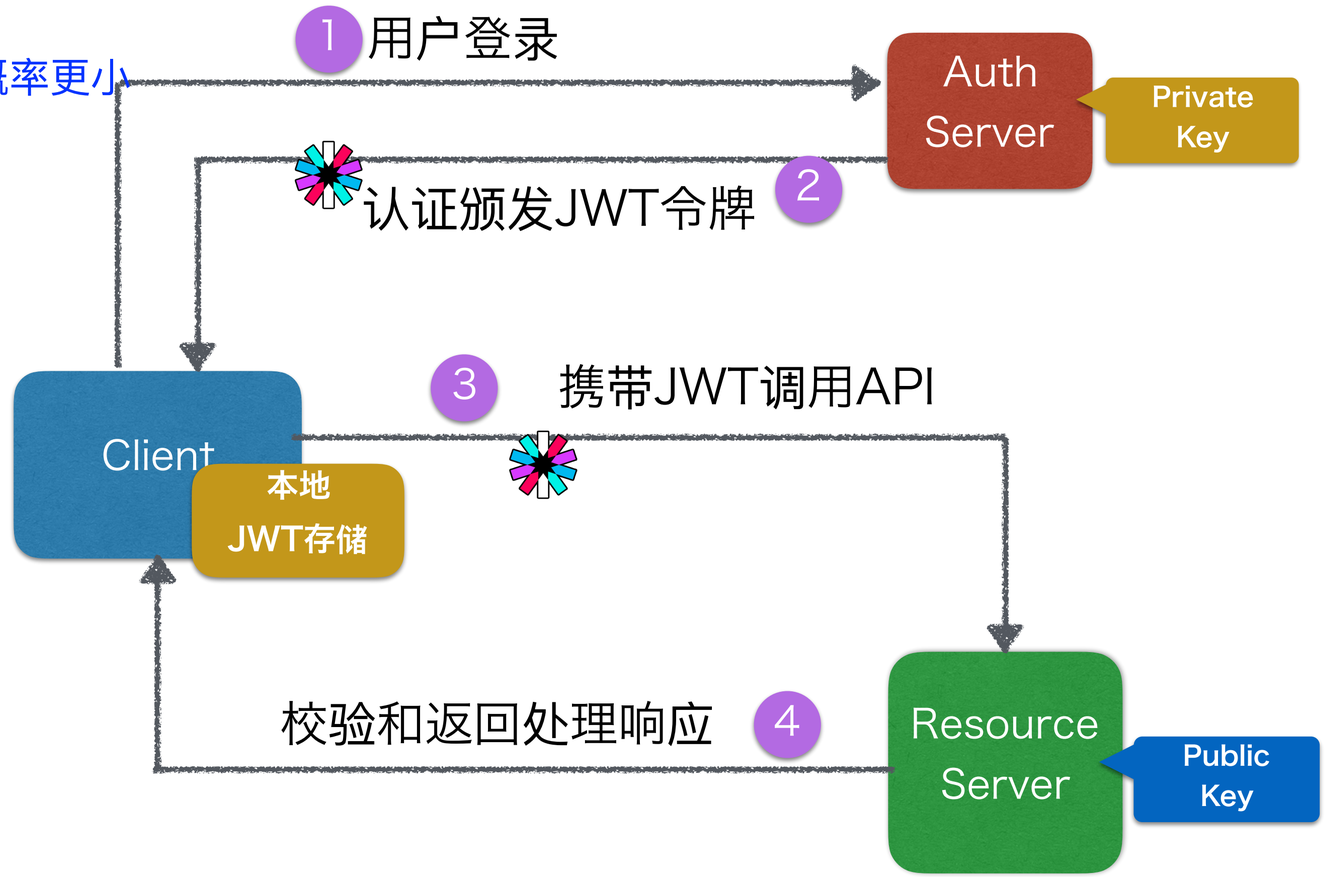


RSA 流程

比 HMAC 更安全

公钥私钥

私钥泄漏概率更小



JWT 优劣

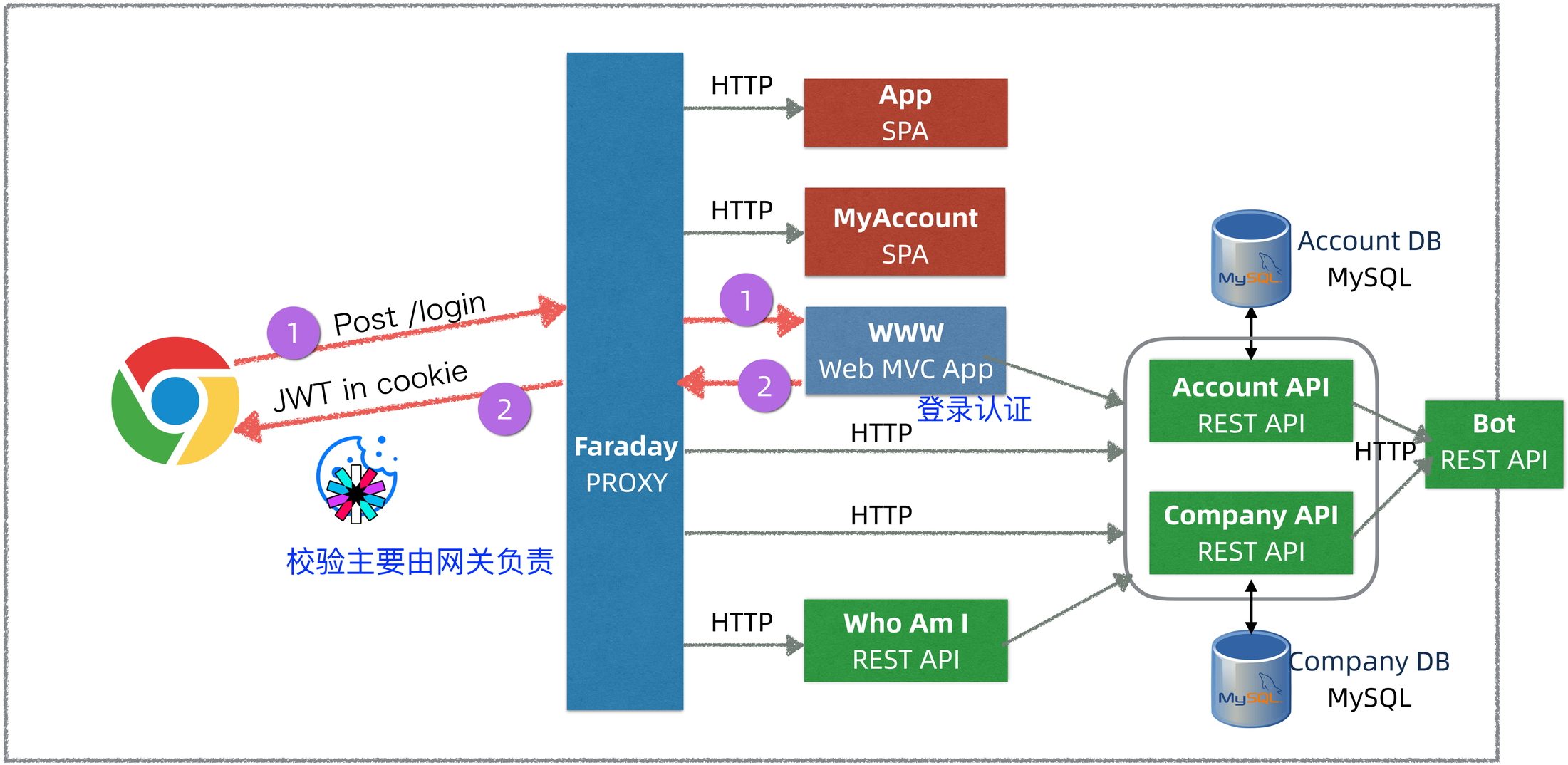
优势	不足
紧凑轻量 对AuthServer压力小 简化AuthServer实现 自校验	更新了信息也不能及时更新 无状态和吊销无法两全 只能重新登录生成再生效 传输开销 会随着声明增多而增大，开销增大

适合安全不敏感的场景

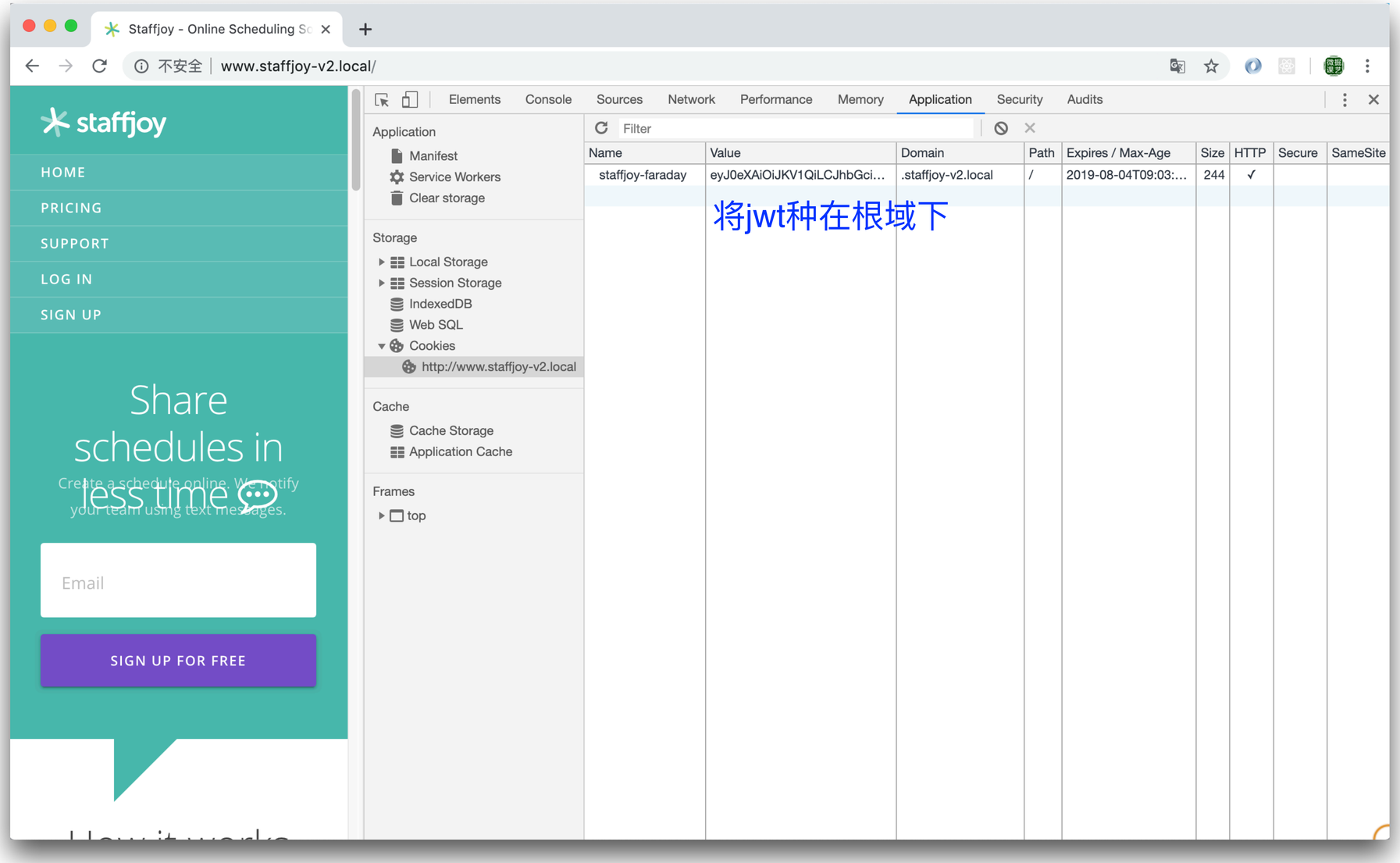
第 7 部分

Staffjoy 安全认证架构和 SSO

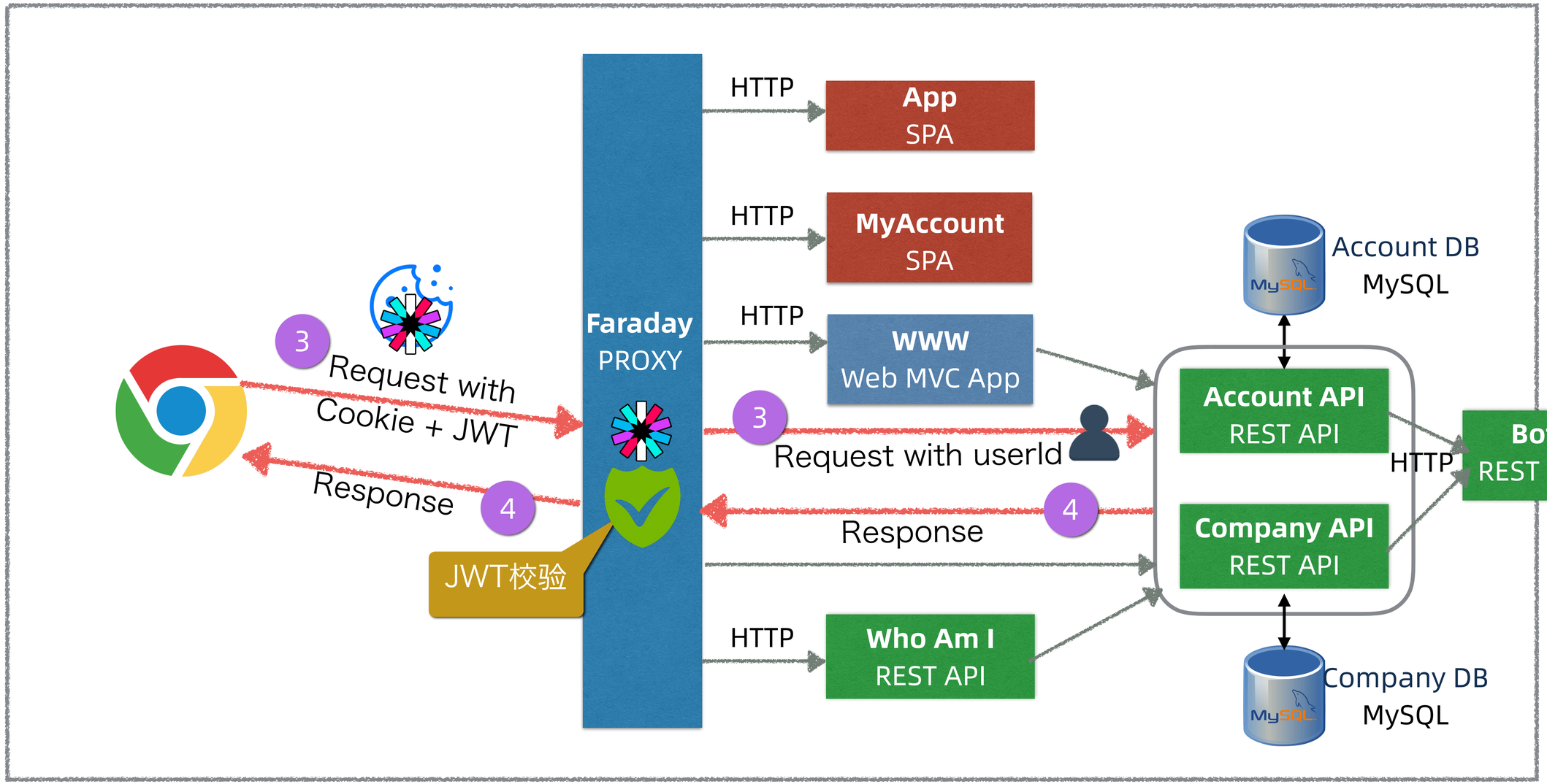
Staffjoy Auth ~ 登录认证阶段

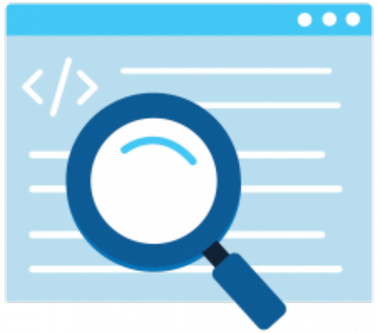


Staffjoy JWT Cookie



Staffjoy Auth ~ 后续访问阶段





第 8 部分

安全认证代码剖析 ~ 用户认证

引入JWT生成和校验库

```xml
<dependency>
    <groupId>com.auth0</groupId>
    <artifactId>java-jwt</artifactId>
    <version>3.6.0</version>
</dependency>
```

JWT生成算法(common/sign)

即 secret

```
private static Map<String, Algorithm> algorithmMap = new HashMap<>();

public static String generateSessionToken(String userId, String signingToken, boolean support, long duration) {
    if (StringUtils.isEmpty(signingToken)) {
        throw new ServiceException("No signing token present");
    }
    Algorithm algorithm = getAlgorithm(signingToken);
    String token = JWT.create()
            .withClaim(CLAIM_USER_ID, userId)
            .withClaim(CLAIM_SUPPORT, support)
            .withExpiresAt(new Date(System.currentTimeMillis() + duration))
            .sign(algorithm);
    return token;
}

private static Algorithm getAlgorithm(String signingToken) {
    Algorithm algorithm = algorithmMap.get(signingToken);
    if (algorithm == null) {
        synchronized (algorithmMap) {
            algorithm = algorithmMap.get(signingToken);
            if (algorithm == null) {
                algorithm = Algorithm.HMAC512(signingToken);
                algorithmMap.put(signingToken, algorithm);
            }
        }
    }
    return algorithm;
}
```

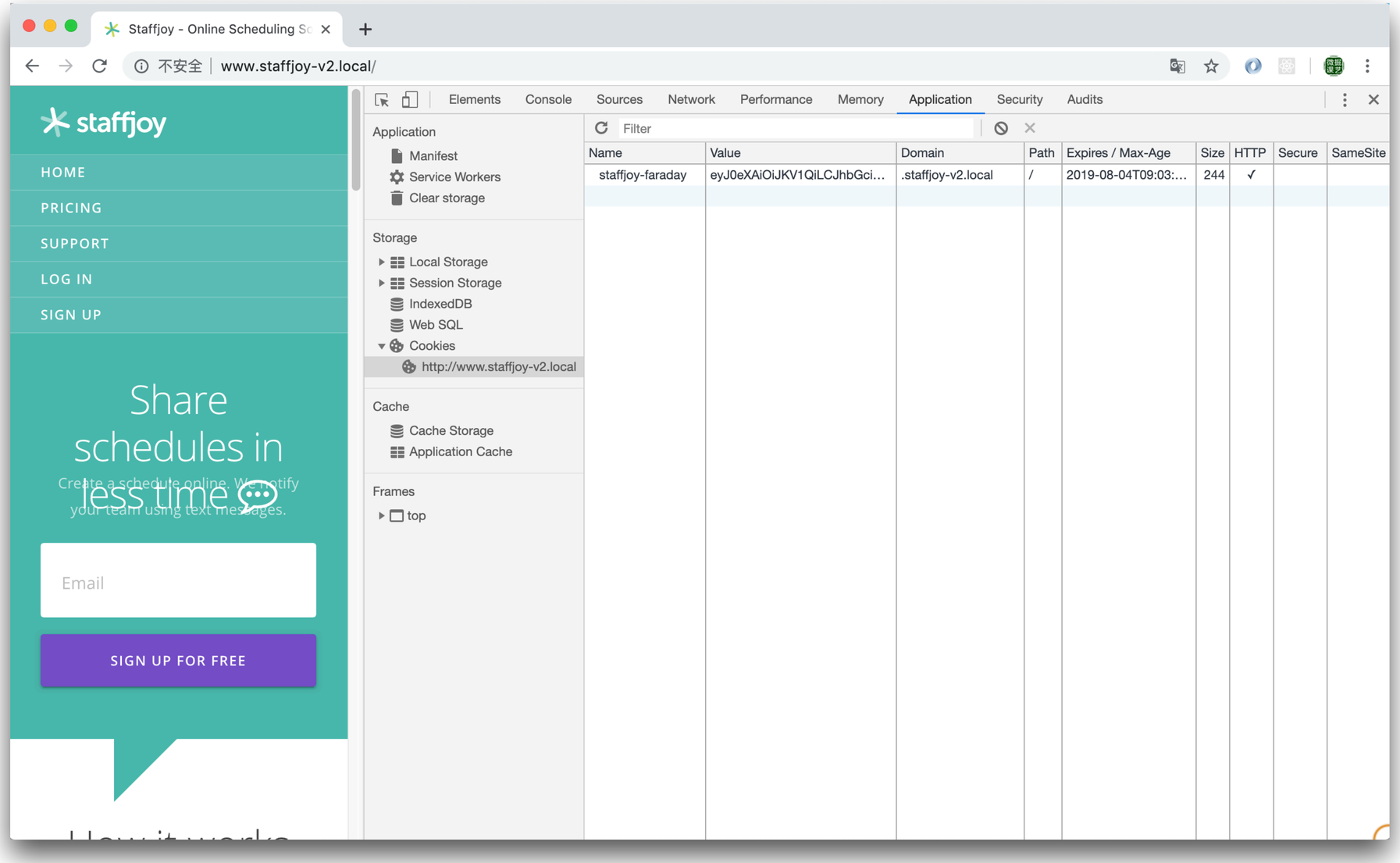
JWT校验算法(common/sign)

```
private static Map<String, JWTVerifier> verifierMap = new HashMap<>();

public static DecodedJWT verifySessionToken(String tokenString, String signingToken) {
    return verifyToken(tokenString, signingToken);
}

static DecodedJWT verifyToken(String tokenString, String signingToken) {
    JWTVerifier verifier = verifierMap.get(signingToken);
    if (verifier == null) {
        synchronized (verifierMap) {
            verifier = verifierMap.get(signingToken);
            if (verifier == null) {
                Algorithm algorithm = Algorithm.HMAC512(signingToken);
                verifier = JWT.require(algorithm).build();
                verifierMap.put(signingToken, verifier);
            }
        }
    }

    DecodedJWT jwt = verifier.verify(tokenString);
    return jwt;
}
```

登录login种Cookie(common/sessions)

```
public static void loginUser(String userId,
                             boolean support,
                             boolean rememberMe,
                             String signingSecret,
                             String externalApex,  根域名
                             HttpServletResponse response) {
    long duration;
    int maxAge;

    if (rememberMe) {
        // "Remember me"
        duration = LONG_SESSION;
    } else {
        duration = SHORT_SESSION;
    }
    maxAge = (int) (duration / 1000);

    String token = Sign.generateSessionToken(userId, signingSecret, support, duration);

    Cookie cookie = new Cookie(AuthConstant.COOKIE_NAME, token);
    cookie.setPath("/");
    cookie.setDomain(externalApex);
    cookie.setMaxAge(maxAge);
    cookie.setHttpOnly(true);
    response.addCookie(cookie);
}
```

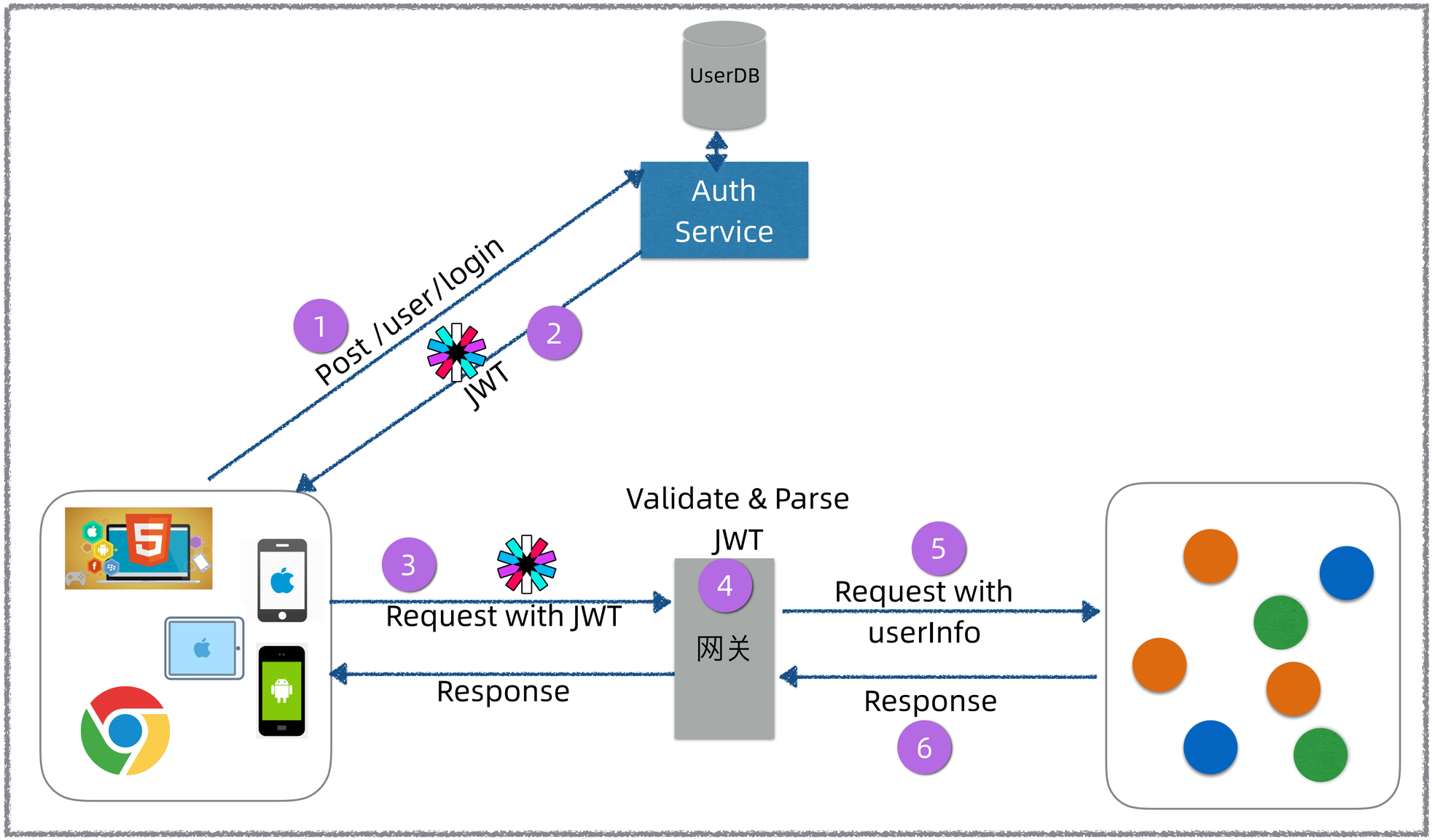
Staffjoy JWT Cookie



Cookie中取出JWT令牌(common/sessions)

```
public static String getToken(HttpServletRequest request) {
    Cookie[] cookies = request.getCookies();
    if (cookies == null || cookies.length == 0) return null;
    Cookie tokenCookie = Arrays.stream(cookies)
            .filter(cookie -> AuthConstant.COOKIE_NAME.equals(cookie.getName()))
            .findAny().orElse( other: null);
    if (tokenCookie == null) return null;
    return tokenCookie.getValue();
}
```

Auth 3.6 ~ JWT + Gateway



JWT校验和取出用户会话数据 (faraday/AuthRequestInterceptor)

```
private Session getSession(HttpServletRequest request) {
    String token = Sessions.getToken(request);
    if (token == null) return null;
    try {
        DecodedJWT decodedJWT = Sign.verifySessionToken(token, signingSecret);
        String userId = decodedJWT.getClaim(Sign.CLAIM_USER_ID).asString();
        boolean support = decodedJWT.getClaim(Sign.CLAIM_SUPPORT).asBoolean();
        Session session = Session.builder().userId(userId).support(support).build();
        return session;
    } catch (Exception e) {
        log.error( message: "fail to verify token",  ...params: "token", token, e);
        return null;
    }
}

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
private static class Session {
    private String userId;
    private boolean support;
}
```

网关传递认证授权信息(faraday/AuthRequestInterceptor)

```
private String setAuthHeader(RequestData data, MappingProperties mapping) {
    // default to anonymous web when prove otherwise
    String authorization = AuthConstant.AUTHORIZATION_ANONYMOUS_WEB;
    HttpHeaders headers = data.getHeaders();
    Session session = this.getSession(data.getOriginRequest());
    if (session != null) {
        if (session.isSupport()) {
            authorization = AuthConstant.AUTHORIZATION_SUPPORT_USER;
        } else {
            authorization = AuthConstant.AUTHORIZATION_AUTHENTICATED_USER;
        }

        this.checkBannedUsers(session.getUserId());

        headers.set(AuthConstant.CURRENT_USER_HEADER, session.getUserId());
    } else {
        // prevent hacking
        headers.remove(AuthConstant.CURRENT_USER_HEADER);
    }
    headers.set(AuthConstant.AUTHORIZATION_HEADER, authorization);

    return authorization;
}
```

登录logout(common/sessions)

```
public static void logout(String externalApex, HttpServletResponse response) {
    Cookie cookie = new Cookie(AuthConstant.COOKIE_NAME,  value: "");
    cookie.setPath("/");
    cookie.setMaxAge(0);
    cookie.setDomain(externalApex);
    response.addCookie(cookie);
}
```

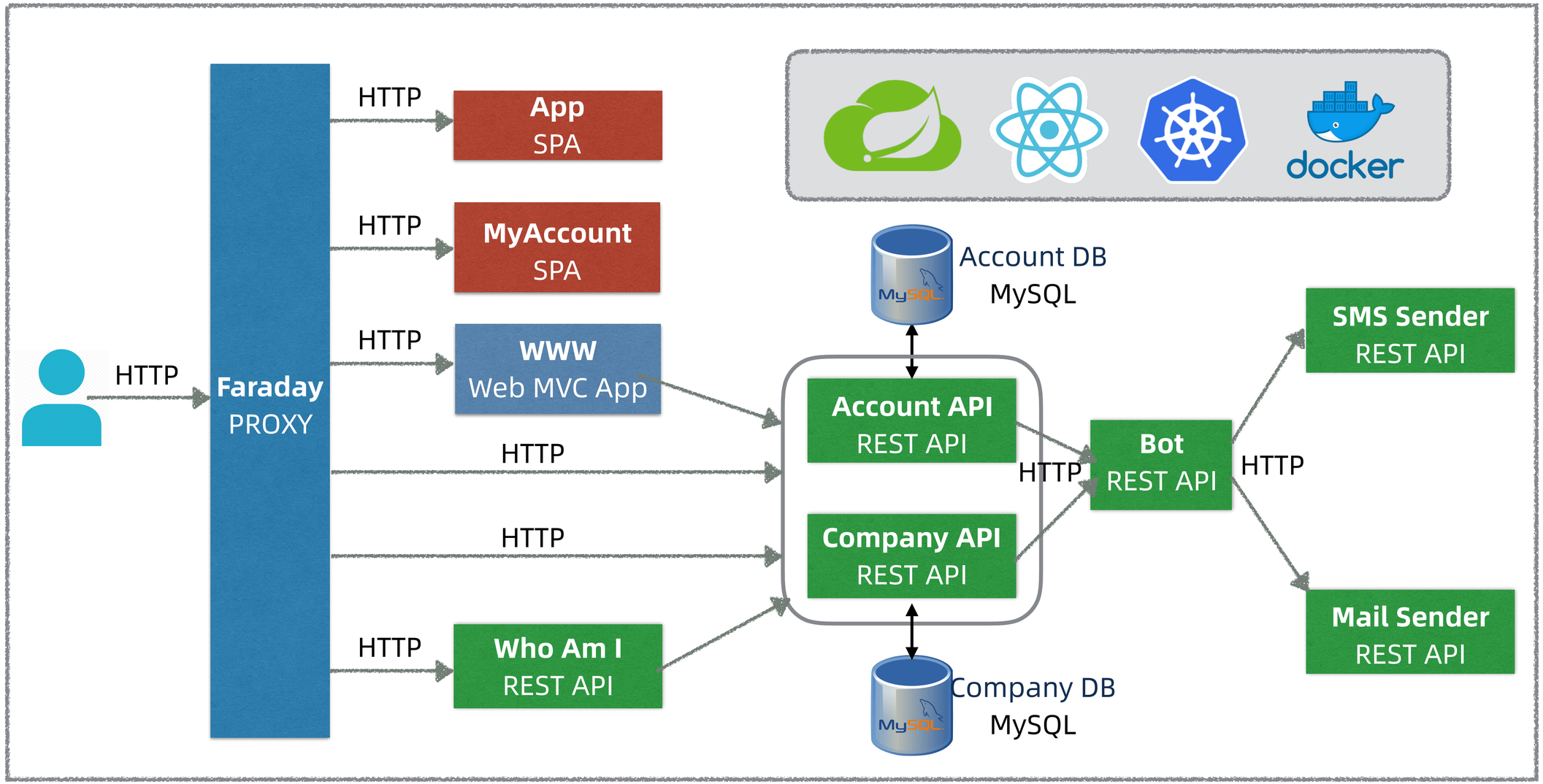
认证上下文助手类(common/AuthContext)

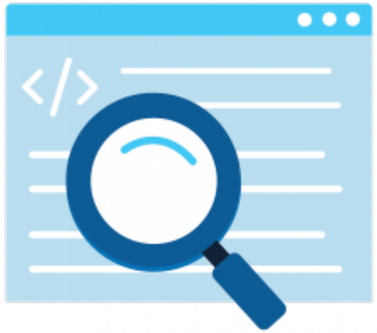
```java
/**
 *
 * A context holder class for holding the current userId and authz info
 *
 * @author bobo
 */
public class AuthContext {

    private static String getRequetHeader(String headerName) {
        RequestAttributes requestAttributes = RequestContextHolder.getRequestAttributes();
        if (requestAttributes instanceof ServletRequestAttributes) {
            HttpServletRequest request = ((ServletRequestAttributes)requestAttributes).getRequest();
            String value = request.getHeader(headerName);
            return value;
        }
        return null;
    }

    public static String getUserId() { return getRequetHeader(AuthConstant.CURRENT_USER_HEADER); }

    public static String getAuthz() {
        return getRequetHeader(AuthConstant.AUTHORIZATION_HEADER);
    }

}
```

Feign客户端传递用户认证信息 向后传递 (common/FeignRequestHeaderInterceptor)

```java
/**
 * Feign interceptor, for passing auth info to backend
 *
 * @author bobo
 */
public class FeignRequestHeaderInterceptor implements RequestInterceptor {

    @Override
    public void apply(RequestTemplate requestTemplate) {
        String userId = AuthContext.getUserId();
        if (!StringUtils.isEmpty(userId)) {
            requestTemplate.header(AuthConstant.CURRENT_USER_HEADER, userId);
        }
    }
}
```

Staffjoy教学版架构





第 9 部分

安全认证代码剖析~服务调用鉴权

服务间调用授权截获器(common/AuthorizeInterceptor)

```java
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Authorize {
    // allowed consumers
    String[] value();
}
```

```java
public class AuthorizeInterceptor extends HandlerInterceptorAdapter {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception
        if (!(handler instanceof HandlerMethod)) {
            return true;
        }

        HandlerMethod handlerMethod = (HandlerMethod) handler;
        Authorize authorize = handlerMethod.getMethod().getAnnotation(Authorize.class);
        if (authorize == null) {
            return true; // no need to authorize
        }

        String[] allowedHeaders = authorize.value();
        String authzHeader = request.getHeader(AuthConstant.AUTHORIZATION_HEADER);

        if (StringUtils.isEmpty(authzHeader)) {
            throw new PermissionDeniedException(AuthConstant.ERROR_MSG_MISSING_AUTH_HEADER);
        }

        if (!Arrays.asList(allowedHeaders).contains(authzHeader)) {
            throw new PermissionDeniedException(AuthConstant.ERROR_MSG_DO_NOT_HAVE_ACCESS);
        }

        return true;
    }
}
```

控制器调用鉴权 (account-svc/AccountController)

```
@PutMapping(path = "/update")
@Authorize(value = {
        AuthConstant.AUTHORIZATION_WWW_SERVICE,
        AuthConstant.AUTHORIZATION_COMPANY_SERVICE,
        AuthConstant.AUTHORIZATION_AUTHENTICATED_USER,
        AuthConstant.AUTHORIZATION_SUPPORT_USER,
        AuthConstant.AUTHORIZATION_SUPERPOWERS_SERVICE
})
public GenericAccountResponse updateAccount(@RequestBody @Valid AccountDto newAccountDto) {
    this.validateAuthenticatedUser(newAccountDto.getId());
    this.validateEnv();

    AccountDto accountDto =  accountService.update(newAccountDto);

    GenericAccountResponse genericAccountResponse = new GenericAccountResponse(accountDto);
    return genericAccountResponse;
}

@PutMapping(path = "/update_password")
@Authorize(value = {
        AuthConstant.AUTHORIZATION_WWW_SERVICE,
        AuthConstant.AUTHORIZATION_AUTHENTICATED_USER,
        AuthConstant.AUTHORIZATION_SUPPORT_USER
})
public BaseResponse updatePassword(@RequestBody @Valid UpdatePasswordRequest request) {
    this.validateAuthenticatedUser(request.getUserId());

    accountService.updatePassword(request.getUserId(), request.getPassword());
```

用户角色和环境鉴权 (account-svc/AccountController)

```
private void validateAuthenticatedUser(String userId) {
    if (AuthConstant.AUTHORIZATION_AUTHENTICATED_USER.equals(AuthContext.getAuthz())) {
        String currentUserId = AuthContext.getUserId();
        if (StringUtils.isEmpty(currentUserId)) {
            throw new ServiceException("failed to find current user id");
        }
        if (!userId.equals(currentUserId)) {
            throw new PermissionDeniedException("You do not have access to this service");
        }
    }
}

private void validateEnv() {
    if (AuthConstant.AUTHORIZATION_SUPERPOWERS_SERVICE.equals(AuthContext.getAuthz())) {
        if (!EnvConstant.ENV_DEV.equals(this.envConfig.getName())) {
            logger.warn( message: "Development service trying to connect outside development environment");
            throw new PermissionDeniedException("This service is not available outside development environments");
        }
    }
}
```

API Client传递服务调用方 (account-api/AccountClient)

```java
public interface AccountClient {

    @PostMapping(path = "/create")
    GenericAccountResponse createAccount(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @RequestBody @

    @PostMapping(path = "/track_event")
    BaseResponse trackEvent(@RequestBody @Valid TrackEventRequest request);

    @PostMapping(path = "/sync_user")
    BaseResponse syncUser(@RequestBody @Valid SyncUserRequest request);

    @GetMapping(path = "/list")
    ListAccountResponse listAccounts(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @RequestParam int

    // GetOrCreate is for internal use by other APIs to match a user based on their phonenumber or email.
    @PostMapping(path= "/get_or_create")
    GenericAccountResponse getOrCreateAccount(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @RequestB

    @GetMapping(path = "/get")
    GenericAccountResponse getAccount(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @RequestParam @No

    @PutMapping(path = "/update")
    GenericAccountResponse updateAccount(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @RequestBody @

    @GetMapping(path = "/get_account_by_phonenumber")
    GenericAccountResponse getAccountByPhonenumber(@RequestHeader(AuthConstant.AUTHORIZATION_HEADER) String authz, @Req
```

授权Header定义(common/AuthConstant)

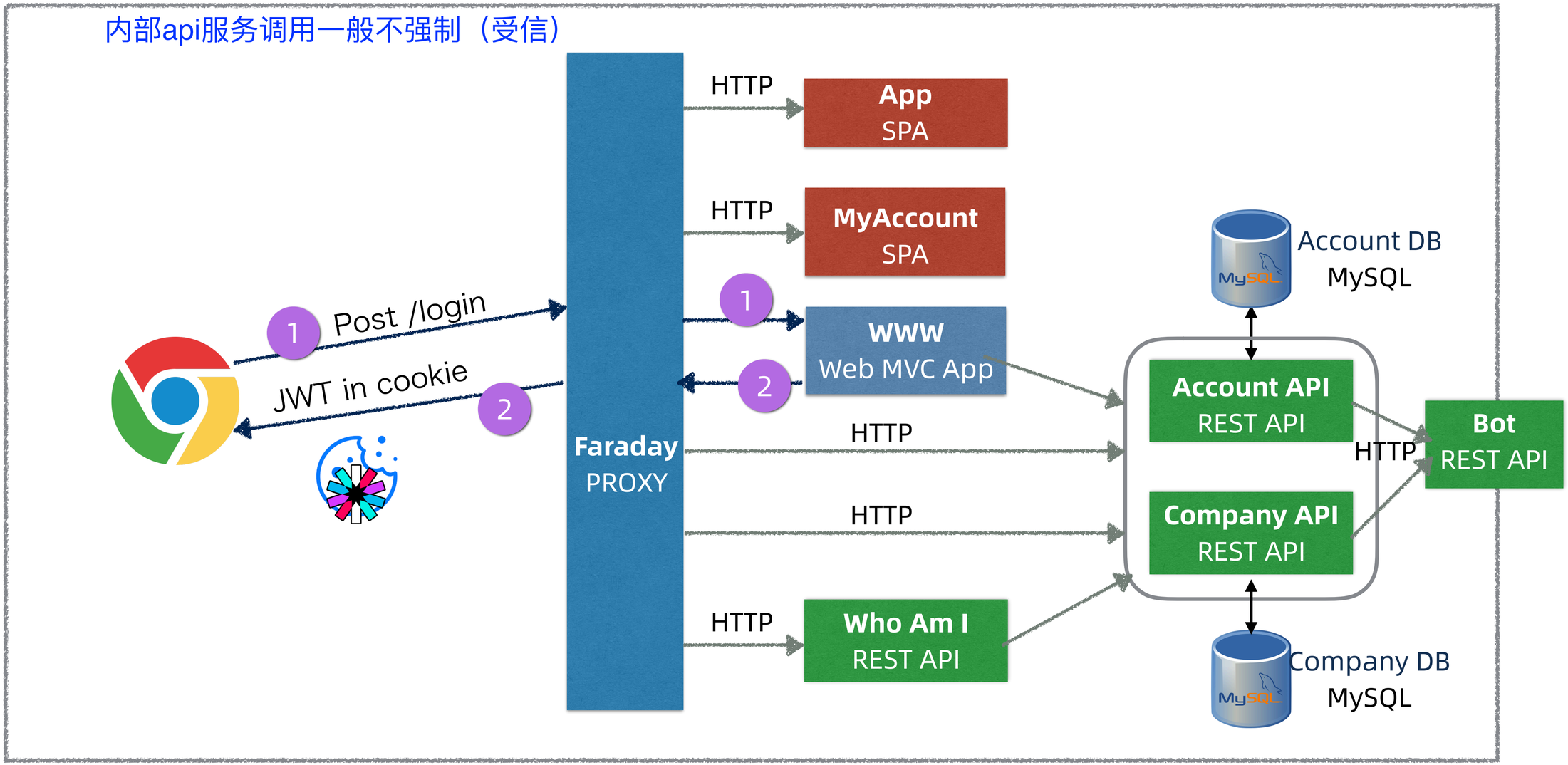
主要是服务名称

```
public class AuthConstant {

    public static final String COOKIE_NAME = "staffjoy-faraday";
    // header set for internal user id
    public static final String CURRENT_USER_HEADER = "faraday-current-user-id";
    //...
    public static final String AUTHORIZATION_HEADER = "Authorization";
    //...
    public static final String AUTHORIZATION_ANONYMOUS_WEB = "faraday-anonymous";
    //...
    public static final String AUTHORIZATION_COMPANY_SERVICE = "company-service";
    //...
    public static final String AUTHORIZATION_BOT_SERVICE = "bot-service";
    //...
    public static final String AUTHORIZATION_ACCOUNT_SERVICE = "account-service";
    //...
    public static final String AUTHORIZATION_SUPPORT_USER = "faraday-support";
    //...
    public static final String AUTHORIZATION_SUPERPOWERS_SERVICE = "superpowers-service";
    //...
    public static final String AUTHORIZATION_WWW_SERVICE = "www-service";
    //...
    public static final String AUTHORIZATION_WHOAMI_SERVICE = "whoami-service";
    //...
    public static final String AUTHORIZATION_AUTHENTICATED_USER = "faraday-authenticated";
    //...
    public static final String AUTHORIZATION_ICAL_SERVICE = "ical-service";
```

Staffjoy Auth Enforcement

用户认证鉴权可以强制(网关保证)

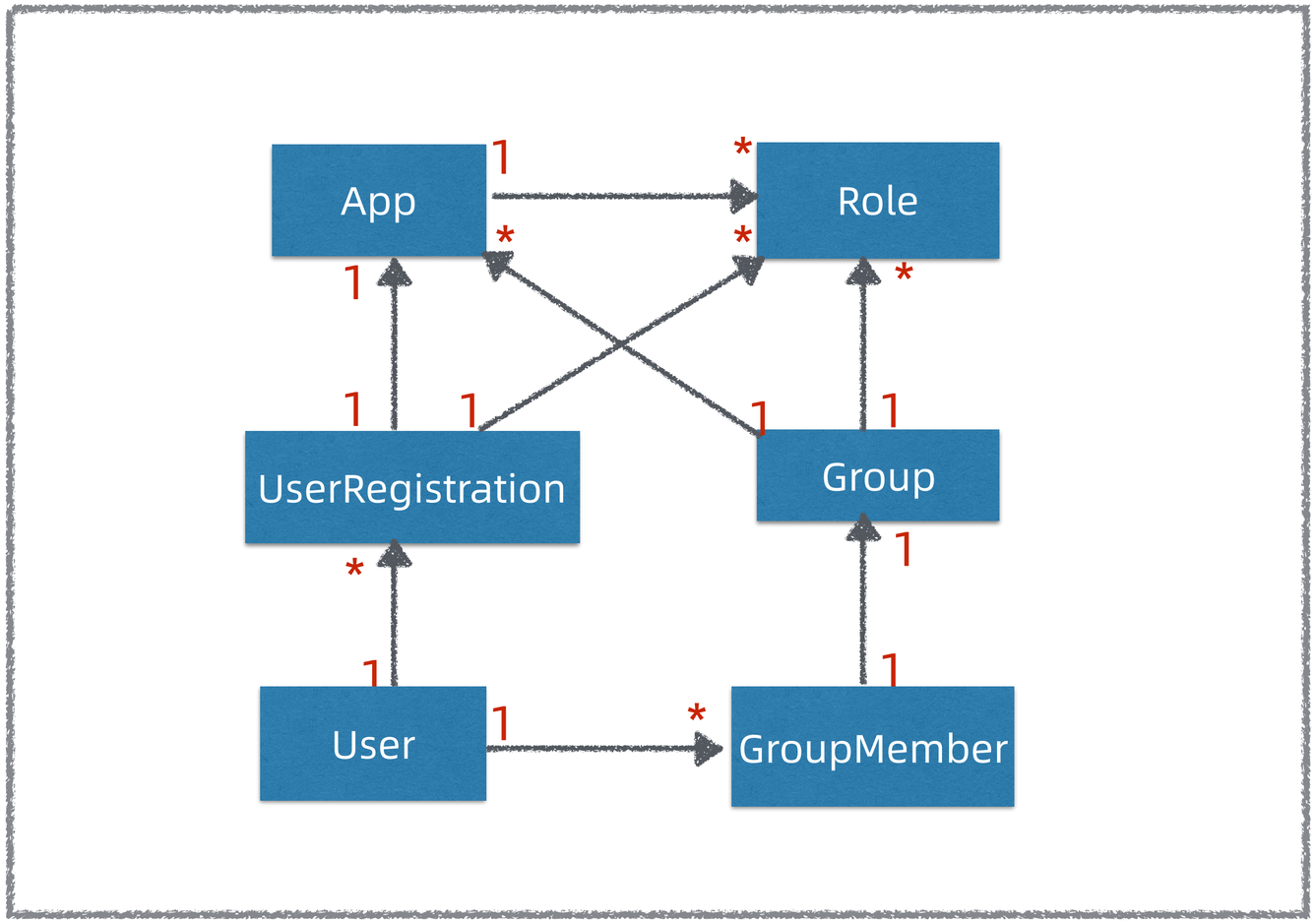
内部api服务调用一般不强制（受信）



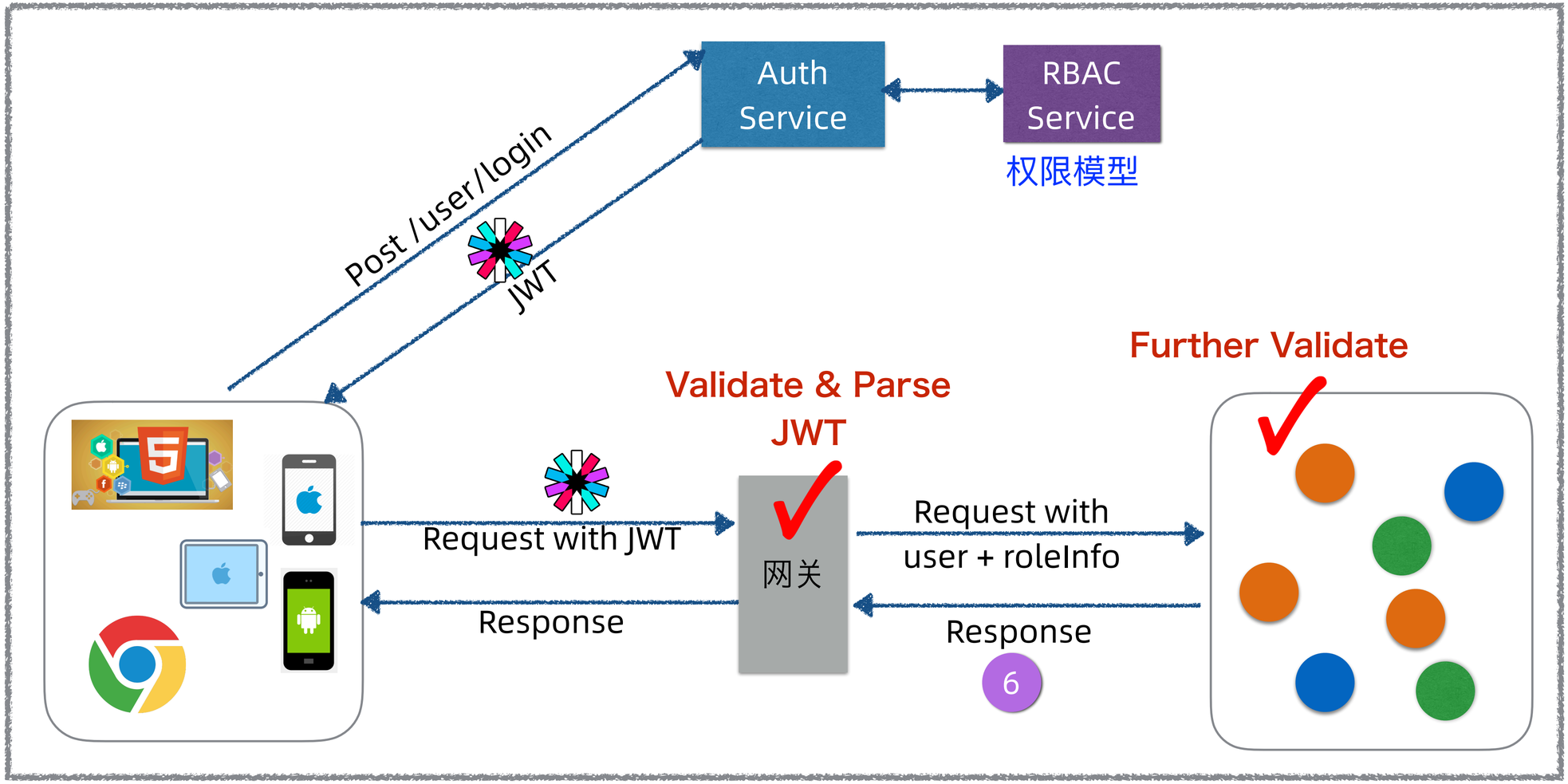
第 10 部分

用户角色鉴权扩展

参考权限模型



Auth 3.7 ~ JWT + RBAC



参考链接

1. **RFC 7519** https://tools.ietf.org/html/rfc7519.html
2. **jwt.io** https://jwt.io/
3. **Java implementation of JSON Web Token (JWT) by Auth0** https://github.com/auth0/java-jwt

极客时间

扫码试看/订阅

《Spring Boot & Kubernetes 云原生微服务实践》