# <https://www.cnblogs.com/ljmatlight/p/8657018.html>

# [CAS 之 Apereo CAS 简介](https://www.cnblogs.com/ljmatlight/p/8657018.html)

# **Background（背景）**

随着公司业务的不断扩展，后台接入子系统不断增多，那么我们将针对不同的平台进行拆分为各自对应的子系统，  
权限是不变的，那么我们不能每个子系统都单独进行登录认证，不然管理人员进行切换系统时会疯掉。  
那么，经过考察选用开源框架 Apereo CAS , 选定版本为 5.2.0。目前系统已在线，并已稳定。  
接下来我会将系统进行脱敏整理出一套完善的基于微服务的CAS单点系统实施方案。  
由于CAS是相对比较大的工程，所以建议使用者认真阅读官方文档进行调整。

# **Intro（介绍）**

Central Authentication Service (CAS)，通常称为CAS。 CAS是一种针对Web的企业多语言单点登录解决方案，并尝试成为您的身份验证和授权需求的综合平台。

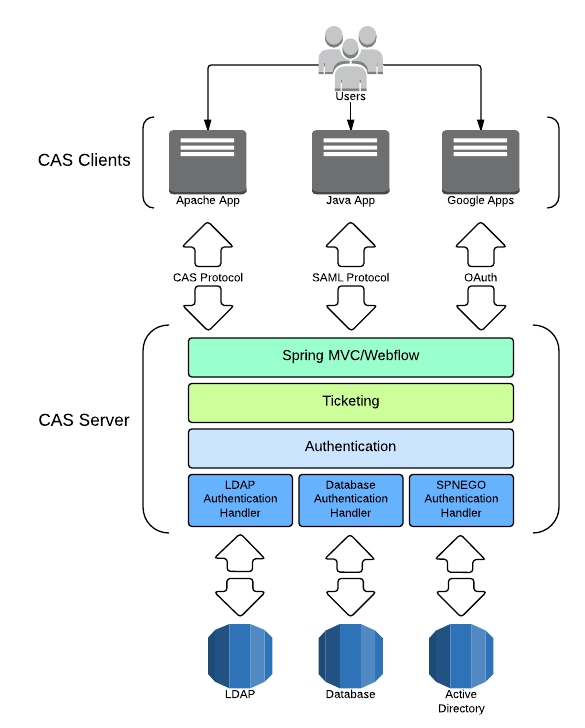
下面是官方的一段简述：

CAS Enterprise Single Sign-On

* Spring Webflow/Spring Boot Java server component.
* 可拔插认证支持 (LDAP, Database, X.509, SPNEGO, JAAS, JWT, RADIUS, MongoDb, etc)
* 多种协议支持 (CAS, SAML, WS-Federation, OAuth2, OpenID, OpenID Connect, REST)
* 通过各种提供商支持多因素身份验证 (Duo Security, FIDO U2F, YubiKey, Google Authenticator, Microsoft Azure, Authy etc)
* 支持外部提供者的委托认证，例如： ADFS, Facebook, Twitter, SAML2 IdPs, etc.
* Built-in support for password management, notifications, terms of use and impersonation.
* Support for attribute release including user consent.
* 实时监控和跟踪应用程序行为，统计信息和日志。
* 用特定的认证策略管理和注册客户端应用程序和服务。
* 跨平台的客户端支持 (Java, .Net, PHP, Perl, Apache, etc).
* Integrations with InCommon, Box, Office365, ServiceNow, Salesforce, Workday, WebAdvisor, Drupal, Blackboard, Moodle, Google Apps, etc.

由以上可以看出， Central Authentication Service (CAS) 支持的功能广泛。

# **Architecture（架构）**



## **System Components （系统组件）**

CAS服务器和CAS客户端组成CAS系统架构的两个物理组件，它们通过各种协议进行通信。

## **CAS Server（CAS服务器）**

CAS服务器是基于Spring框架构建的Java servlet，其主要职责是验证用户并通过发布和验证票证来授予对启用CAS的服务（通常称为CAS客户端）的访问权限。 当服务器在成功登录时向用户发出票证授予票证（TGT）时，将创建SSO会话。 根据用户的请求，通过使用TGT作为标记的浏览器重定向向服务发出服务票据（ST）。 ST随后通过反向信道通信在CAS服务器上进行验证。 CAS Protocol文档中详细描述了这些交互。

### **CAS Clients（CAS客户端）**

术语“CAS客户”在其通用中有两个不同的含义。 CAS客户端是可以通过支持的协议与服务器进行通信的任何CAS支持的应用程序。 CAS客户端也是一个软件包，可以与各种软件平台和应用程序集成，以便通过某种身份验证协议（例如CAS，SAML，OAuth）与CAS服务器进行通信。 已经开发了支持多种软件平台和产品的CAS客户端。

Platforms: （软件平台）

* Apache httpd Server ([mod\_auth\_cas module](https://github.com/apereo/mod_auth_cas))
* Java ([Java CAS Client](https://github.com/apereo/java-cas-client))
* .NET (.NET CAS Client)
* PHP (phpCAS)
* Perl (PerlCAS)
* Python (pycas)
* Ruby (rubycas-client)

Applications:（平台）

* Canvas
* Atlassian Confluence
* Atlassian JIRA
* Drupal
* Liferay
* uPortal
* …

## **Supported Protocols （支持协议）**

客户端通过几种支持的协议与服务器进行通信。 所有支持的协议在概念上都是相似的，但有些协议的特征或特征使得它们适用于特定的应用程序或用例。 例如，CAS协议支持委托（代理）认证，并且SAML协议支持属性发布和单一注销。

Supported protocols:

* [CAS (versions 1, 2, and 3)](https://apereo.github.io/cas/5.2.x/protocol/CAS-Protocol.html)
* [SAML 1.1 and 2](https://apereo.github.io/cas/5.2.x/protocol/SAML-Protocol.html)
* [OpenID Connect](https://apereo.github.io/cas/5.2.x/protocol/OIDC-Protocol.html)
* [OpenID](https://apereo.github.io/cas/5.2.x/protocol/OpenID-Protocol.html)
* [OAuth 2.0](https://apereo.github.io/cas/5.2.x/protocol/OAuth-Protocol.html)
* [WS Federation](https://apereo.github.io/cas/5.2.x/protocol/WS-Federation-Protocol.html)

## **Software Components（软件组件）**

根据三层子系统来描述CAS服务器是有帮助的：

* Web（Spring MVC / Spring Webflow）
* [Ticketing](https://apereo.github.io/cas/5.2.x/installation/Configuring-Ticketing-Components.html)
* [Authentication](https://apereo.github.io/cas/5.2.x/installation/Configuring-Authentication-Components.html)

几乎所有的部署考虑和组件配置都涉及这三个子系统。 Web层是与包括CAS客户端在内的所有外部系统进行通信的端点。 Web层委托票务子系统为CAS客户端访问生成票证。 SSO会话以成功认证时颁发票证授予票证开始，因此票务子系统经常委托给认证子系统。  
认证系统通常只在SSO会话开始时处理请求，尽管还有其他情况可以调用它（例如强制认证）。

# **Conclusions（结论）**



由上图可以看出， Central Authentication Service (CAS) 在很多大公司内进行了很好的应用。

# **Recommendations（建议）**

It is recommended to deploy CAS locally using the WAR Overlay method.

1. 建议使用WAR Overlay方法在本地部署CAS。
2. 建议明白几点
   * 2.1 何种应用在何种情况下产生何种事物用于何种目的，白话一点就是要清楚 一种事物的生命周期及变换过程。
   * 2.2 一种新的事物的产生必然有其缘由，知其然知其所以然 才能不断地成长。
   * 2.3 一般情况下，相对完善的事物产生的同时，都会有与之对应的文档。那么学习的最好途径就是仔细阅读实践。
   * 2.4 即使是官方推出的，对自身应用来讲也不一定是最好的，要根据实际情况进行对应的实践验证。
   * 2.5 尽信书不如无书。

# **参考资料**

[Apereo CAS](https://apereo.github.io/cas/5.2.x/index.html)  
<https://github.com/apereo/cas>