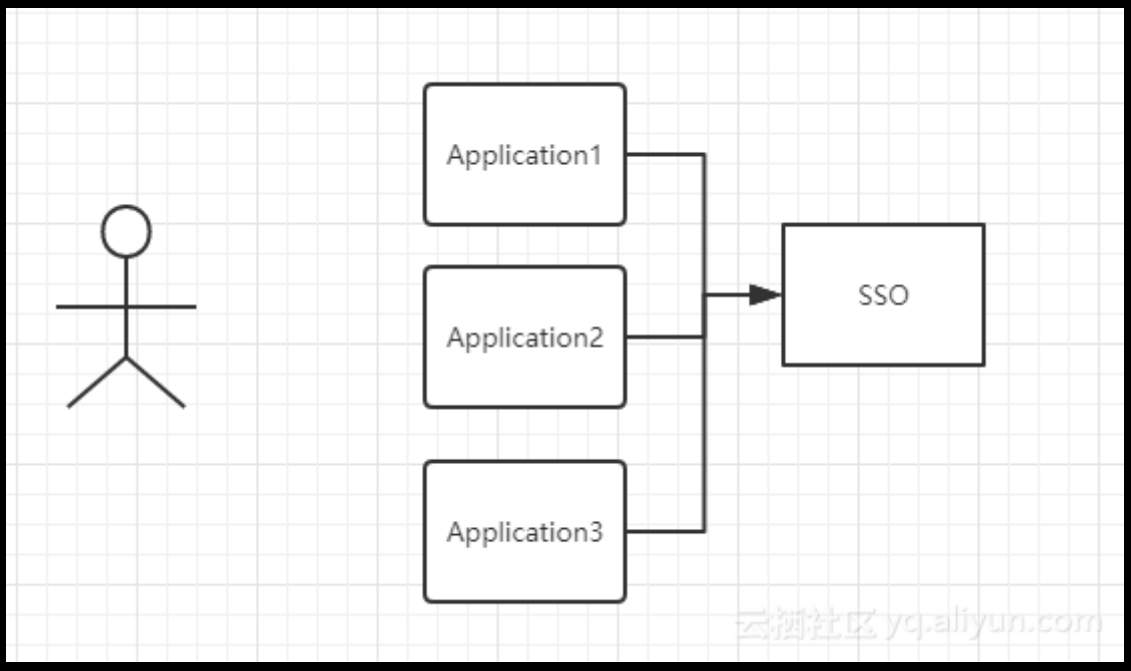
单点登录 SSO

概述

“统一身份认证”

在企业发展初期，企业使用的系统很少，通常一个或者两个，每个系统都有自己的登录模块，运营人员每天用自己的账号登录，很方便。但随着企业的发展，用到的系统随之增多，运营人员在操作不同的系统时，需要多次登录，而且每个系统的账号都不一样，这对于运营人员来说，很不方便。

单点登录英文全称 **Single Sign On**，简称就是 SSO。它的解释是：在多个应用系统中，只需要登录一次，就可以访问其他相互信任的应用系统。



如图所示，图中有 4 个系统，分别是 Application1、Application2、Application3、和 SSO。Application1、Application2、Application3 没有登录模块，而 SSO 只有登录模块，没有其他的业务模块，当 Application1、Application2、Application3 需要登录时，将跳到 SSO 系统，SSO 系统完成登录，其他的应用系统也就随之登录了。这完全符合我们对单点登录（SSO）的定义。

单点登录的常见落地实现技术有哪些?

身份认证技术:

1. cas（单点登录）
2. Spring Security OAuth2（第三方登录授权：QQ 登陆）
3. jwt（客户端 token：原生）

安全控制框架:

1. spring-security
2. shiro:

CAS:
缺点：cas 单点登录技术适用于传统应用的场景比较多， 官方示例也是以 javaWeb 为准， **对微服务化应用，前后端分离应用，支持性较差**。

JWT:
特别适用于分布式站点的单点登录（SSO）场景。JWT 的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该 token 也可直接被用于认证，也可被加密。

基于 JWT 认证协议，自己开发 SSO 服务和权限控制。

CAS

CAS 全称为 **Central Authentication Service 即中央认证服务**，是一个企业多语言单点登录的解决方案，并努力去成为一个身份验证和授权需求的综合平台。

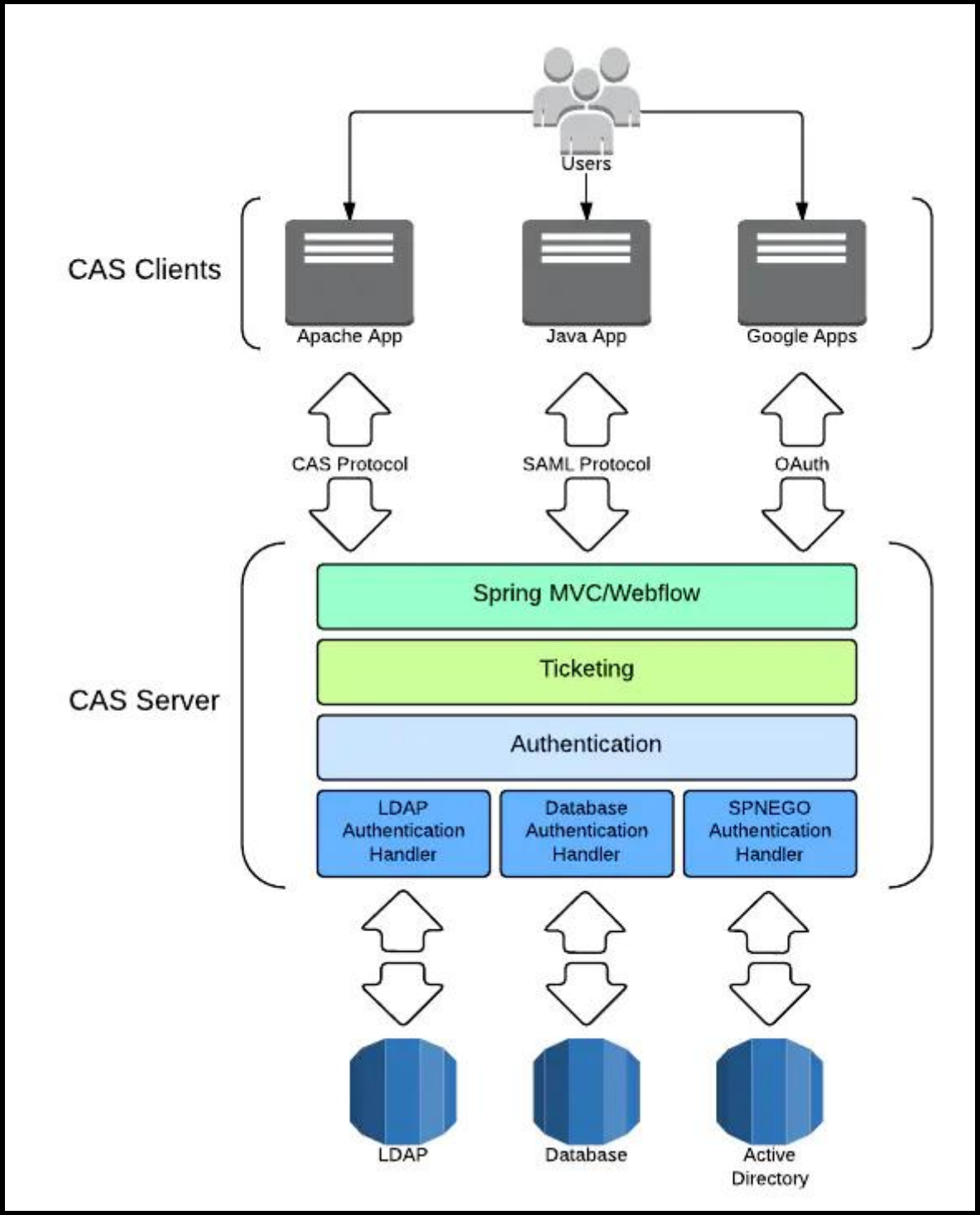
CAS 是由 Yale 大学发起的一个企业级的、开源的项目，旨在为 Web 应用系统提供一种可靠的单点登录解决方法（属于 Web SSO ）

CAS 协议至少涉及三方：客户端 Web 浏览器，请求身份验证的 Web 应用程序和 CAS 服务器。 它也可能涉及后端服务，如数据库服务器，它没有自己的 HTTP 接口，但与 Web 应用程序进行通信。
在 CAS 的结构中主要分两部分，一部分是 CAS Server，另一部分是 CAS Client。
1. CAS Server：CAS Server 负责完成对用户的认证工作 ，需要独立部署 ，CAS Server 会处理用户名 / 密码等凭证(Credentials)。
2. CAS Client：负责处理对客户端受保护资源的访问请求，需要对请求方进行身份认证时，重定向到 CAS Server 进行认证。(原则上，客户端应用不再接受任何的用户名密码等 Credentials)。

CAS 协议是一个简单而强大的基于票据的协议，它涉及一个或多个客户端和一台服务器。即在 CAS 中，通过 TGT(Ticket Granting Ticket)来获取 ST(Service Ticket)，通过 ST 来访问具体服务。

其中主要的关键概念：
TGT（Ticket Granting Ticket）是存储在 TGCcookie 中的代表用户的 SSO 会话。
该 ST（Service Ticket），作为参数在 GET 方法的 URL 中，代表由 CAS 服务器授予访问 CASified 应用程序（包含 CAS 客户端的应用程序）具体用户的权限。

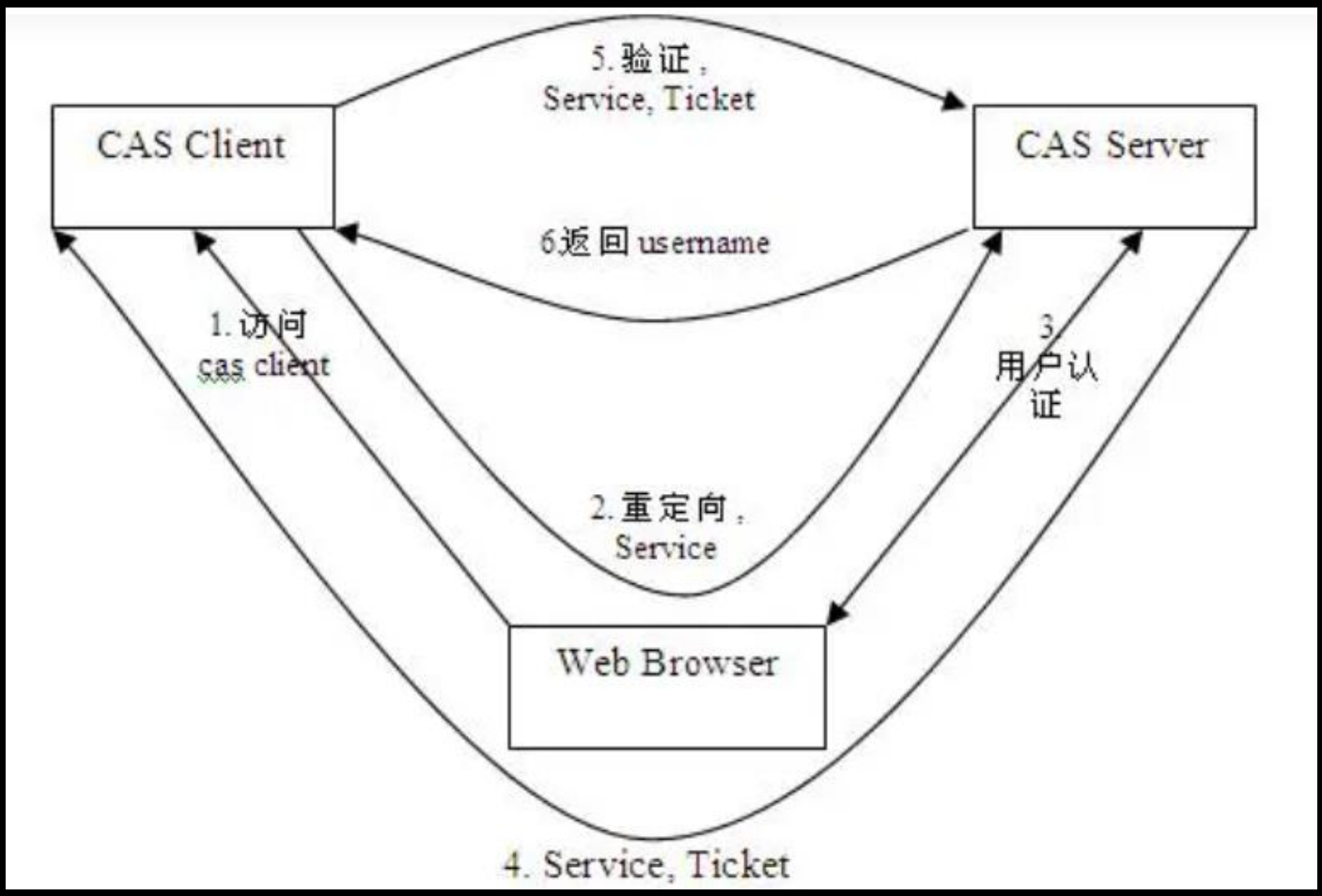
架构：



优点：

1. 开源的、多协议的 **SSO 解决方案.**
2. 支持多种认证机制
3. 安全策略：使用票据（ Ticket ）来实现支持的认证协议
4. 支持授权：可以决定哪些服务可以请求和验证服务票据（ Service Ticket ）
5. 提 供高可用性
6. 支持多种客户端： Java 、 .Net 、 PHP 、 Perl 、 Apache, uPortal 等。
7.

协议过程：



同域下的单点登录

一个企业一般情况下只有一个域名，通过二级域名区分不同的系统。比如我们有个域名叫做：a.com，同时有两个业务系统分别为：app1.a.com 和 app2.a.com。我们要做单点登录（SSO），需要一个登录系统，叫做：sso.a.com。

我们只要在 sso.a.com 登录，app1.a.com 和 app2.a.com 就也登录了。通过上面的登陆认证机制，我们可以知道，在 sso.a.com 中登录了，其实是在 sso.a.com 的服务端的 session 中记录了登录状态，同时在浏览器端(Browser)的 sso.a.com 下写入了 Cookie。那么我们怎么才能让 app1.a.com 和 app2.a.com 登录呢？这里有两个问题：

1. Cookie 是不能跨域的，我们 Cookie 的 domain 属性是 sso.a.com，在给 app1.a.com 和 app2.a.com 发送请求是带不上的。
2. sso、app1 和 app2 是不同的应用，它们的 session 存在自己的应用内，是不共享的。

针对第一个问题，sso 登录以后，可以将 Cookie 的域设置为顶域，即.a.com，这样所有子域的系统都可以访问到顶域的 Cookie。我们在设置 Cookie 时，只能设置顶域和自己的域，不能设置其他的域。