The Anatomy of a Secure Web Application Using Java – Redux

CON2323

By John Field, Shawn McKinney

@architectedsec, @shawnmckinney

JavaOne

October 28, 2015

Pivotal

Be A “Full Stack” Developer

“No one can know everything about everything,
but you should be able to visualize what happens
up and down the stack
as an application does its thing.”

-- Carlos Bueno of Facebook, c. 2010

Pivotal

Be A “Full Security Stack” Developer

“No one can know everything about everything,
but you should be able to visualize what happens
up and down the stack
as an application does its thing securely.”

-- John and Shawn, c. 2014

Pivotal

Why Is Security So Hard?

- Security is omitted from most programming examples.
- Done right, security is orthogonal to the business use case.
- Testing security is tricky. Stubs & mocks may be unrealistic, incomplete.
- Implementation details often unique to an organization.
- A good security architecture composes patterns.
- The patterns transcend the individual applications.
- Security patterns often look similar, but differ in subtle ways.

Pivotal

Anatomy of a Secure Web Application Using Java

- An anatomical model of a secure Java application, just like the original plastic model kits:

  “The Visible Man” and
  “The Visible Woman”

Image credit: http://amhistory.si.edu/img/collections_xlarge/99-2741_428px.jpg

Pivotal

Our “Full Security Stack” Quest

1. Authentication for the Cloud
   - SAML-based single sign-on for fortress-saml-demo
2. Authorization for the Cloud
   - Extending apache-fortress-demo with OAuth2
3. End-to-End security for Java Web Applications
   - apache-fortress-demo
4. Forklifting a secure Web application into Cloud Foundry
   - apache-fortress-demo

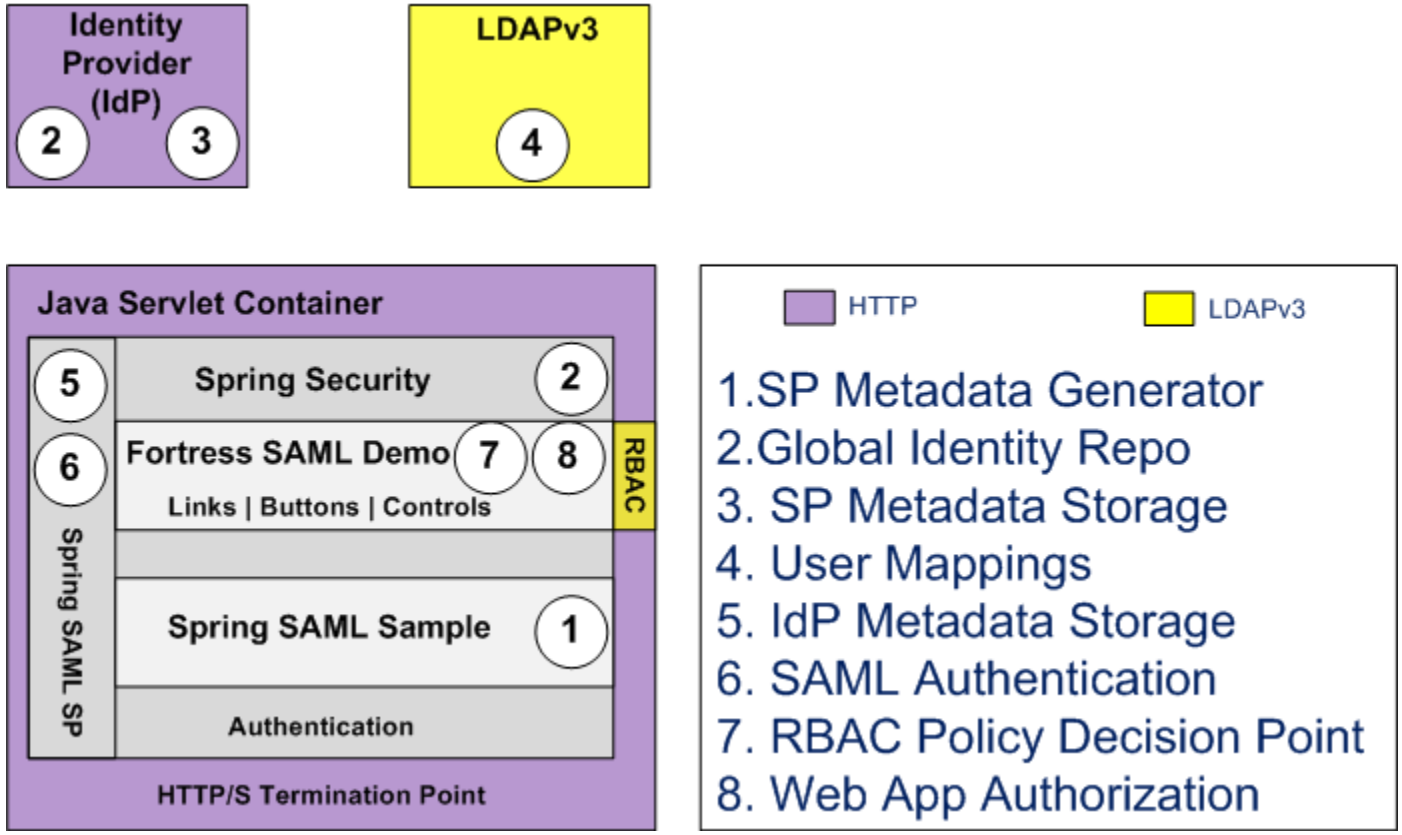
Based on examples from real-world customer use cases.

Pivotal

Go Mets!!

Demo #1

Apache Fortress SAML Demo



http://iamfortress.net/2015/09/01/apache-directory-fortress-saml-demo/

symas

Take the Crown!

symas

System Architecture Take 1

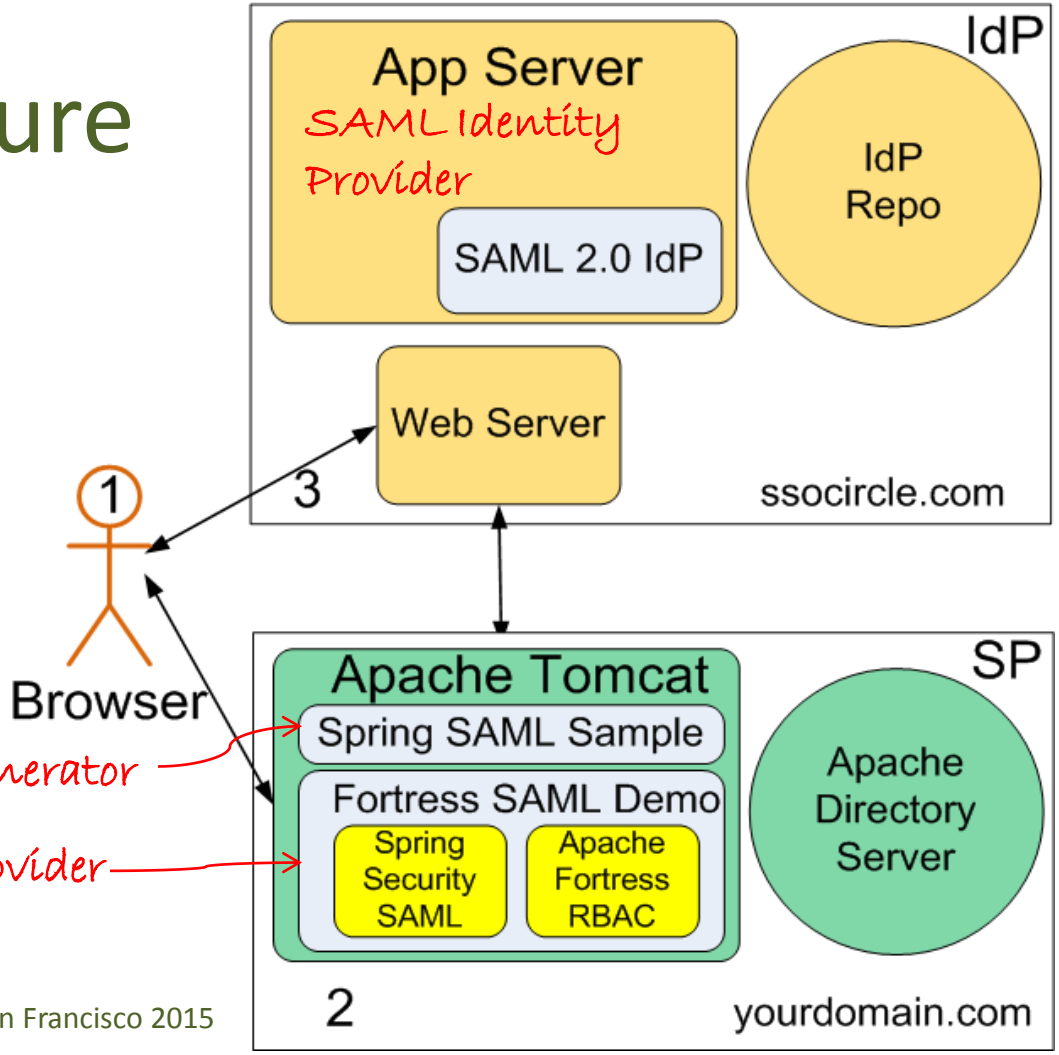
(as downloaded)

Use

SSO Circle

Identity

Provider



System Architecture Take 2

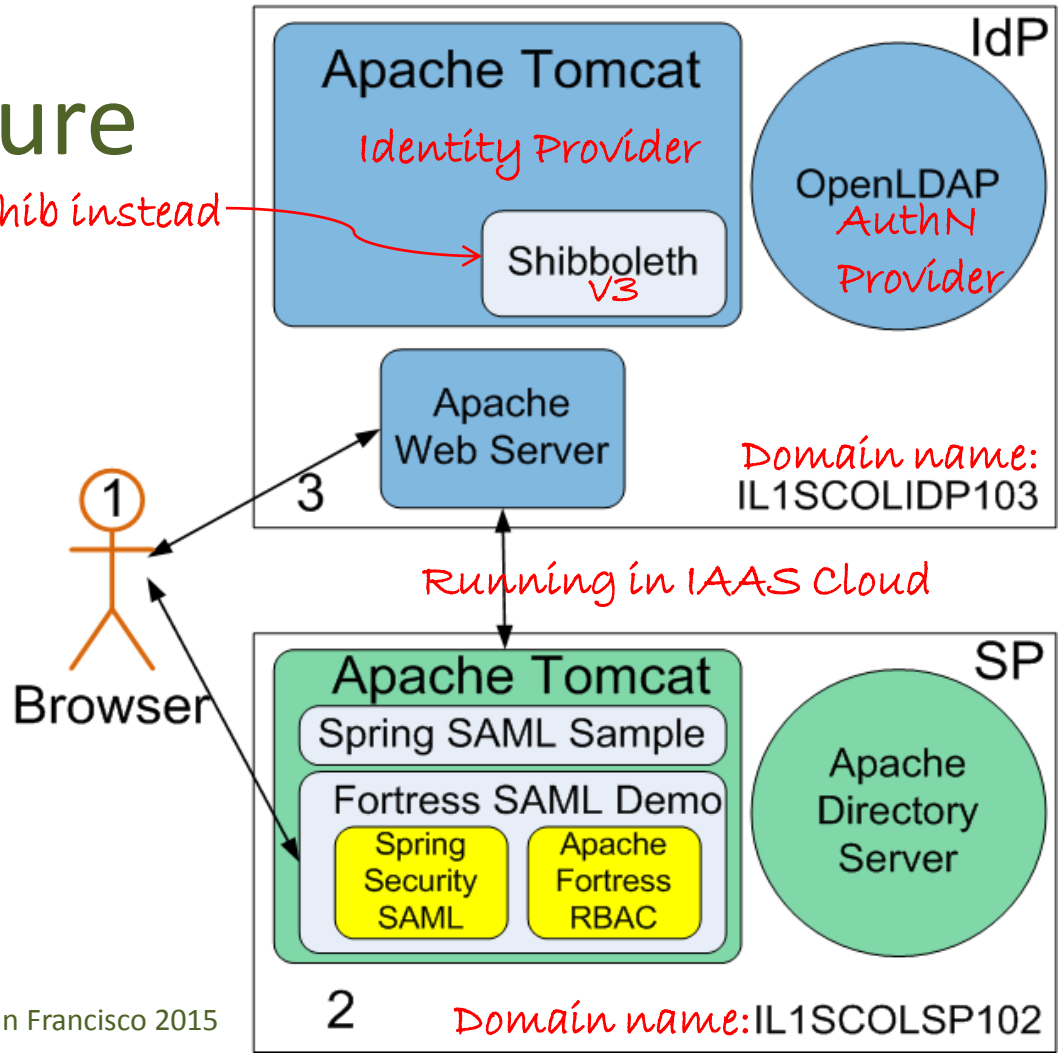
(for today's demo)

Use

Shibboleth

Identity

Provider



System Architecture Take 3

(for another day)

Enterprises would like to do this

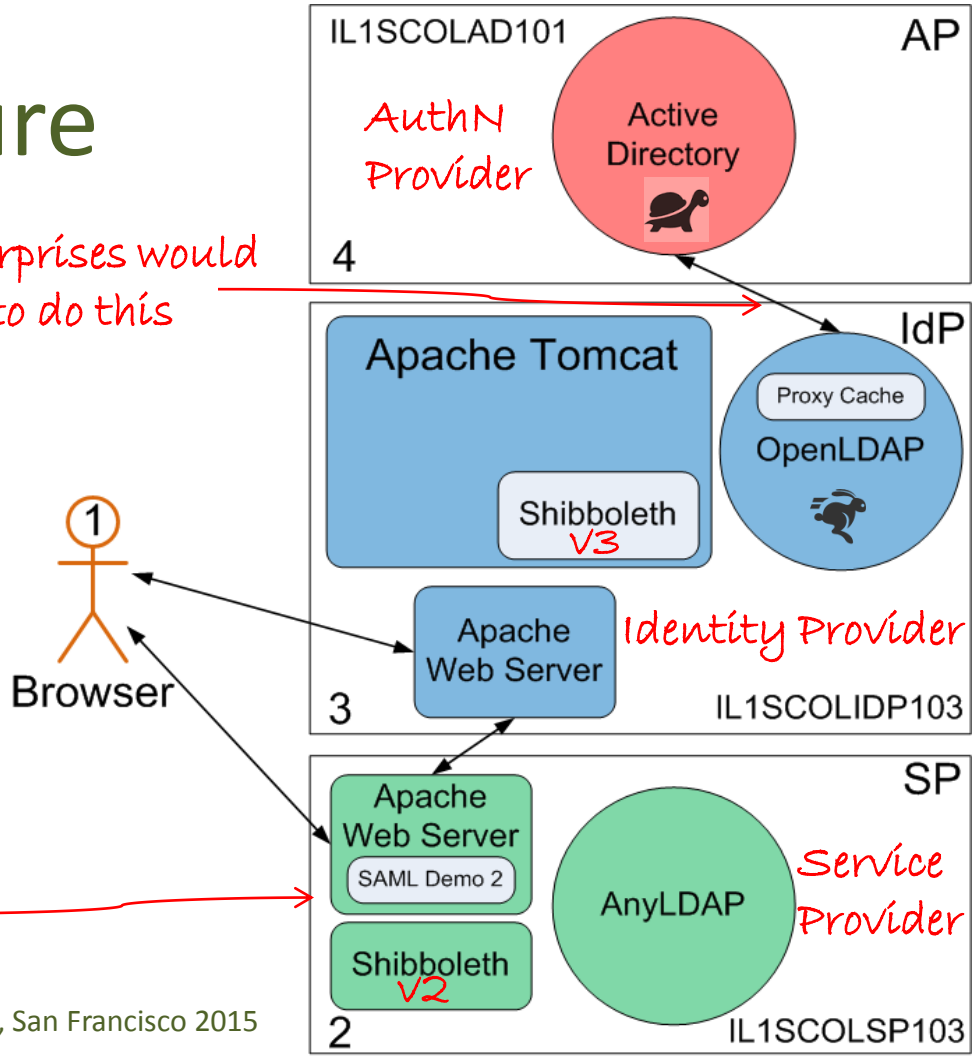
Use

Active Directory

Authentication

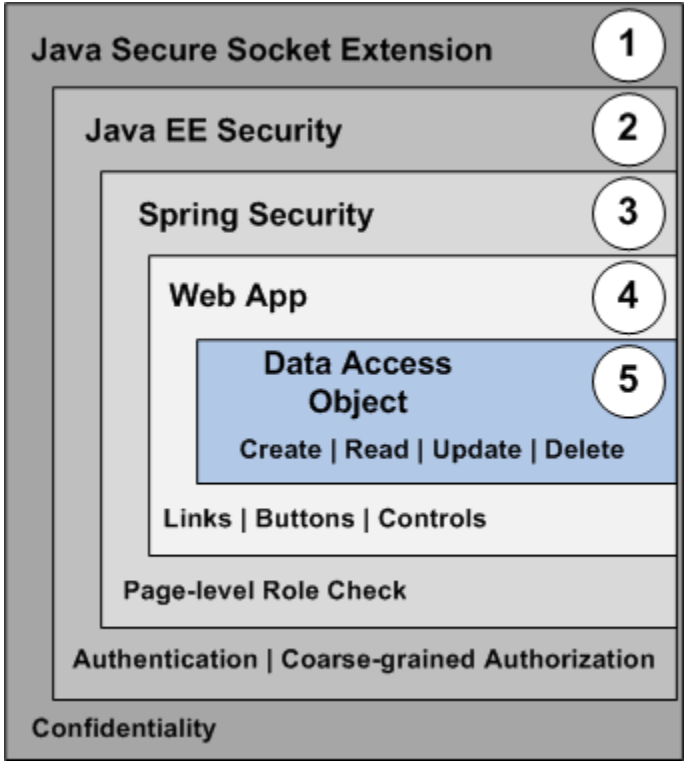
Provider

Could be any platform



The Five Security Layers of Java Web Apps

1. Java Secure Socket Extension (JSSE)
2. Java EE Security
3. Spring Security
4. Web App Framework
5. Database Functions



Security Layers with SAML Demo

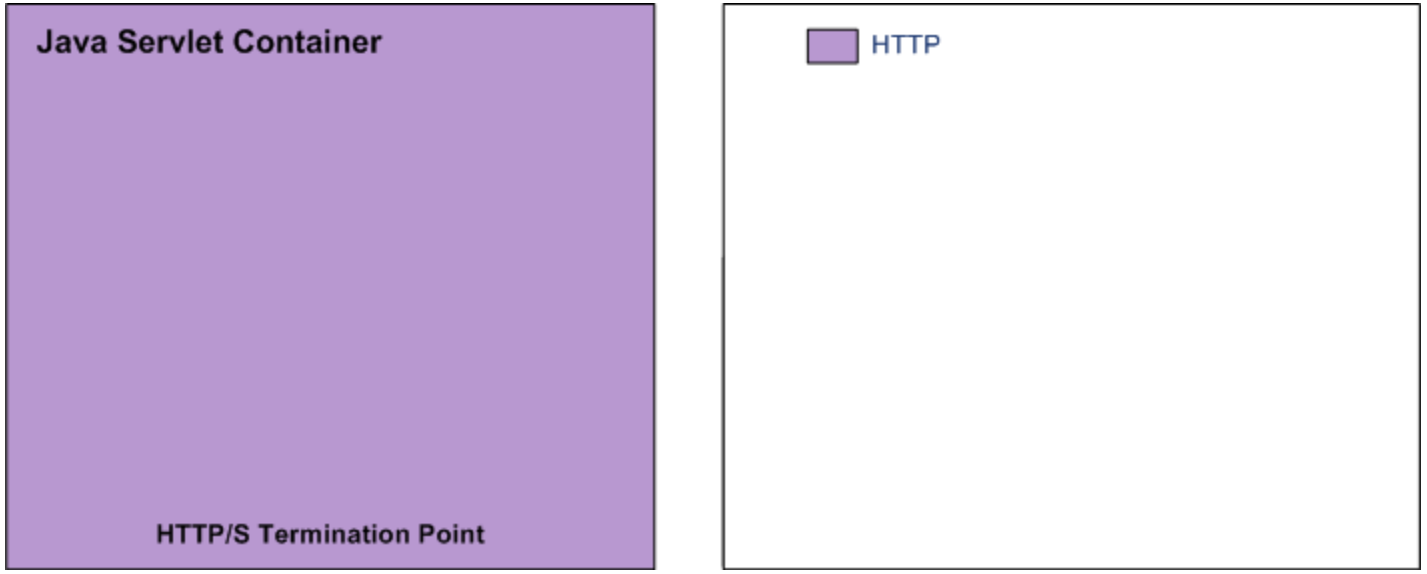
1. JSSE
2. ~~Java EE Security~~ ← Turned off (for now)
3. Spring Security ← SAML 2.0 authN
4. Web App Framework ← Fine-grained AuthZ

Two Areas of Access Control

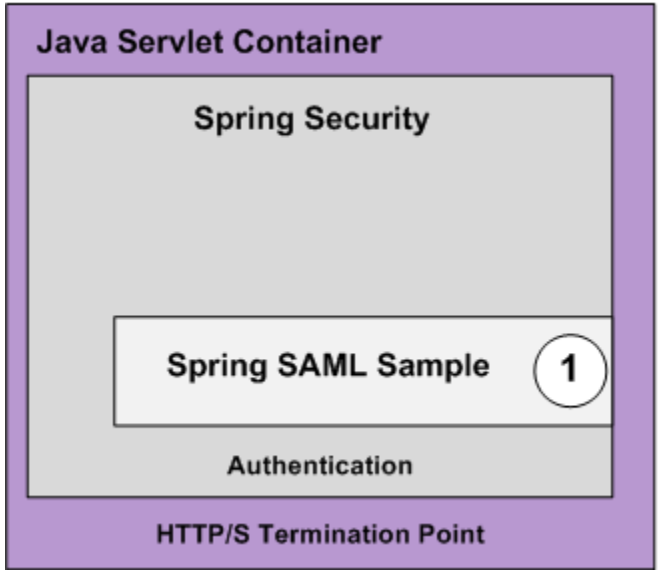
1. Spring Security Declarative checks

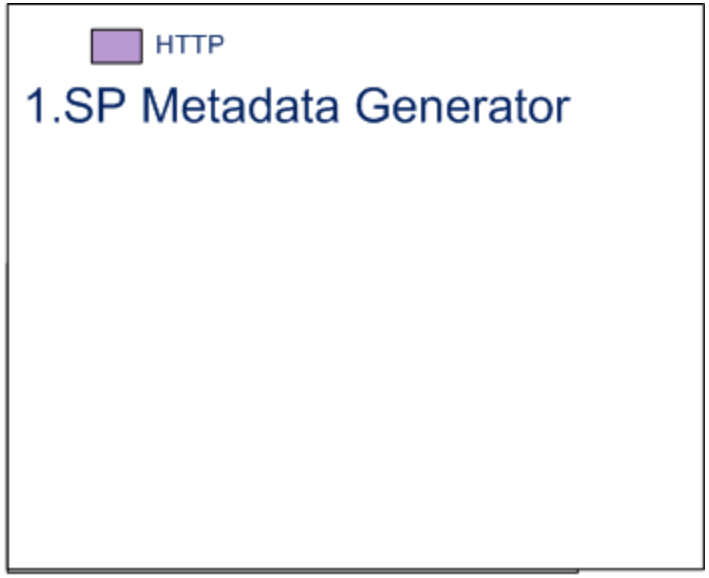
2. Apache Fortress Programmatic checks

Start with Tomcat Servlet Container



1. Deploy the Spring SAML Sample





Get the Spring SAML Sample

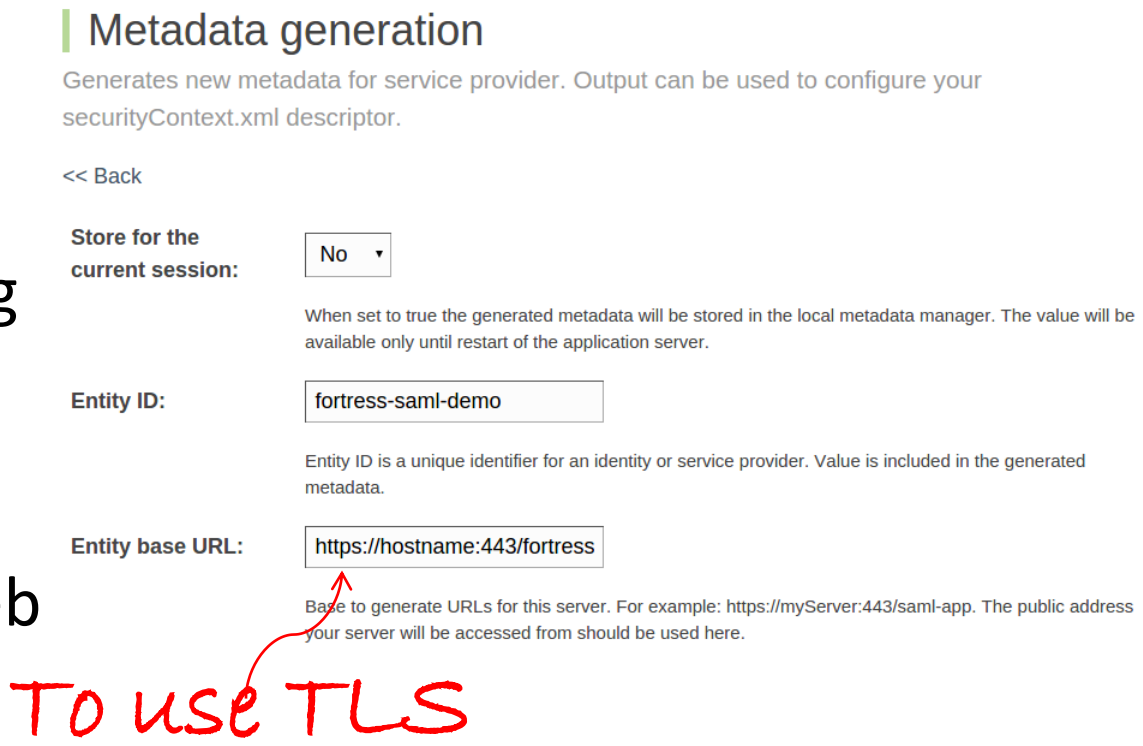
Pick one:

1. spring-security-saml - Spring's SAML sample is the first place java developers should look for basic SAML 2.0 programming concepts.
2. shibboleth-sample-java-sp - Unicon's is derived from above and is how to learn about Spring SAML's SP with a Shibboleth's IdP.

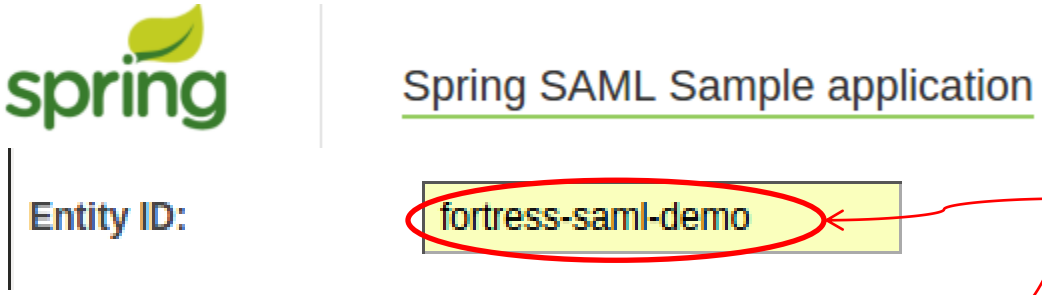
Generate SAML Service Provider Metadata

Matching Fields:

- Entity ID must match Spring config in web app
- Entity base URL must match the web app's URL.



Spring SAML Metadata Generation Tip

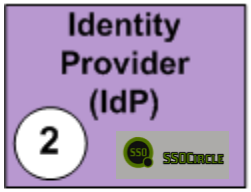


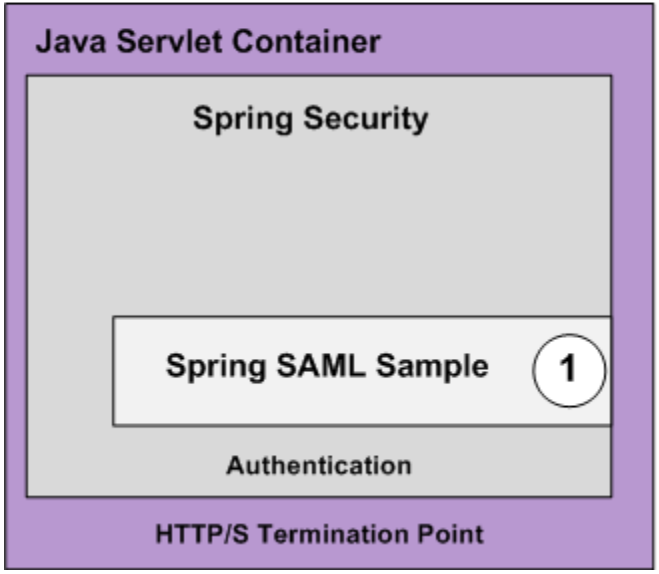
These entityId's must match

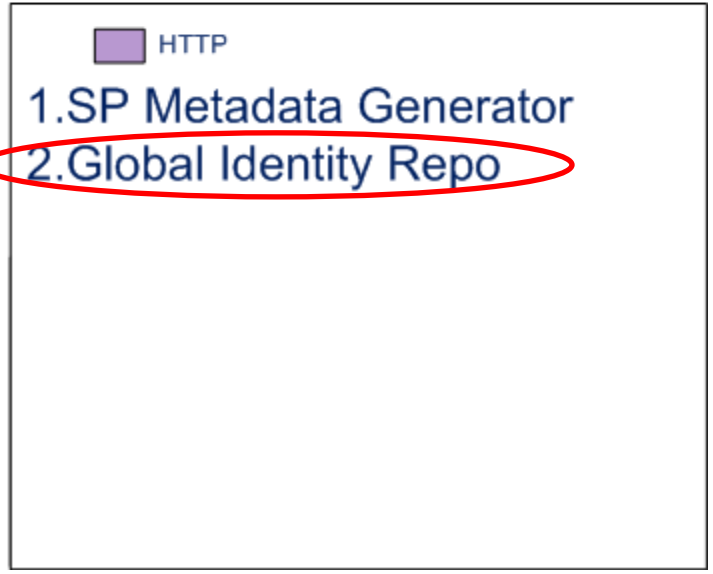
```
<bean id="metadataGeneratorFilter" class="org.springframework…MetadataGeneratorFilter">
 <constructor-arg>
 <bean class="org.springframework…MetadataGenerator”>
  <property name="entityId" value="fortress-saml-demo"/>
 </bean>
 </constructor-arg>
</bean>
```

Bind the service provider with the IdP.

2. Setup Global Identity Provider







Setup SSOCircle SAMLv2.0 IdP

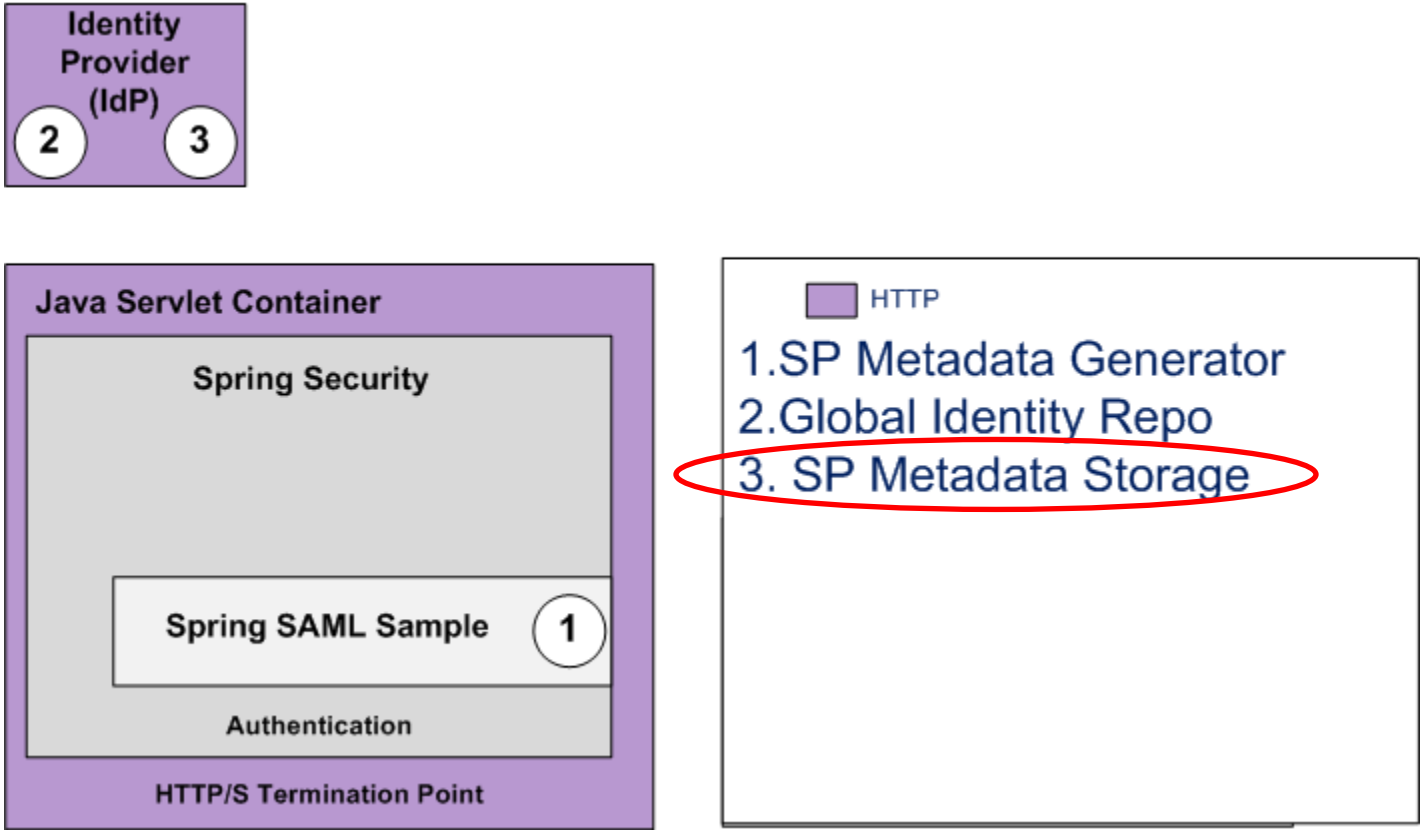
Creating your Identity with SSOCircle (from their website)

For creating your account you need to follow a few steps:

- Register at the SSOCircle SAMLv2.0 Identity Provider
- Provide the required data
- Agree to the Terms of Use
- After successful creation you will receive an email asking for confirmation of your registration. Confirm by navigating to the link supplied in the email.
- Now your account is activated and ready for use.

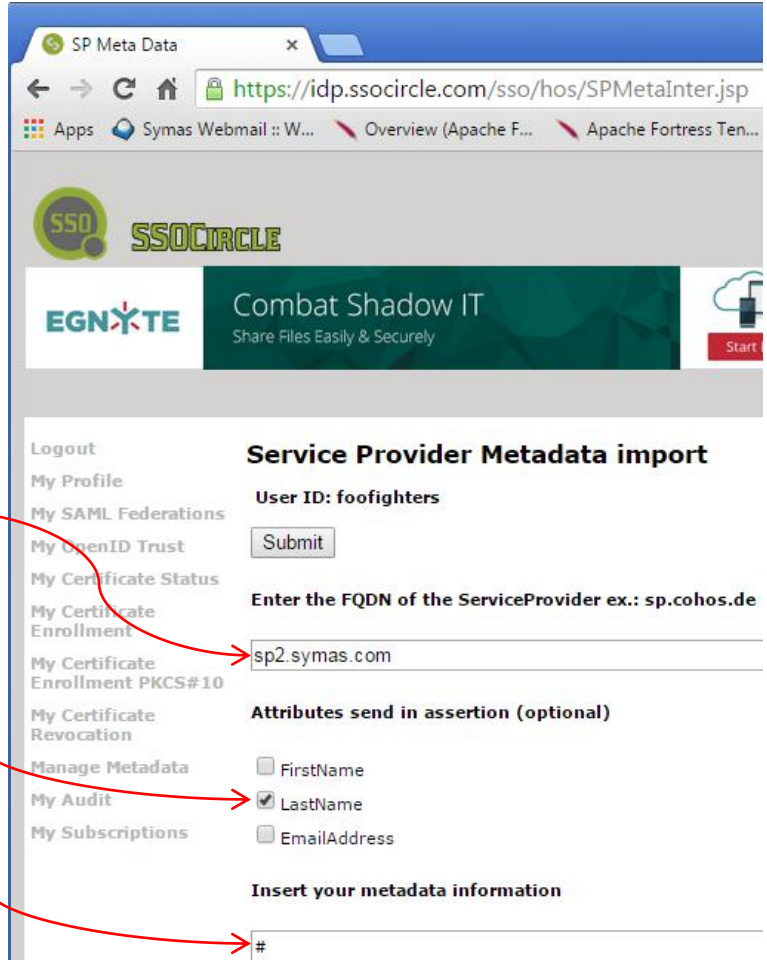
http://www.ssocircle.com/en/portfolio/publicidp/

3. Import Service Provider Metadata into IdP



Import SP Metadata

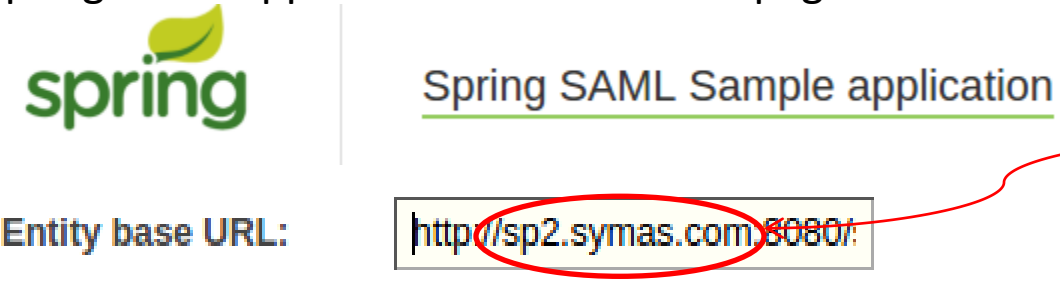
- Logon SSOCircle
- Click on ***Manage Metadata***
- ***FQDN*** must match SP's host name
- Check the ***LastName*** box
- Paste your metadata here



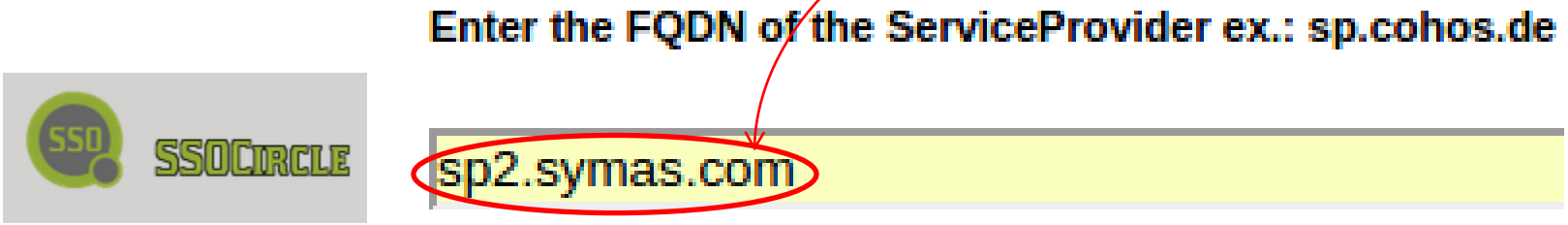
Import SP Metadata Tip

The FQDN matches base url from SP metadata gen step

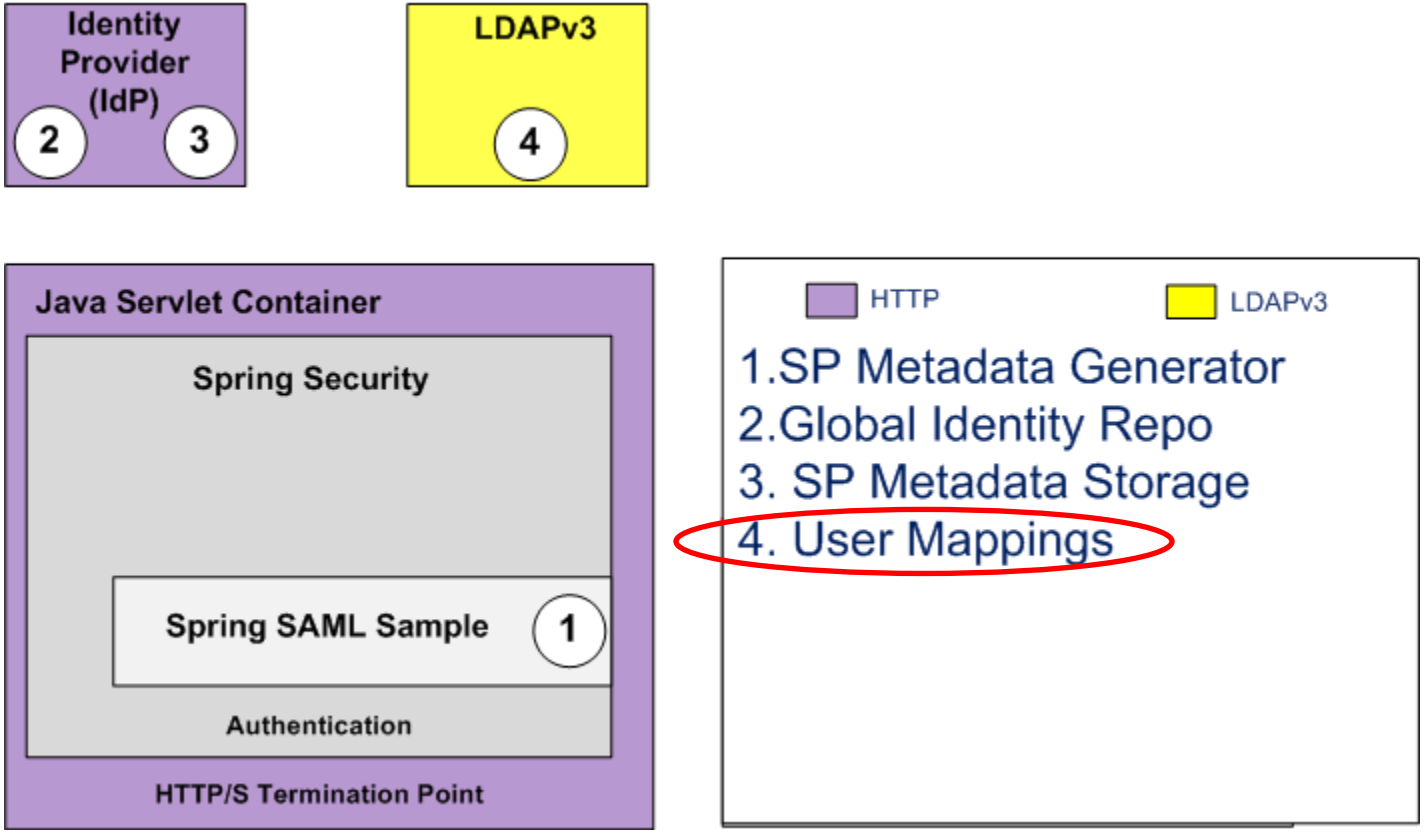
Spring SAML app Metadata Generation page:



SSOCircle Service Provider Metadata Import page:



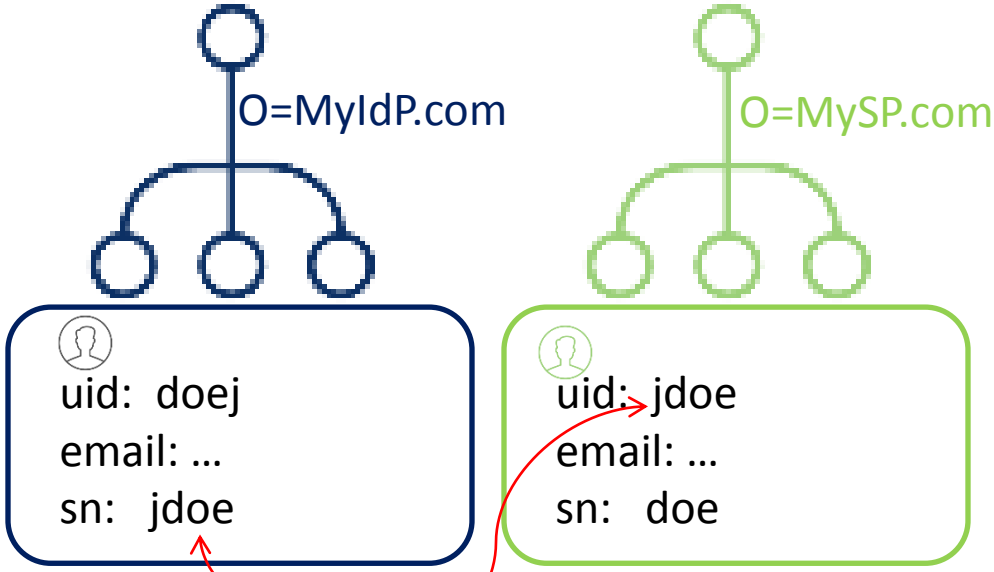
4. IdP and SP User Account Mapping



IdP and SP User Account Mapping

1. Mapping rules are specific to partners.

2. The mapping must be a one-to-one unique pairing.



fortress saml demo maps the sn on the IdP-side with uid field on the SP-side

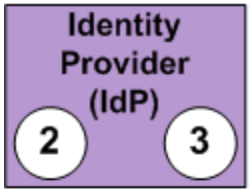
SAML Attribute Statement

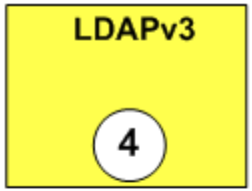
```
<?xml version="1.0" encoding="UTF-8"?><samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol”
Destination="http://sp2.symas.com:8080/fortress-saml-demo/saml/SSO"
…
<saml:AttributeStatement>
…
  <saml:Attribute Name="LastName">
  <saml:AttributeValue …
      xsi:type="xs:string">sam3</saml:AttributeValue>

  </saml:Attribute>
 </saml:AttributeStatement>
…
</samlp:Response>
```

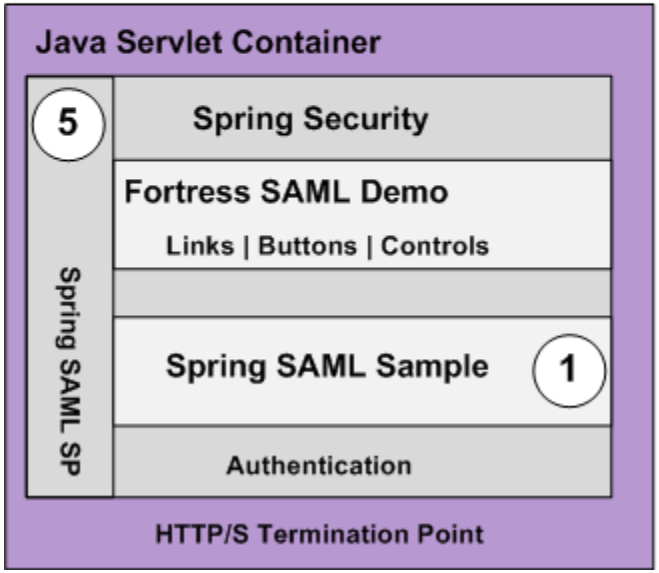
host name entered during SP Metadata import

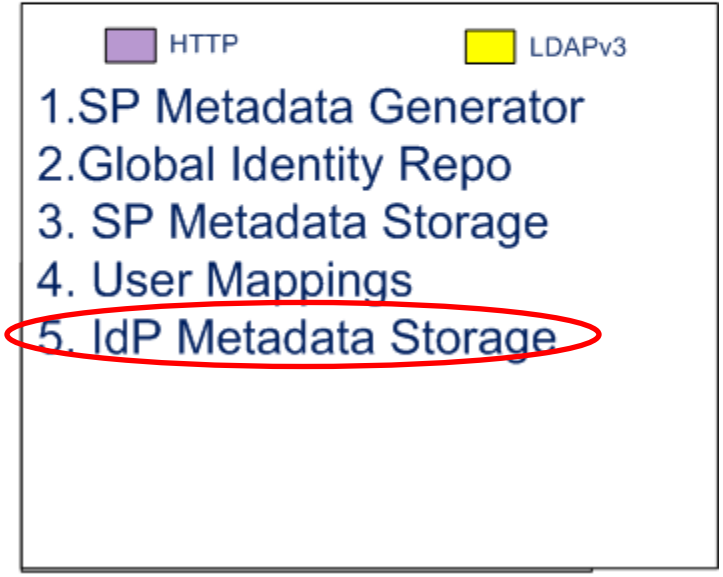
Last Name linked to userid in rbac

5. Load IdP Metadata into Service Provider









Point SP to SAML IdP

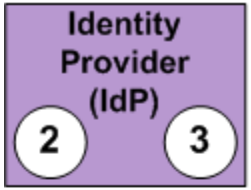
Point to the Identity Provider in securityContext.xml

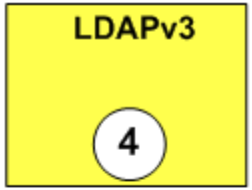
```
<bean id="metadata" class="org.springframework.security.saml.metadata.CachingMetadataManager">
 <constructor-arg>
  <list>
   <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider">
    <constructor-arg>
     <value type="java.lang.String"
```

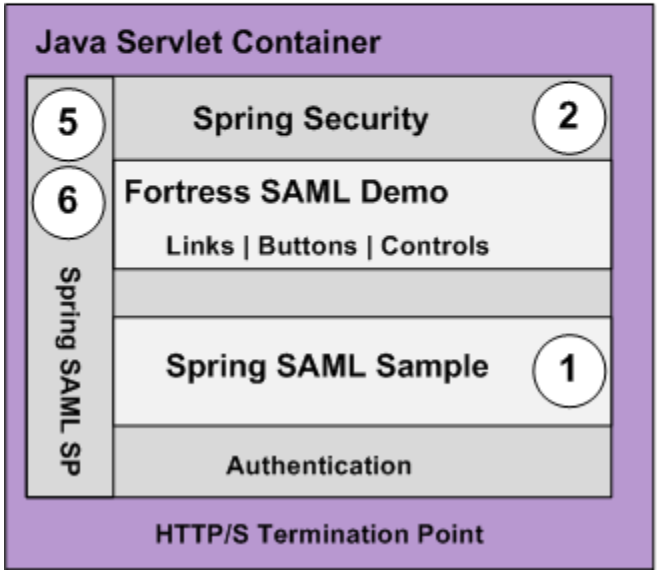
http://idp.ssocircle.com/idp-meta.xml

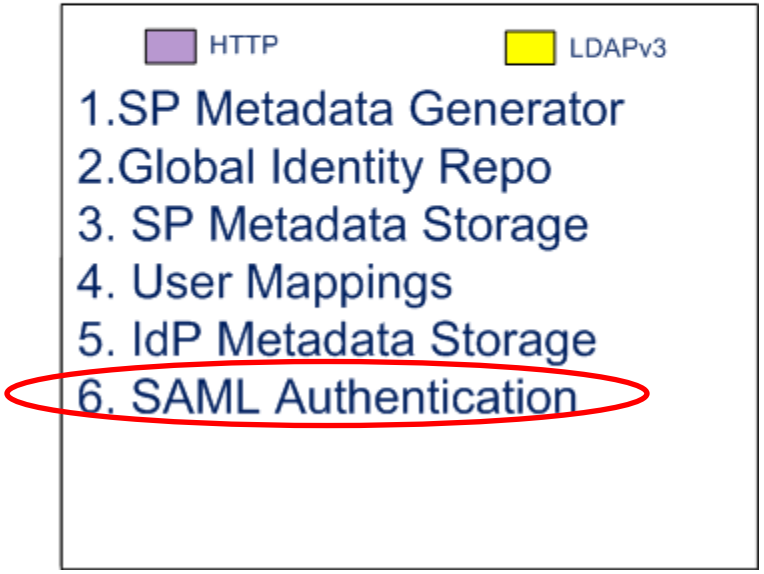
```
     </value>
    </constructor-arg>
    <constructor-arg>
     <value type="int">5000</value>
     </constructor-arg>
     <property name="parserPool" ref="parserPool"/>
   </bean>
  </list>
 </constructor-arg>
</bean>
```

6. Enable Spring SAML Authentication









Enable Spring SAML Security

Add dependencies to pom:

```
<dependency>
  <groupId>org.springframework.security.extensions</groupId>
  <artifactId> spring-security-saml2-core </artifactId>
  <version>1.0.1.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId> spring-security-config </artifactId>
  <version> 3.1.2.RELEASE* </version>
  <scope>compile</scope>
</dependency>
```

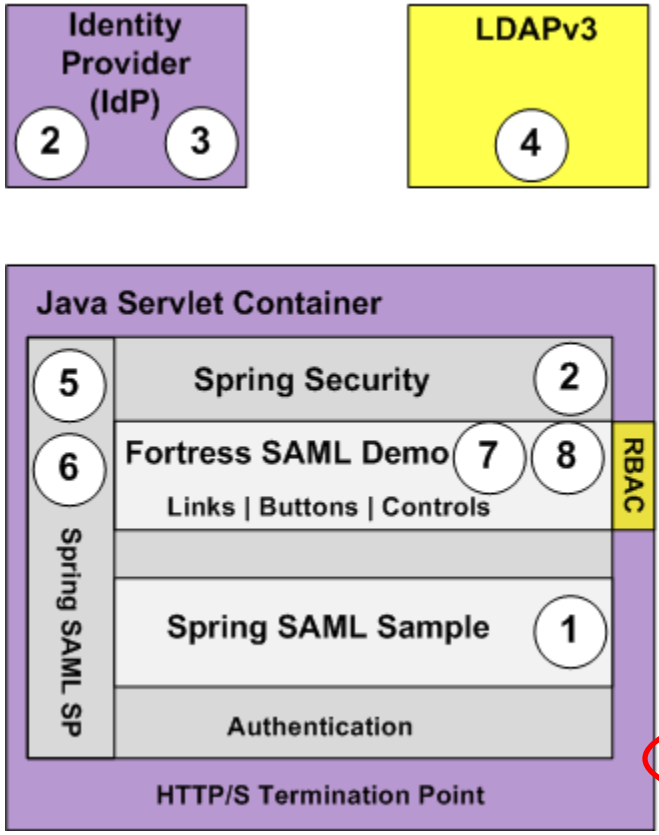
* backlog item

Enable SAML Authentication Filters

In the securityContext.xml

```
<security:http entry-point-ref="samlEntryPoint" use-expressions="false">

  <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
      <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
  <security:custom-filter before="FIRST" ref="metadataGeneratorFilter"/>
  <security:custom-filter after="BASIC_AUTH_FILTER" ref="samlFilter"/>
</security:http>

<bean id="samlFilter" class="org.springframework.security.web.FilterChainProxy">
    <security:filter-chain-map request-matcher="ant">
    <security:filter-chain pattern="/saml/login/**" filters="samlEntryPoint"/>
    <security:filter-chain pattern="/saml/logout/**" filters="samlLogoutFilter"/>
    <security:filter-chain pattern="/saml/metadata/**" filters="metadataDisplayFilter"/>
    <security:filter-chain pattern="/saml/SSO/**" filters="samlWebSSOProcessingFilter"/>
    <security:filter-chain pattern="/saml/SSOHoK/**" filters="samlWebSSOHoKProcessingFilter"/>
    <security:filter-chain pattern="/saml/SingleLogout/**" filters="samlLogoutProcessingFilter"/>
  </security:filter-chain-map>
</bean>
```

7. Setup RBAC Policy Decision Point



Enable RBAC Policy Decision Point

```
<dependency>
  <groupId>
    org.apache.directory.fortress
  </groupId>
  <artifactId>
      fortress-realm-impl
  </artifactId>
  <version>1.0</version>
</dependency>
```

Identity Propagation SAML->RBAC

1. Spring SAML filter creates security principal based on attributes found in the SAML attribute assertion.
2. Web app calls to get the Security Principle from Spring:

```
ServletContext.getUserPrinciple();
```

standard API call

3. Web app parses the attributes contained within principal :

```
uid=getSurName((SAMLCredential)principal.getCredentials());
```

4. Web app creates a new RBAC session using attribute(s) pulled from the principal :

```
j2eePolicyMgr.createSession( new User( uid ), true );
```

isTrusted

5. Web app pushes RBAC session into HTTP session.

Apache Fortress Saml Demo

- Three Pages
- Each has buttons controlled by RBAC permissions.
- One role per page.
- Users may be assigned to one or more roles.

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True

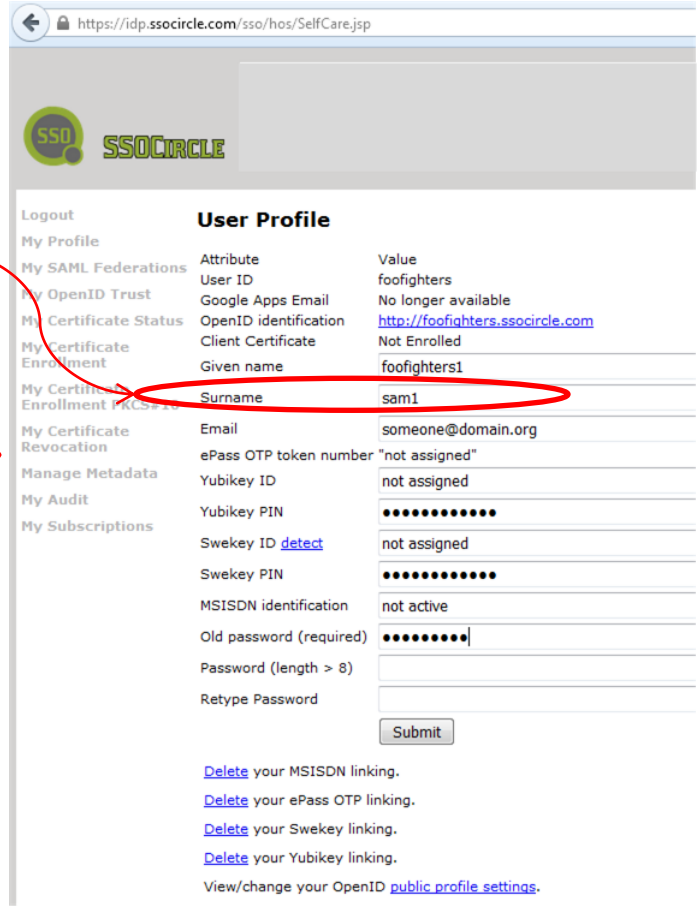
To Change Demo Users

uid=sam1,ou=People,dc=example,dc=com

DN: uid=sam1,ou=People,dc=example,dc=com

Attribute Description	Value
objectClass	***extensibleObject (auxiliary)***
objectClass	***ftMods (structural)***
objectClass	***ftProperties (structural)***
objectClass	***ftUserAttrs (structural)***
objectClass	***inetOrgPerson (structural)***
objectClass	***organizationalPerson (structural)***
objectClass	***person (structural)***
objectClass	***top (abstract)***
cn	**Sam One**
sn	**One**
description	Fortress SAML Demo User 1
displayName	Sam One
ou	org.samlsample.users
uid	sam1
userPassword	SSHA hashed password
ftCstr	*sam1$0$$$$$$$*
ftId	*a59bf210-7101-4bb9-b089-9205d594d108*
ftProps	*init:*
ftRA	*samRole1*

Change Surname field in SSO Circle Profile to use different rbac users.



Apache Fortress SAML Demo

- https://github.com/shawnmckinney/fortress-saml-demo

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True

Cloud Native Architecture

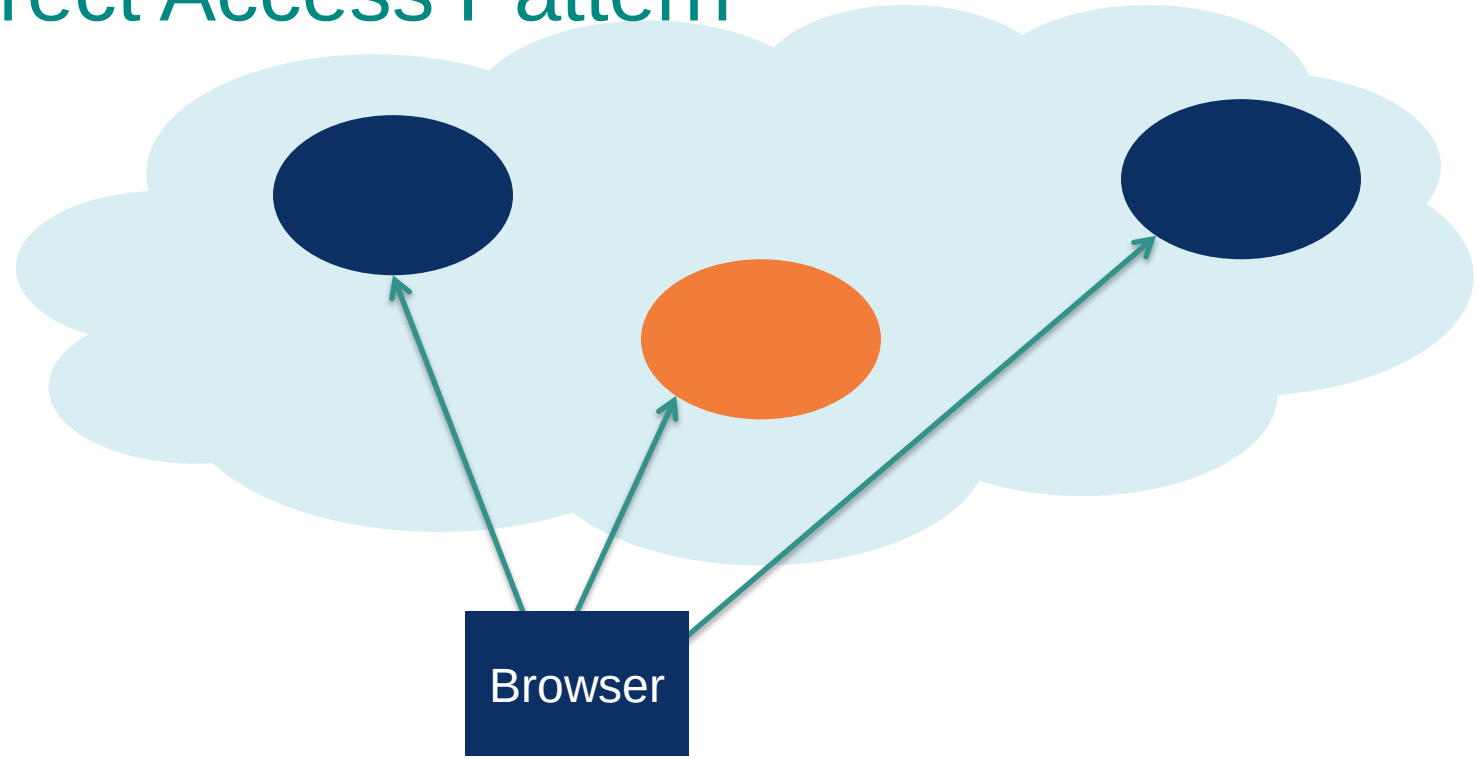
- Q: What does “Cloud Native” really mean?
- A: 12 Factor Apps?
- Yes, but…the 12 Factor guidance is ***essentially silent*** on security considerations!
  - The security properties are implied.
  - You need to “read between the lines”.

Cloud Native Architecture

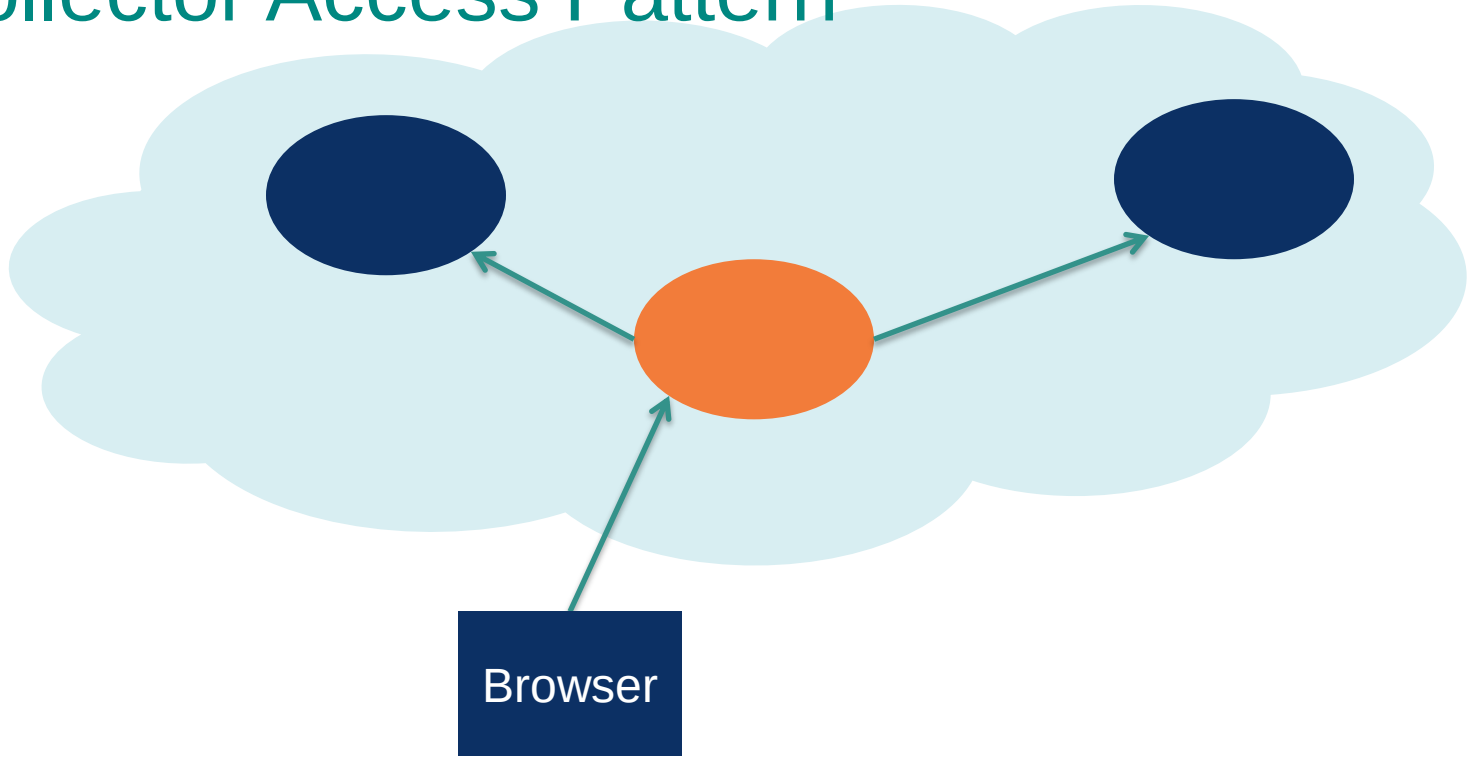
- Assume HTTP everywhere
- µSvcs are highly cohesive
- µSvcs are loosely coupled
- Expectation of flexible interoperability of µSvcs
- Potential for (frequent) Re-composition of µSvcs
- Everything is late-bound
- Examples…

Pivotal

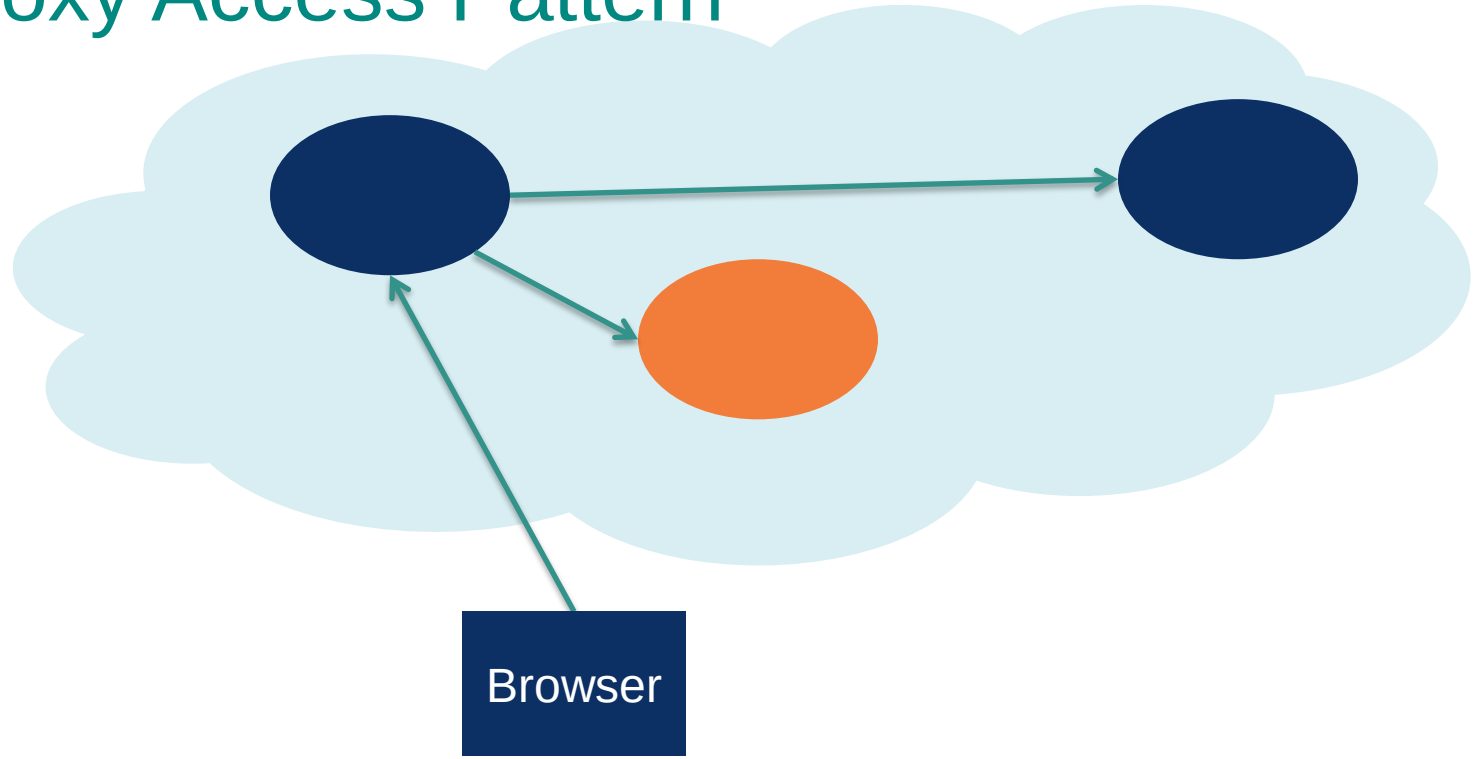
Direct Access Pattern



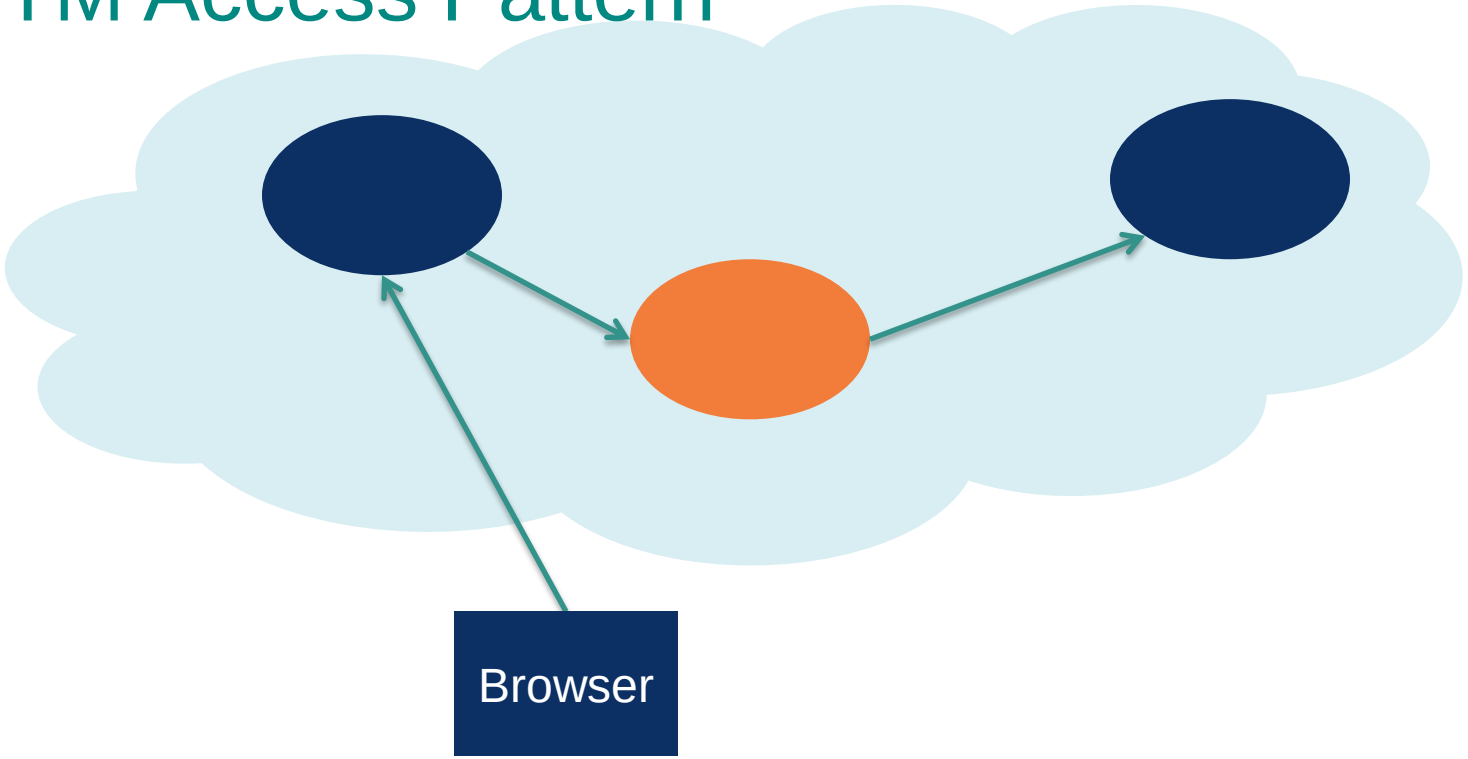
Collector Access Pattern



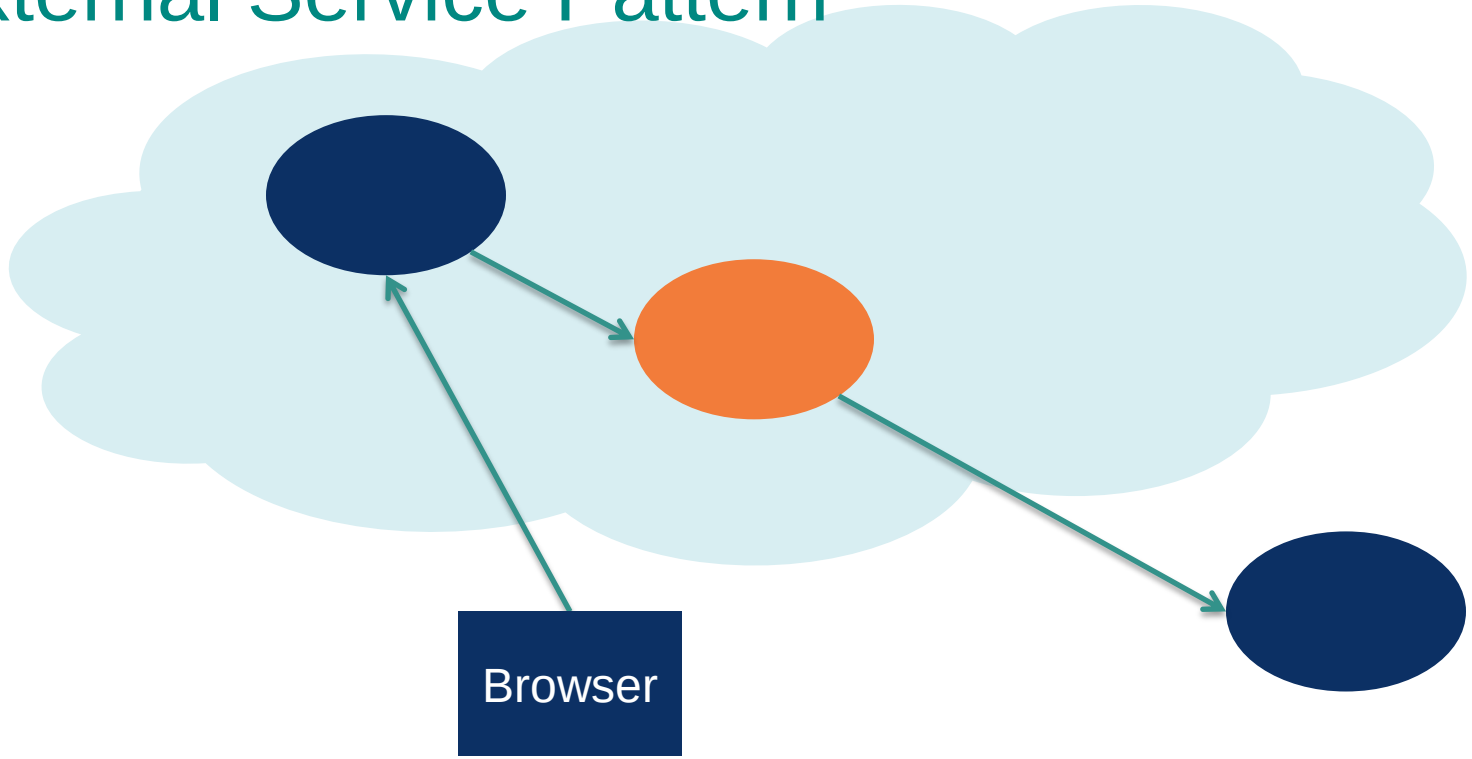
Proxy Access Pattern



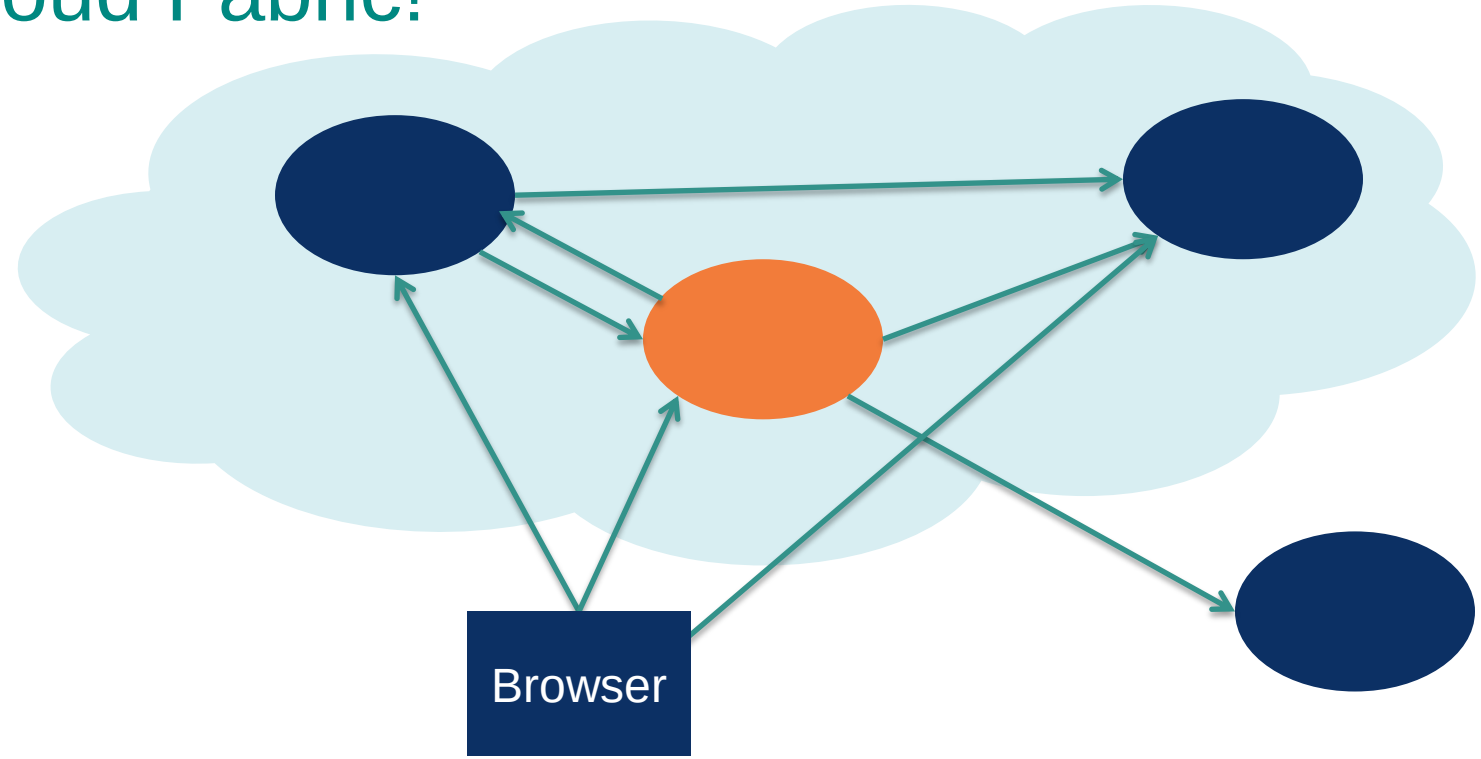
MITM Access Pattern



External Service Pattern



Cloud Fabric!



Security for Cloud Native Applications

- Deployment agility applies to the security aspects too!
- Loose coupling, and late binding to your security services.
- Your code may be required to fulfill different security roles.

Pivotal

TL;DR:

Use SAML2 and OAuth2.

Pivotal

Why OAuth2?

1. Necessary to be a “Full Security Stack” Developer
   - OAuth2 is an essential core competency
2. Widely adopted
   - Standards Track IETF RFC 6749, 6750, 7523…et. al.
3. The MEPs enable all naturally occurring security patterns
   - Native cloud, or forklifted
4. A key part of Spring Cloud Security
   - SSO, and authorization token exchange

Pivotal

Why is OAuth2 so hard to Understand?

- No, you're not dumb
  - ***OAuth2 really is confusing***.
- It's confusing because of it's flexibility
- In fact, it's actually as simple as it can be.

Pivotal

OAuth2 Grant Types

- There are 4 ways to run the AuthZ protocol.
- These profiles are called “Grant Types”
  1. Authorization Code Grant
  2. Implicit Grant
  3. ~~Resource Owner Password Grant~~
  4. Client Credentials Grant

OAuth2 Architectural Roles

1. Resource Owner – the end user.
2. Resource Service – the target.
3. Client Service – the MITM, to be authorized.
4. Authorization Service – the TTP token issuer.

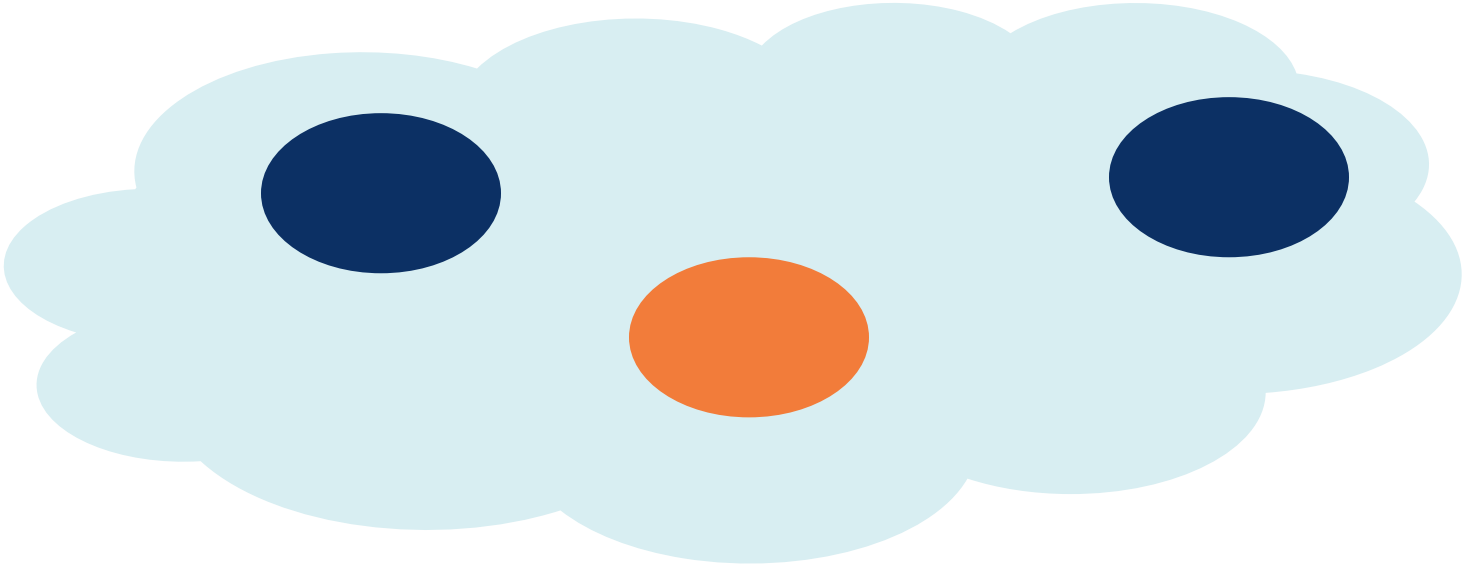
Pivotal

Understanding OAuth2 Scopes

- Scopes represent the authorization in OAuth2.
- When requesting a token, clients specify scopes.
- Scopes are not groups, or roles.
- Scopes are just strings, tags
  - A private contract between RS and CL
- AZ server does not interpret
- User may allow/deny the scope(s)

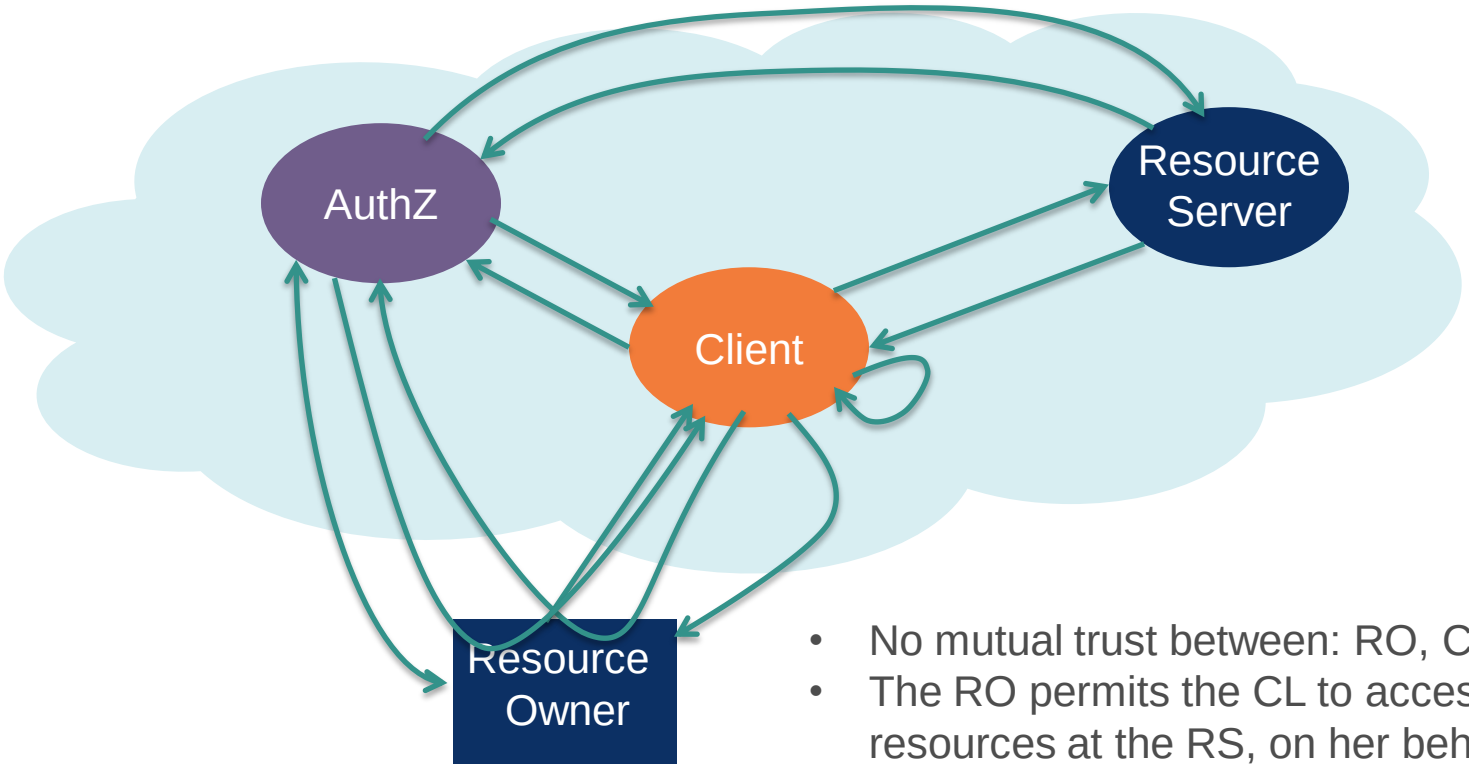
Pivotal

An Authorization Code Grant



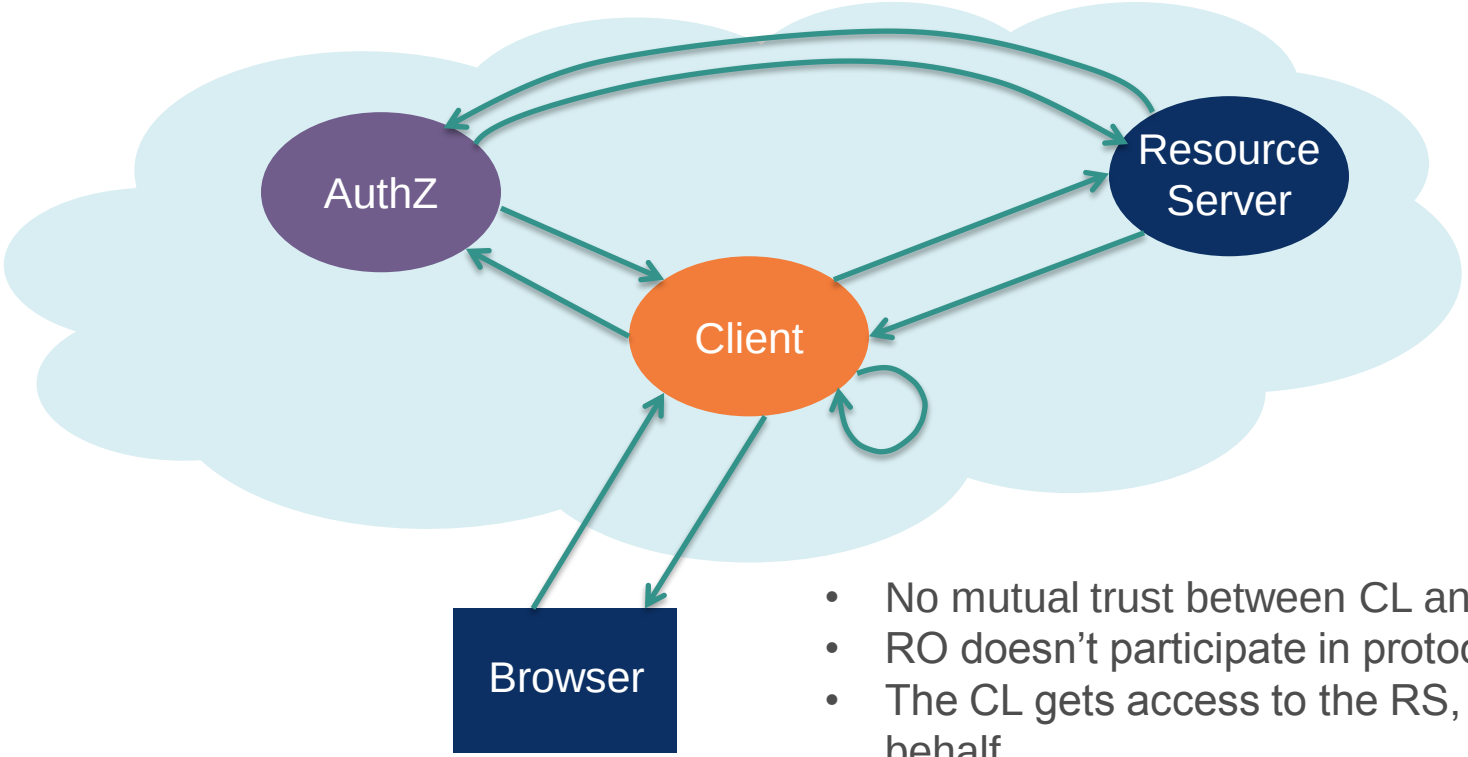
Browser

An Authorization Code Grant



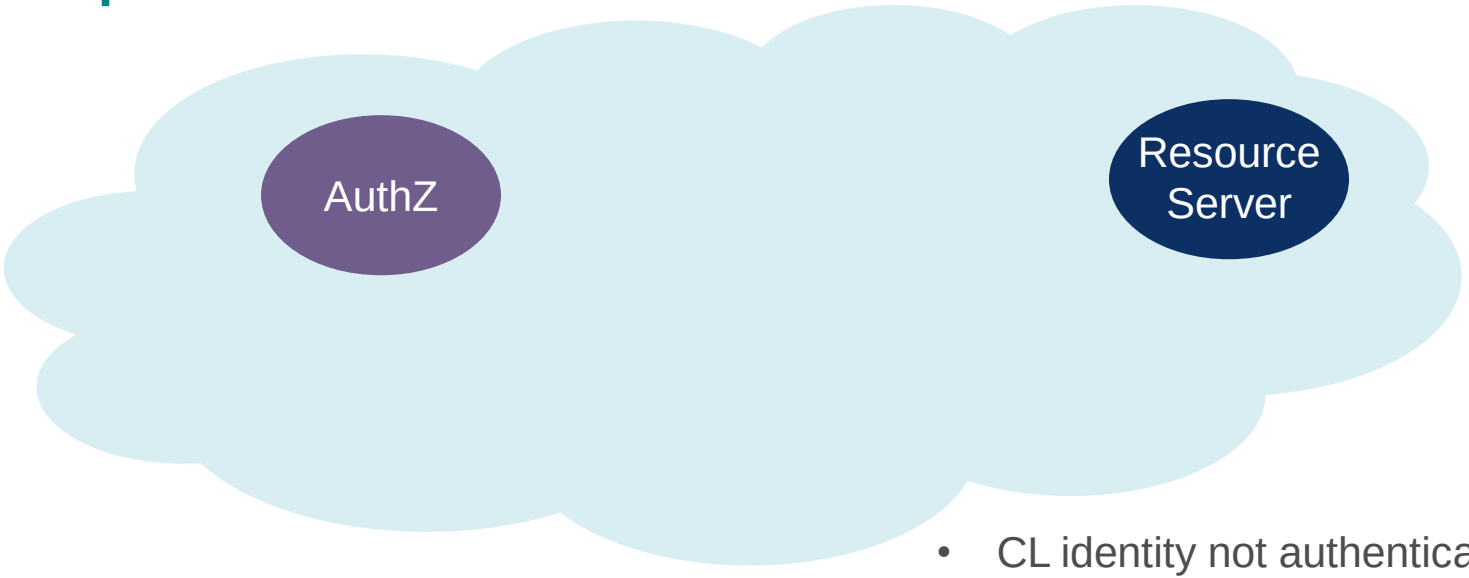
- No mutual trust between: RO, CL, RS.
- The RO permits the CL to access her resources at the RS, on her behalf.

A Client Credentials Grant



- No mutual trust between CL and RS.
- RO doesn't participate in protocol.
- The CL gets access to the RS, on it's own behalf.

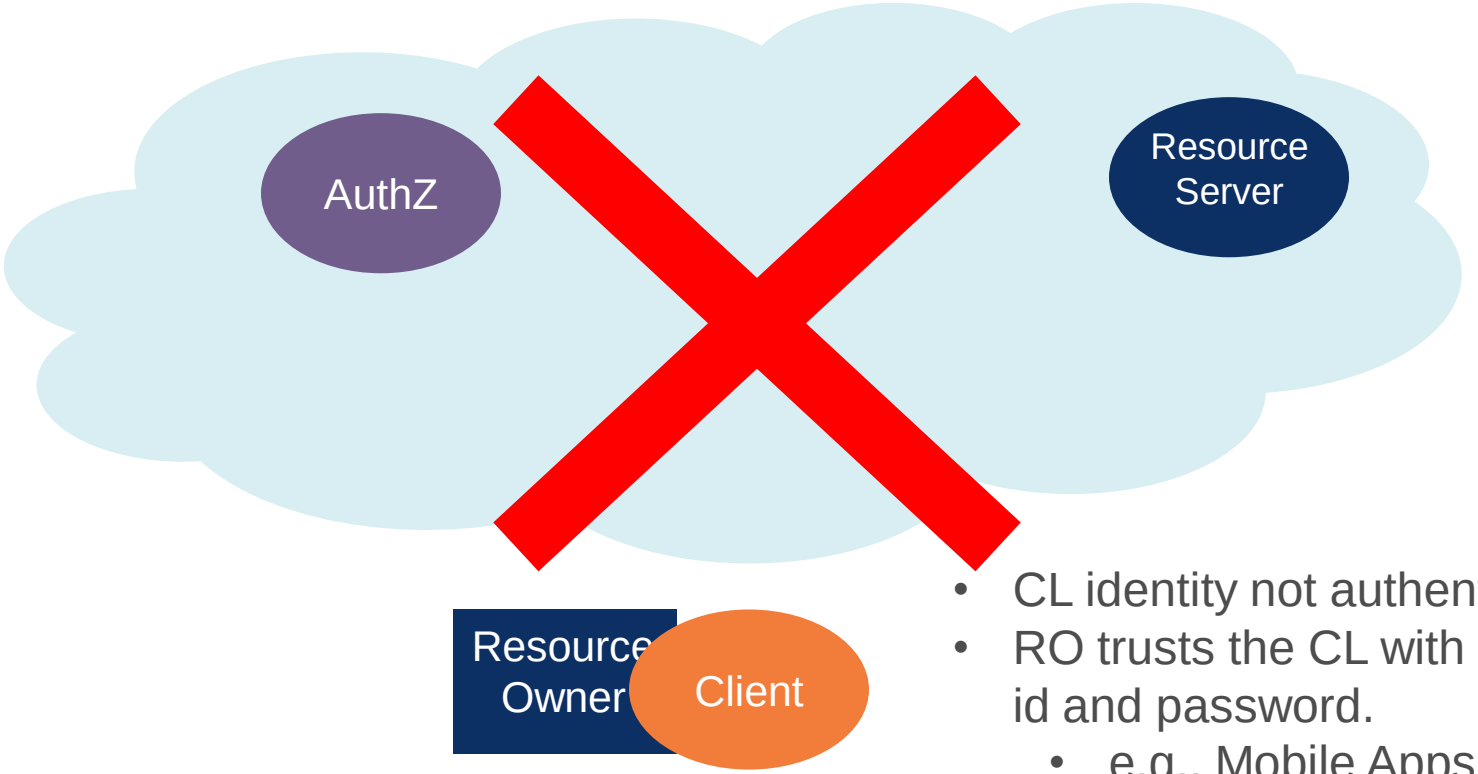
An Implicit Grant



- CL identity not authenticated.
- RO directly obtains token for CL id, and passes it on to RS.
  - e.g., JavaScript Application
- CL trusts RO to not misuse token

Pivotal

A Resource Owner Credentials Grant



- CL identity not authenticated.
- RO trusts the CL with her user id and password.
  - e.g., Mobile Apps

Pivotal

Making OAuth2 Easier

Spring Security

Spring Boot

OAuth2, With spring-security-oauth

- Annotation driven OAuth2 participant roles:
  - @EnableResourceServer
  - @EnableAuthorizationServer
  - @EnableOAuth2Client
  - @EnableSSOClient

Pivotal

OAuth2, With spring-security-oauth

- @EnableResourceServer
  - Implement class like *ResourceServerConfigurer()*
- @EnableAuthorizationServer
  - Implement class *AuthorizationServerConfigurer()*

OAuth2 Client Request

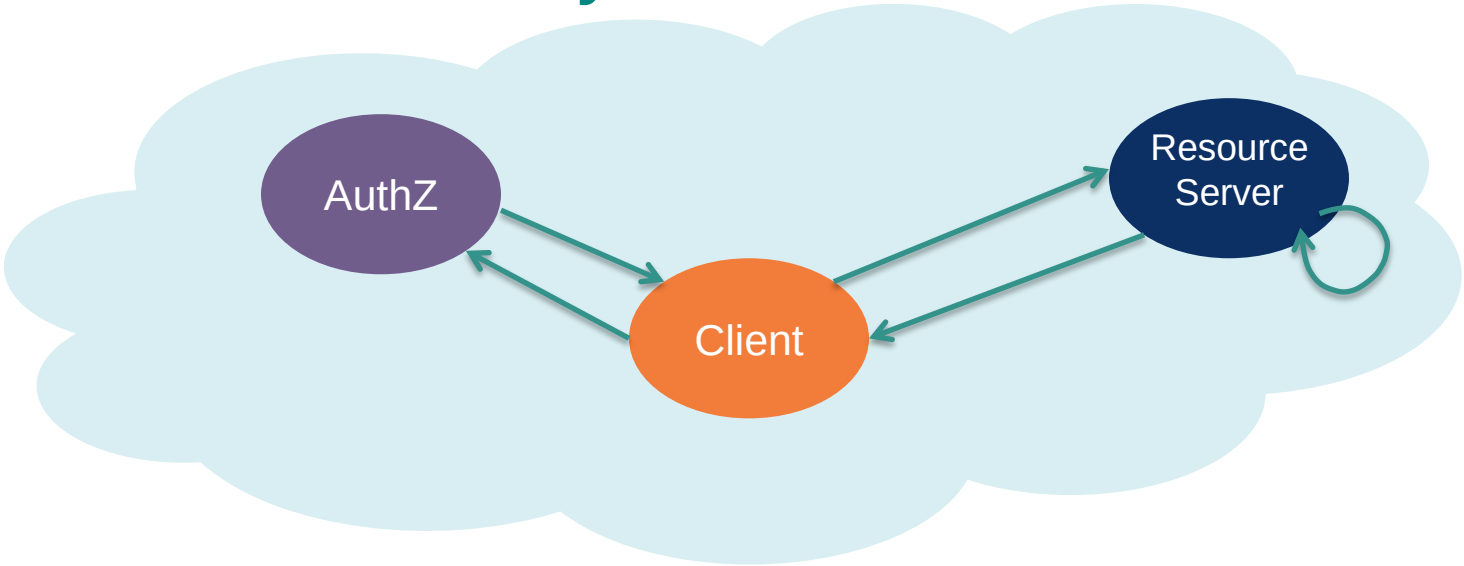
- Client should implement configuration Class
  - OAuth2RestTemplateConfigurer()
- Client makes authorized HTTP requests via
  - OAuth2RestTemplate()

Demo: Client Credentials Grants

- 2-Party
  - Apache Fortress Demo Page 4 – the Client Role
  - A back-end data service – does both RS and AZ roles.
- 3-Party
  - CLI Application – the Client role
  - A back-end JPA service – the Resource Server role
  - UAA – the Authorization Server role

Pivotal

Demo: Three-Party Client Credentials Grant

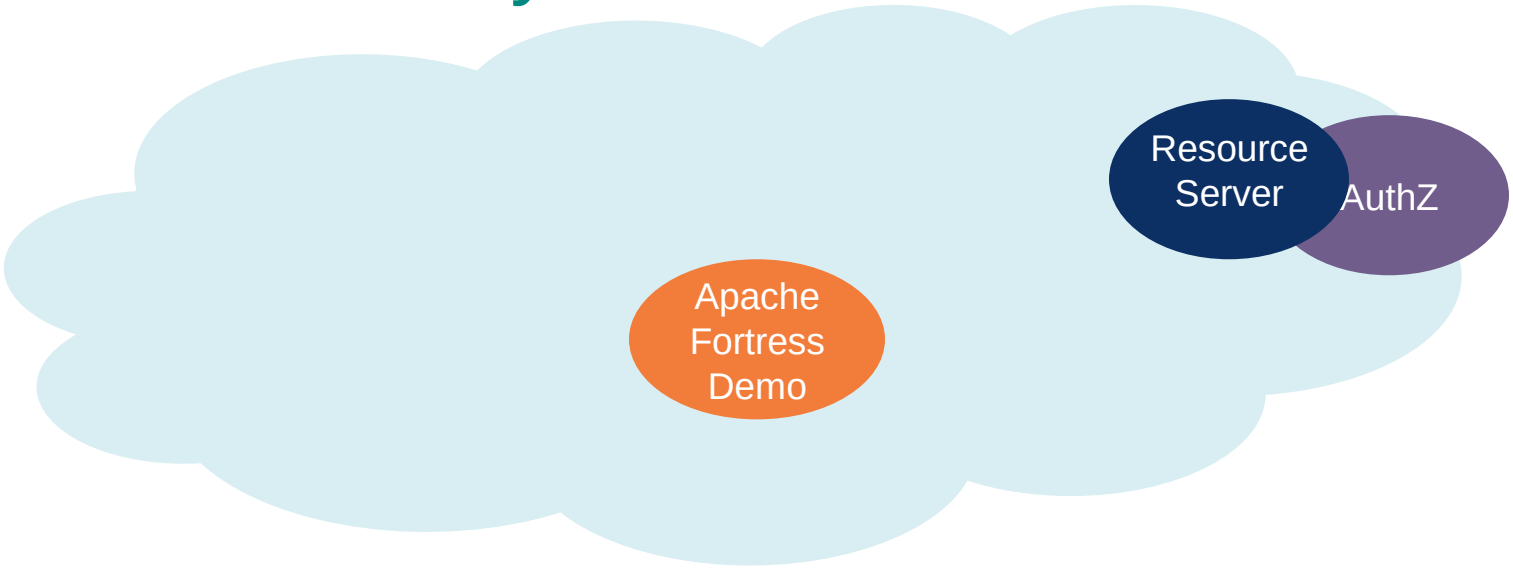


Browser

- No mutual trust between CL and RS.
- RO doesn't participate in protocol.
- The CL gets access to the RS, on it's own behalf.

Pivotal

Demo: Two-Party Client Credentials Grant



- Same as previous case, except the RS and AZ are one and the same.

OAuth2 in a Nutshell

Grant Type	Description	Active Roles	Combined Roles	Comments
Authorization Code	AZ authorizes the CL to act at the RS, on behalf of the RO.	AZ, CL, RO, RS	None	• All roles active. • Nobody trusts anybody. • The AZ Server is the PEP. • The RO is the PDP.
Implicit	AZ authorizes the RO to act at RS, on behalf of the CL.	AZ, CL, RS	CL → RO	• CL trusts RO to handle its credentials. • No intermediate code issued. • Token issued directly, and available to RO. • No authentication of CL identity.
Resource Owner Credentials	AZ authorize CL to act as the RO at the RS.	AZ, RO, RS	RO → CL	• RO trusts CL to handle its credentials. • No authentication of CL identity.
Client Credentials	AZ authorizes CL to access RS on its own behalf.	AZ, CL, RS	RO == NULL	• RO identity is completely out of scope.

Pivotal

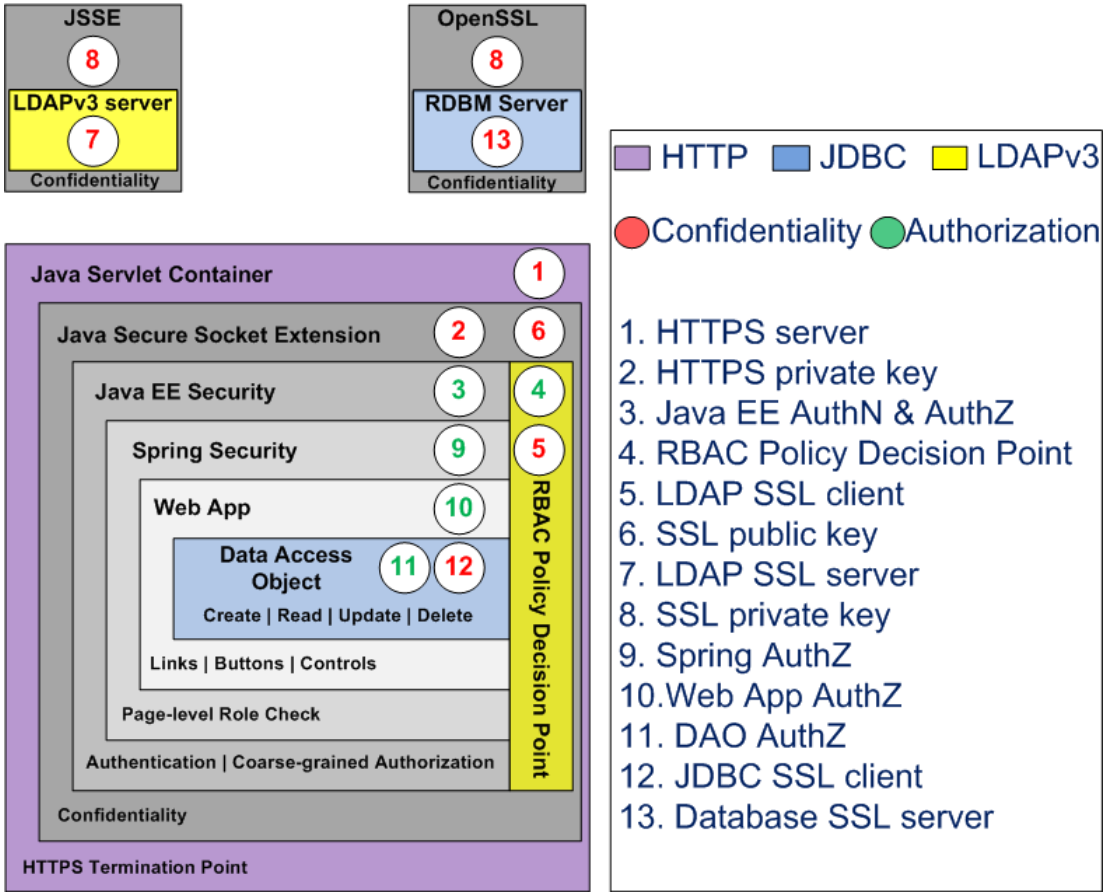
SAML2 and OAuth2: Perfect Together

- SAML2 enables enterprise-centric SSO
  - IdP usually in different trust domain; enterprise provisioning.
- Both protocols enable apps to use ephemeral tokens
  - Rather than synchronous access to a statically configured PDP.
- OAuth2 is a consistent, but flexible approach to µSvcs AuthZ
  - 4 different MEP supported.
- OAuth2 enables user-centric SSO (OpenID Connect)
  - IdP may be in different trust domain; user may self-provision.

Pivotal

Tutorial #2

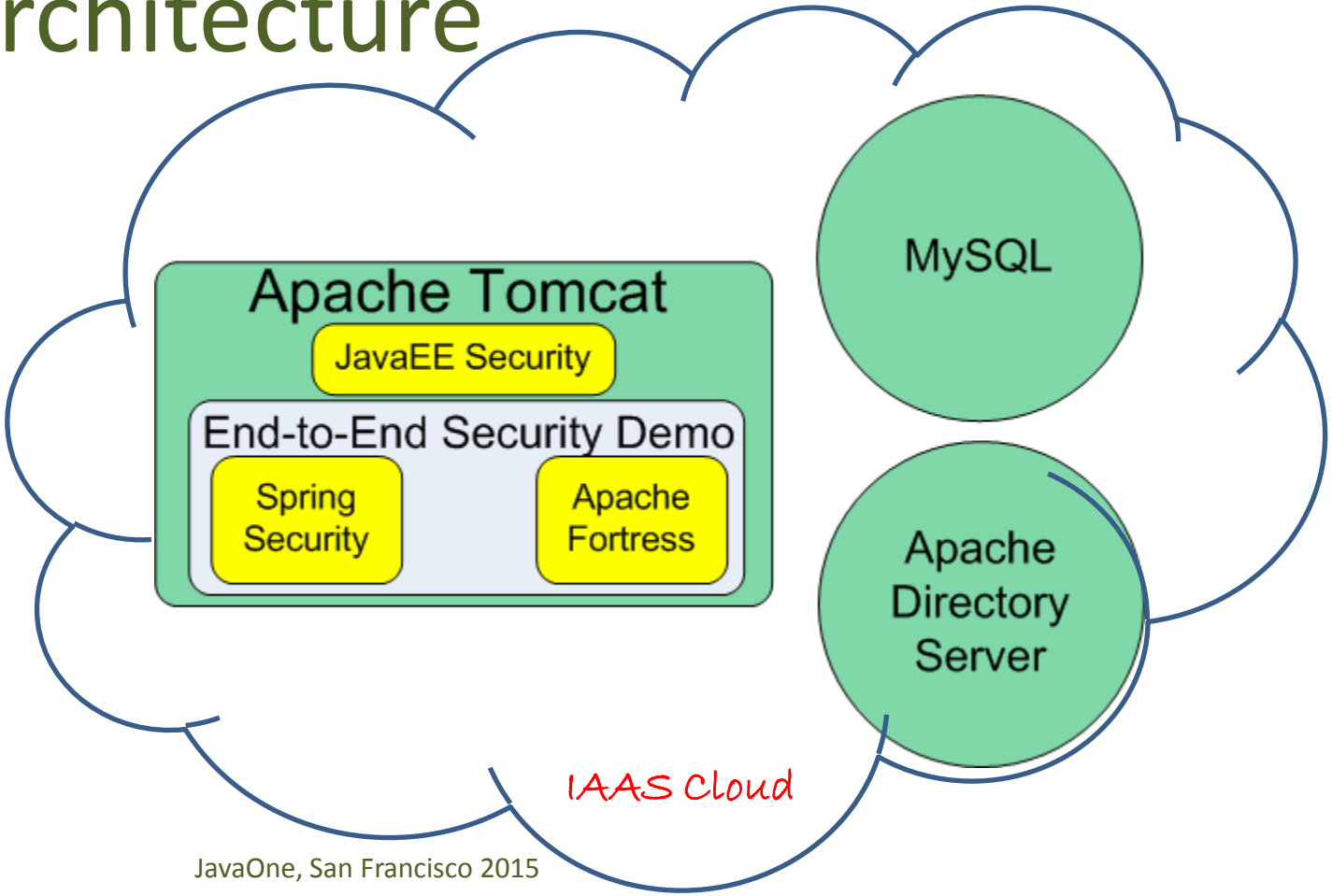
Apache Fortress End-to-End Security Tutorial



http://iamfortress.net/2015/02/16/apache-fortress-end-to-end-security-tutorial/

symas

System Architecture



Security Layers with Fortress Demo

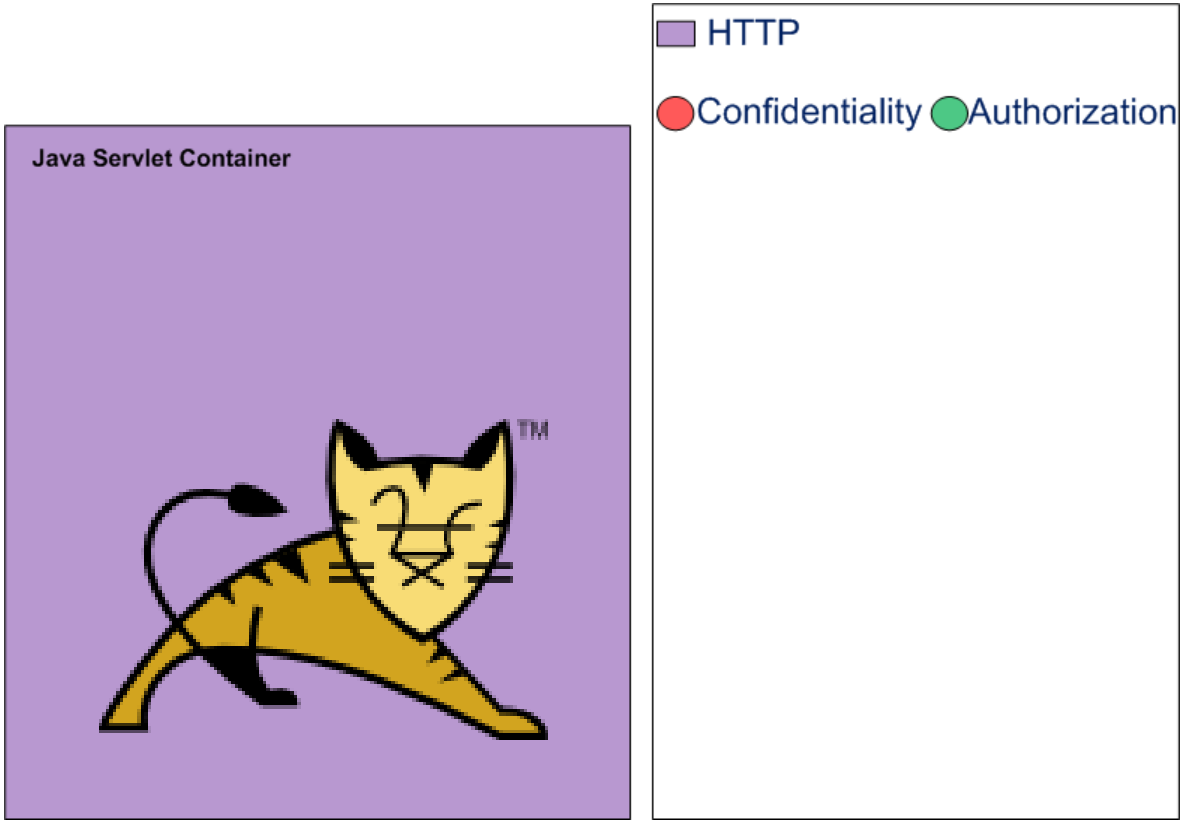
1. JSSE ← Confidentiality and Itegrity
2. Java EE Security ← authN and coarse-grained authZ
3. Spring Security ← medium-grained authZ
4. Web App Framework ← fine-grained authZ
5. Database Functions ← very fine-grained authZ

Two Areas of Access Control

1. Java EE and Spring Role Declarative checks

2. RBAC Permission Programmatic checks

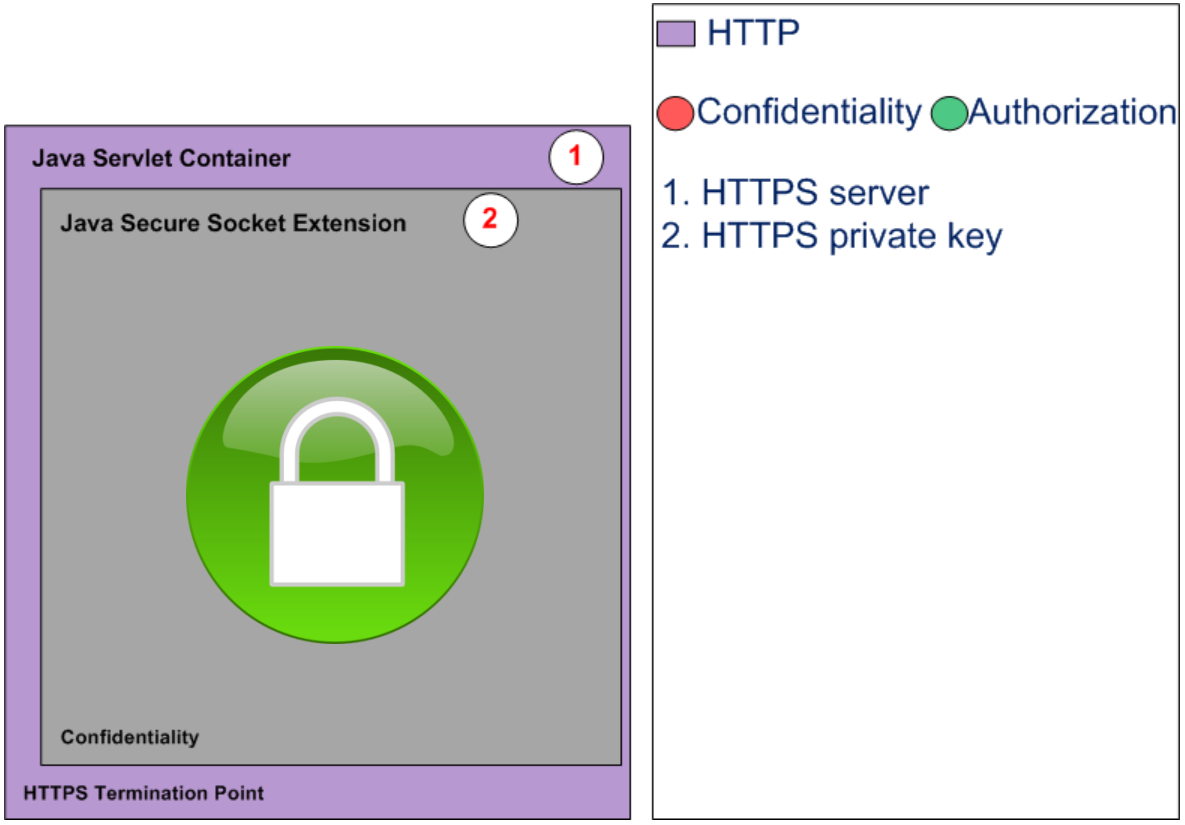
Start with Tomcat Servlet Container



1 & 2. Enable HTTPS

1. Update the Server.xml

2. Add private key

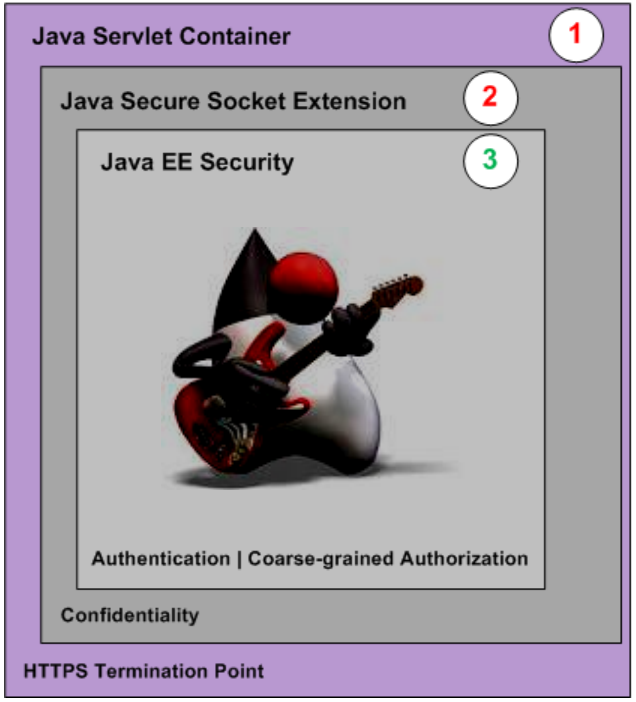


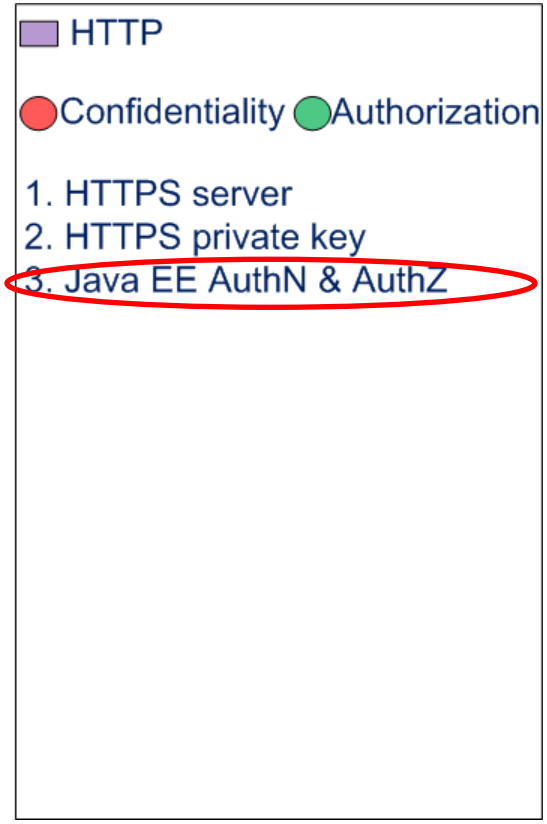
3. Enable Java EE Security

a. Drop the proxy jar

b. Update web.xml

c. Add context.xml





Enable Java EE Security Realm

Drop the Fortress Realm Proxy Jar in Tomcat's lib folder

```
[root@IL1SCOLSP102 lib]# pwd
/usr/local/tomcat7/lib
[root@IL1SCOLSP102 lib]# ls -l fortress*
-rw-r--r-- 1 root root 15119 Aug 29 12:37 fortress-realm-proxy-1.0-RC41-SNAPSHOT.jar
[root@IL1SCOLSP102 lib]#
```

Fortress Realm Proxy loads implementation jars from the app via a URLClassloader 'trick'.

```
[root@IL1SCOLSP102 lib]# pwd
/usr/local/tomcat7/webapps/apache-fortress-demo/WEB-INF/lib
[root@IL1SCOLSP102 lib]# ls -l fortress*
-rw-r--r-- 1 root root 502112 Aug 30 06:55 fortress-core-1.0-RC41-SNAPSHOT.jar
-rw-r--r-- 1 root root  22005 Aug 29 12:20 fortress-realm-impl-1.0-RC41-SNAPSHOT.jar
-rw-r--r-- 1 root root 789927 Aug 29 12:40 fortress-web-1.0-RC41-SNAPSHOT-classes.jar
[root@IL1SCOLSP102 lib]#
```

Enable Java EE Security Realm

Add to App's Web.xml

```
<security-constraint>
  <display-name>My Project Security Constraint</display-name>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <url-pattern>/wicket/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>DEMO2_USER</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>MySecurityRealm</realm-name>
  <form-login-config>
    <form-login-page>/login/login.html</form-login-page>
```

1. Java EE container protects this URL Automatically.

2. All users must have this role to gain entry.

3. Route un-authN requests to my form.

https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/webapp/WEB-INF/web.xml

Enable Java EE Security Realm

Add context.xml to META-INF folder:

```
<Context reloadable="true">
 <Realm className=
   "org.apache.directory.fortress.realm.tomcat.Tc7AccessMgrProxy"

   defaultRoles="ROLE_DEMO2_SUPER_USER,DEMO2_ALL_PAGES,
                 ROLE_PAGE1, ROLE_PAGE2, ROLE_PAGE3"
   containerType="TomcatContext"
   realmClasspath=""
 />
</Context>
```

Fortress Tomcat Realm engaged

Activate these roles into RBAC session.

impl jars will be found in app's war

https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/resources/META-INF/context.xml

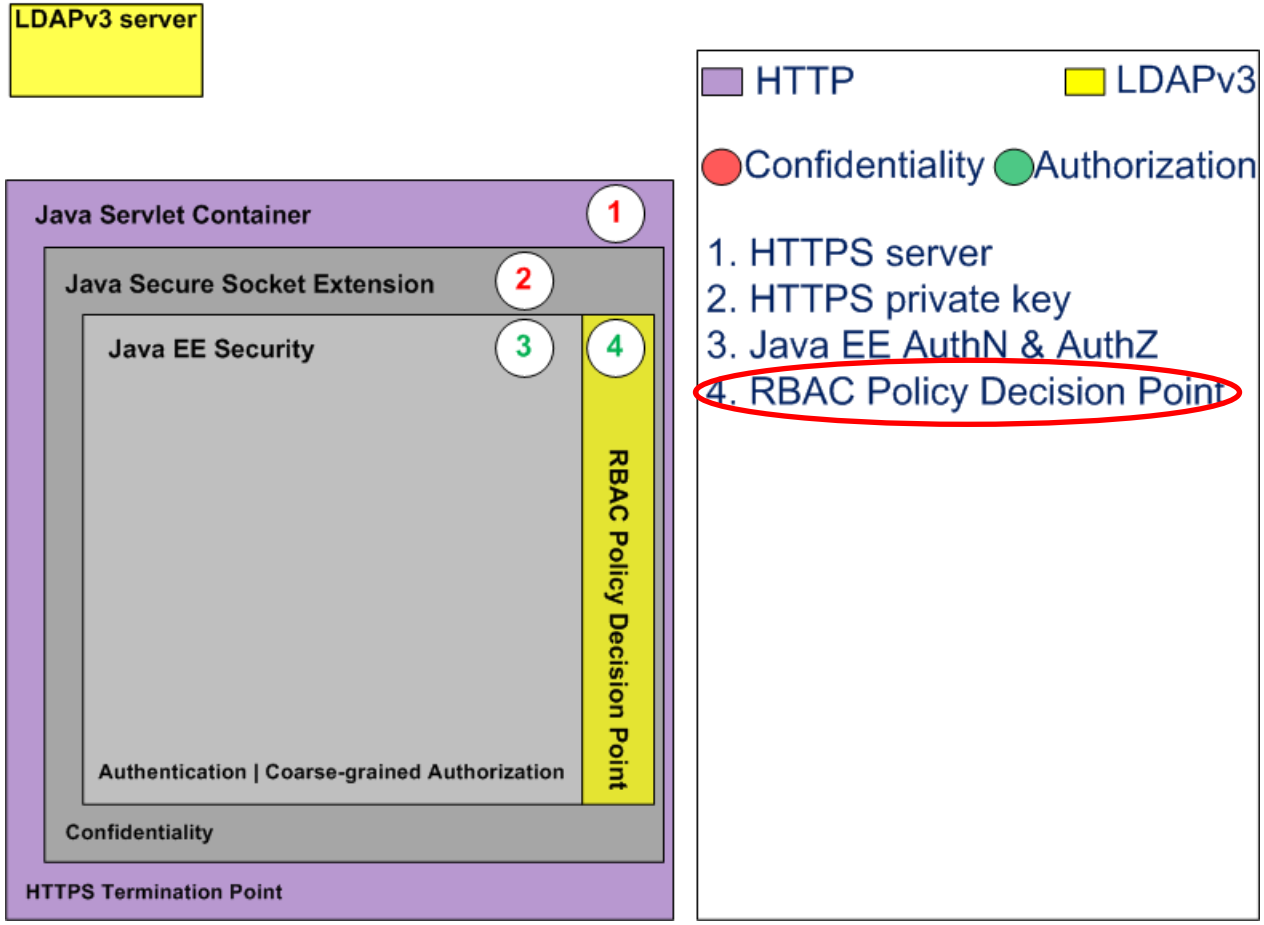
4. Setup RBAC PDP

Policy Decision Point

a. Install

b. Configure

c. Use



Install Fortress RBAC Policy Decision Point

Download and install Apache Directory Fortress

The Fortress ten minute guide provides instructions:

1. Apache Directory Server
2. Apache Directory Studio
3. Apache Fortress Core
4. Apache Fortress Realm
5. Apache Fortress Web
6. Apache Fortress Rest

https://directory.apache.org/fortress/gen-docs/latest/apidocs/org/apache/directory/fortress/core/doc-files/ten-minute-guide.html

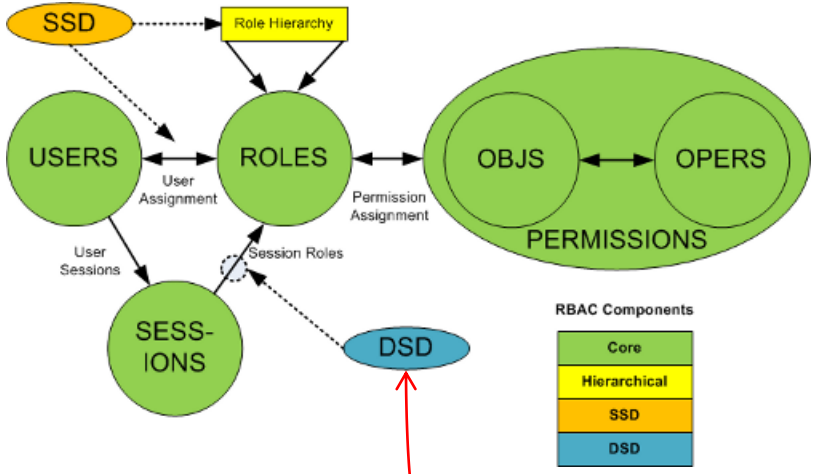
Use ANSI RBAC INCITS 359 Specification

RBAC0:
- Users, Roles, Perms, Sessions

RBAC1:
- Hierarchical Roles

RBAC2:
- Static Separation of Duties

RBAC3:
- Dynamic Separation of Duties



Today we demo this

Dynamic Separation of Duty Use Case

Set Name	Role Name	Type	Cardinality
Teenager	Videogame	Dynamic	2
	Movie	Pick any one activity	
	Homework		

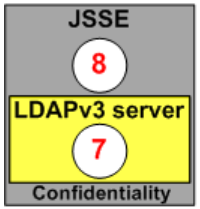
Static Separation of Duty Use Case

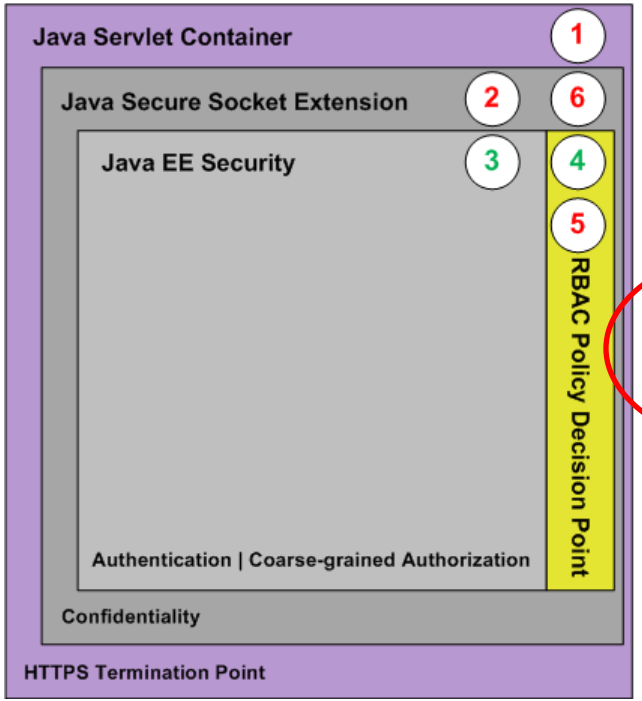
Set Name	Role Name	Type	Cardinality
Teenager	Football	Static	3
	Band	Enroll in any two activities	
	Debate		

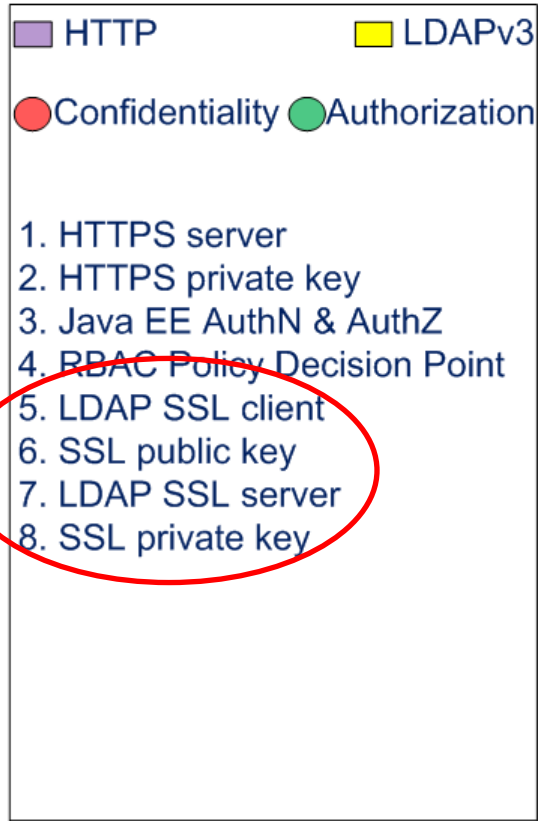
Other SoD Use Cases

Many possibilities apply to financial, government, health care, education, domestic, and business use cases.

5 – 8 Enable LDAP SSL



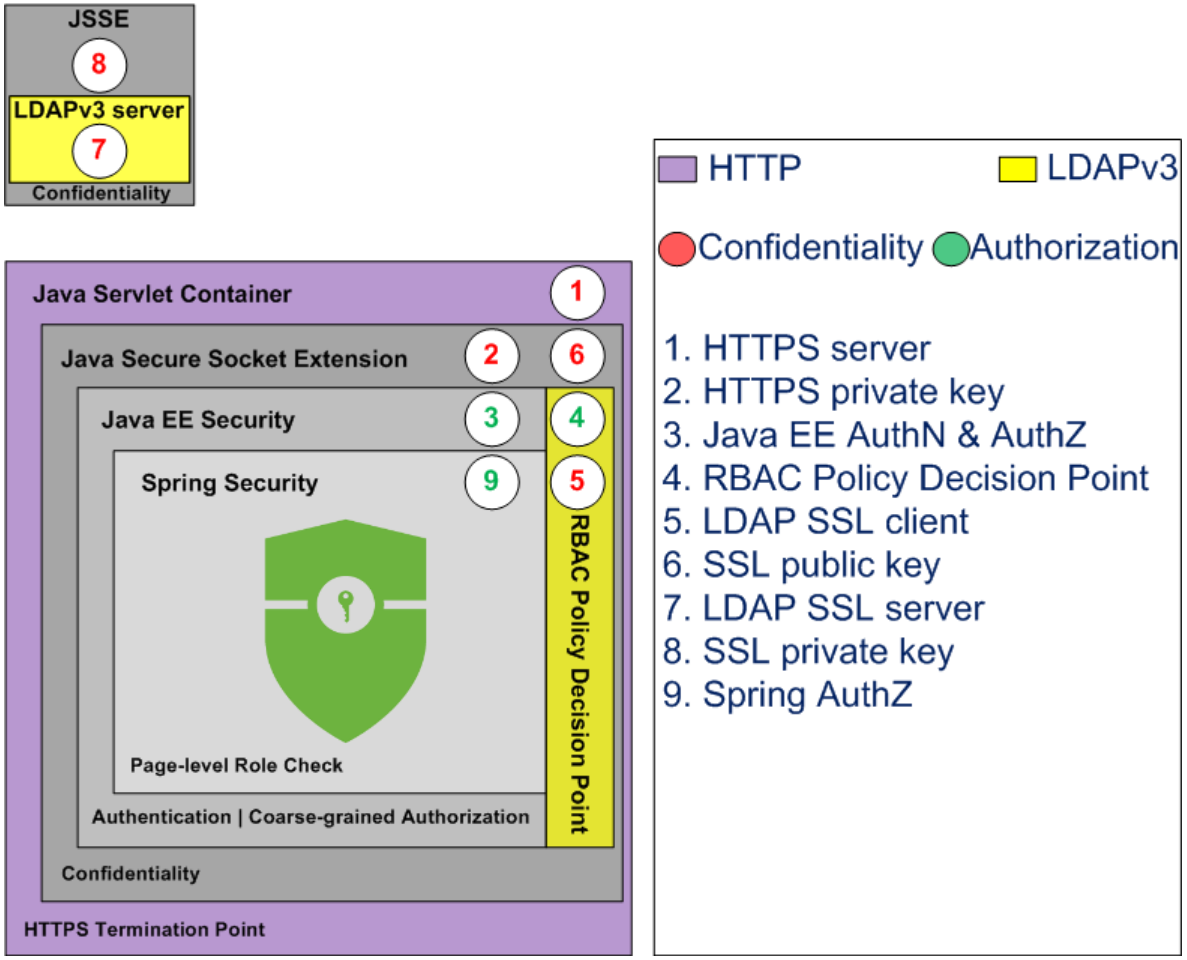




9. Enable Spring Security

a. Enable AuthZ

b. Role mapping



Enable Spring Security

Add dependencies to pom:

```
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-core </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-config </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-web </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
```

Enable Spring Security Interceptor

```
<bean id="fsi“
="org.springframework.security.web.access.intercept.FilterSecurityInterceptor">

  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager" ref="httpRequestAccessDecisionManager"/>
  <property name="securityMetadataSource">
   <sec:filter-security-metadata-source use-expressions="false">
        <sec:intercept-url pattern=
          "…/com.mycompany.page1"
          access="ROLE_PAGE1"
        />
   </sec:filter-security-metadata-source>
  </property>
</bean>
```

page-level authorization (declarative)

By default name must contain ROLE_

Role Mapping

Identity Propagation Java EE -> Spring Security

Spring Security uses PreAuthenticatedAuthentication filter to get java EE role mappings.

From the applicationContext.xml:

```
<bean id="preAuthenticatedAuthenticationProvider”
class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider">
  <property name="preAuthenticatedUserDetailsService"  ref="preAuthenticatedUserDetailsService"/>
</bean>
…
```

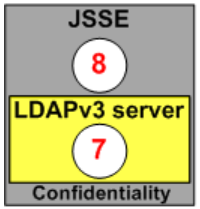
Role Mapping

Share Roles Between Java EE and Spring

Complete list of eligible roles found in app's web.xml:

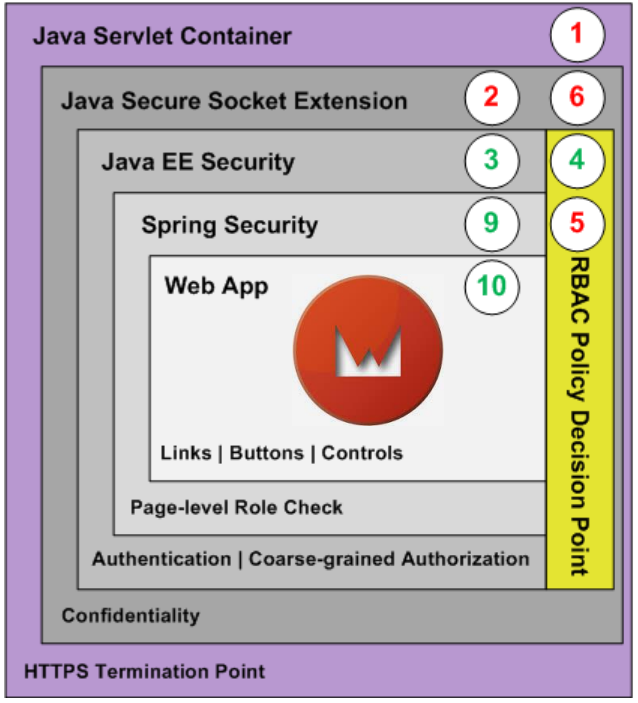
```
<!-- Declared in order to be used by Spring Security -->
<security-role>
  <role-name>ROLE_DEMO2_SUPER_USER</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE1</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE2</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE3</role-name>
</security-role>
```

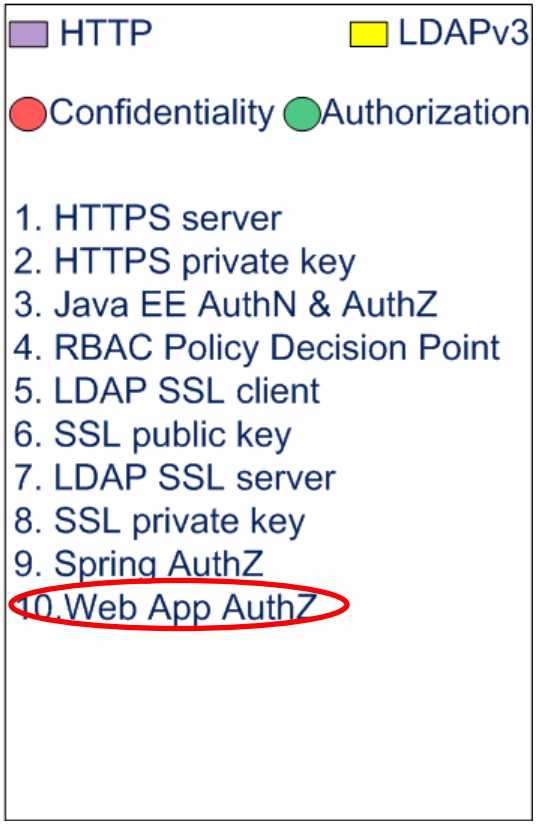
10. Web App Authorization



Add fine-grained checks:

a. Page links
b. Buttons
c. Other controls





Add Web Framework Security

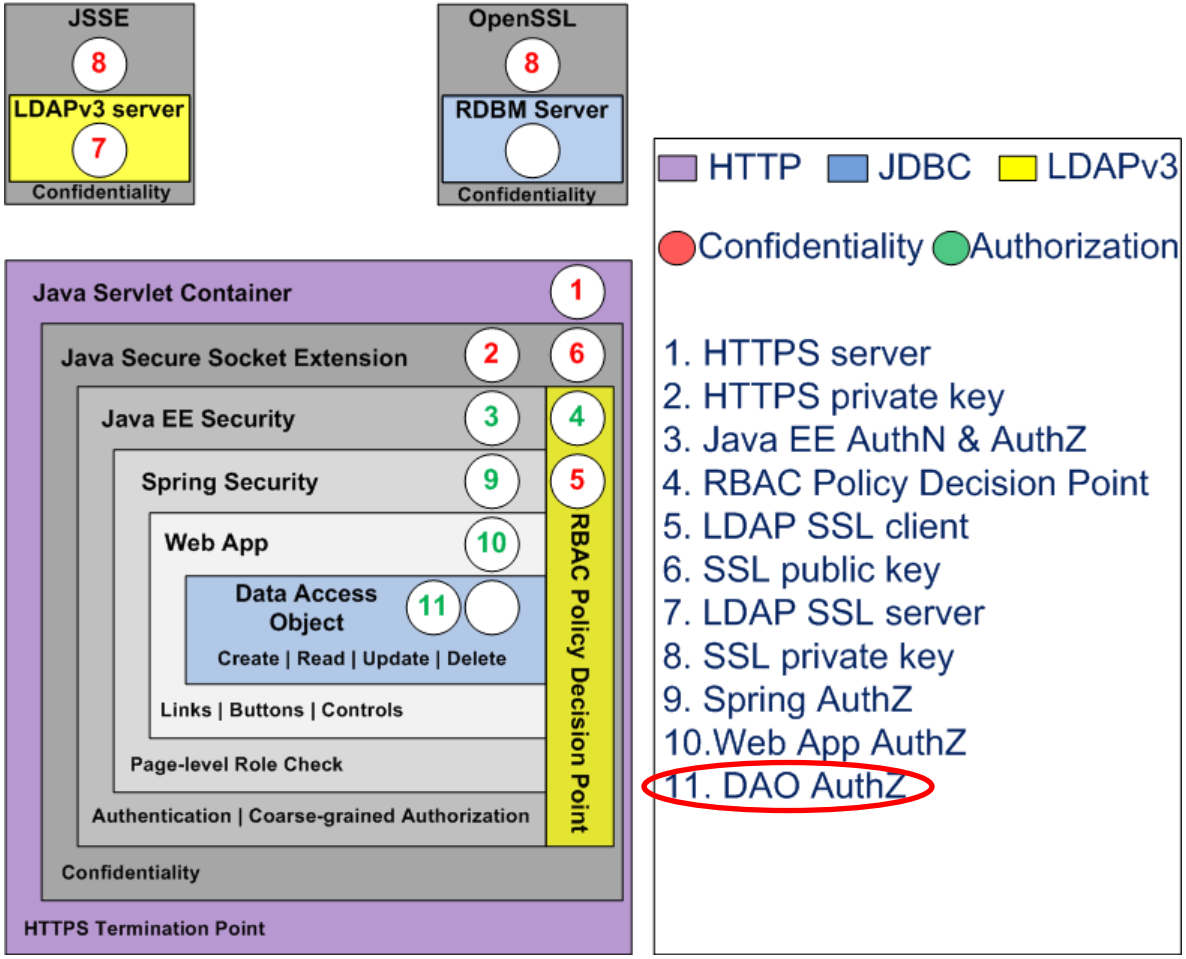
```
add(
   new SecureIndicatingAjaxButton( "Page1", "Add" )
   @Override
   protected void onSubmit( ... )
   {
      if( checkAccess( customerNumber ) )
      {
        // do something here:
      }
      else
      {
         target.appendJavaScript( ";alert('Unauthorized');" );
      }
   }
   });
```

fine-grained authorization (programmatic)

11. DAO Authorization

Add fine-grained Checks to:

a. Create
b. Read
c. Update
d. Delete



Add Security Aware DAO components

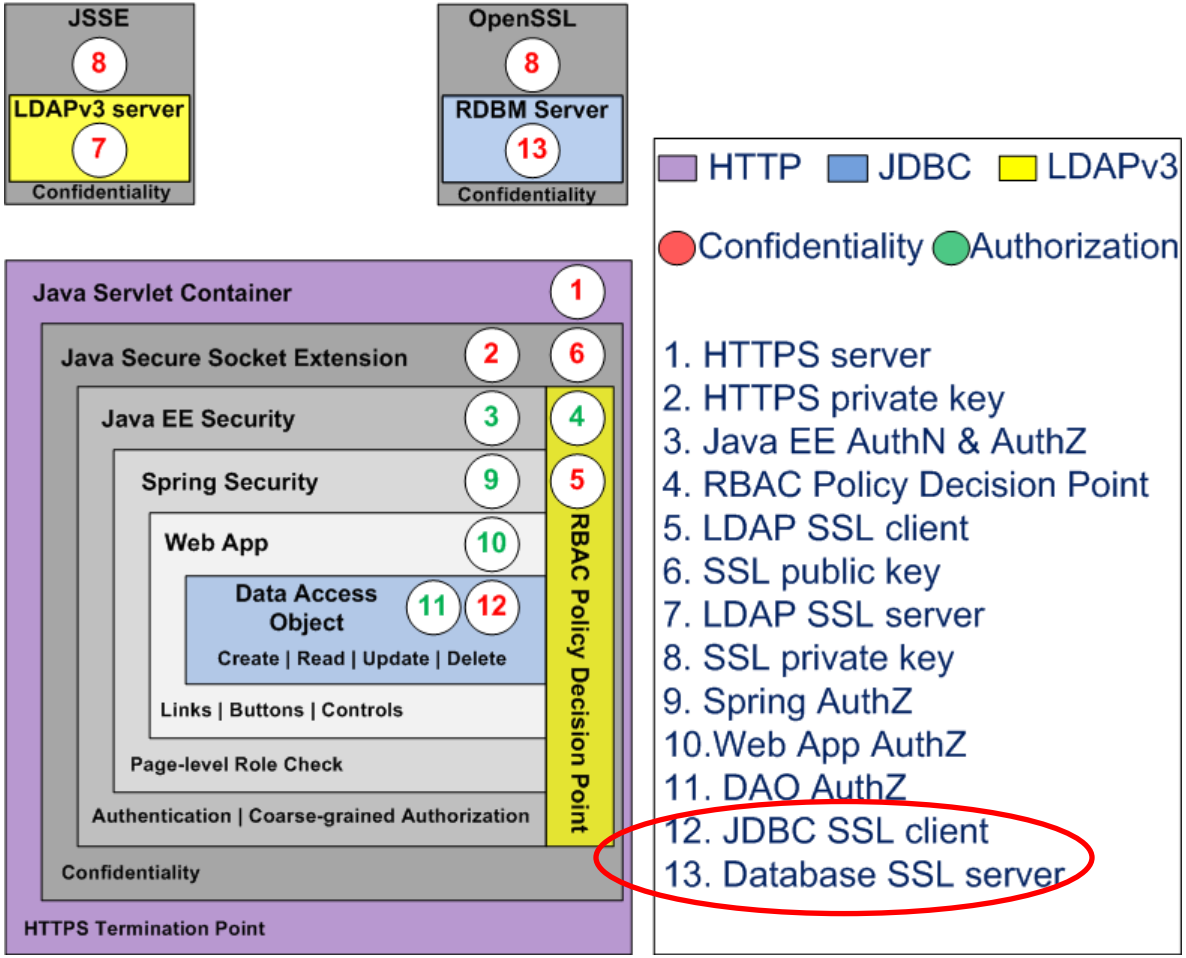
```
public Page1EO updatePage1( Page1EO entity )
{
   ...
 if(checkAccess(“Page1”,“Update”,entity.getCust()))
{
   // Call DAO.update method...
}
else
 throw new RuntimeException("Unauthorized”);
...
return entity;
}
```

fine-grained authorization (programmatic)

12, 13. Enable DB SSL

12. Client
 a. public key
 b. config

13. Server
 a. private key
 b. config



Apache Fortress Demo

- Three Pages and Three Customers
- One role for every page to customer combo
- Users may be assigned to one or more roles
- At most one role may be activated

Pages	Customer 123	Customer 456	Customer 789
Page One	PAGE1_123	PAGE1_456	PAGE1_789
Page Two	PAGE2_123	PAGE2_456	PAGE2_789
Page Three	PAGE3_123	PAGE3_456	PAGE3_789

Demo Usage Policy

- Both super and power users may access everything.
- But power users are limited to one role activation at a time.
- Super users are not restricted.

Super & Power Users	Customer 123	Customer 456	Customer 789
Page1	True	True	True
Page2	True	True	True
Page3	True	True	True

User123	Customer 123	Customer 456	Customer 789
Page1	True	False	False
Page2	True	False	False
Page3	True	False	False

User1	Customer 123	Customer 456	Customer 789
Page1	True	True	True
Page2	False	False	False
Page3	False	False	False

User1_123	Customer 123	Customer 456	Customer 789
Page1	True	False	False
Page2	False	False	False
Page3	False	False	False

Apache Fortress Demo

- https://github.com/shawnmckinney/apache-fortress-demo

User-tic-tac-toe	Customer 123	Customer 456	Customer 789
Page1	False	True	True
Page2	True	False	False
Page3	True	False	False

Understanding “Forklifted” Applications

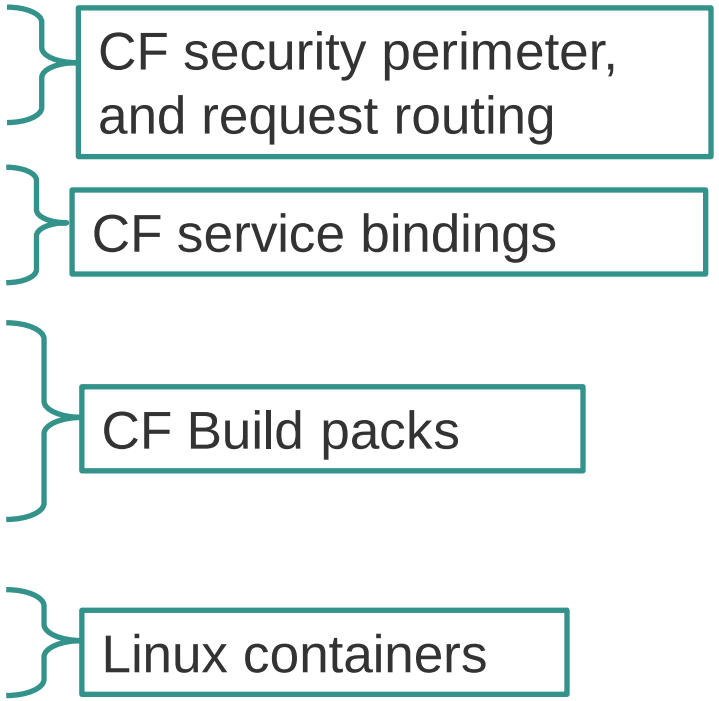
- Refers to migrating traditional enterprise applications like apache-fortress-demo into Cloud Foundry via “cf push.”
  - Helps illustrate differences in:
    - Security Architecture
    - Dev Ops processes

- It sounds worse than it is.

Pivotal

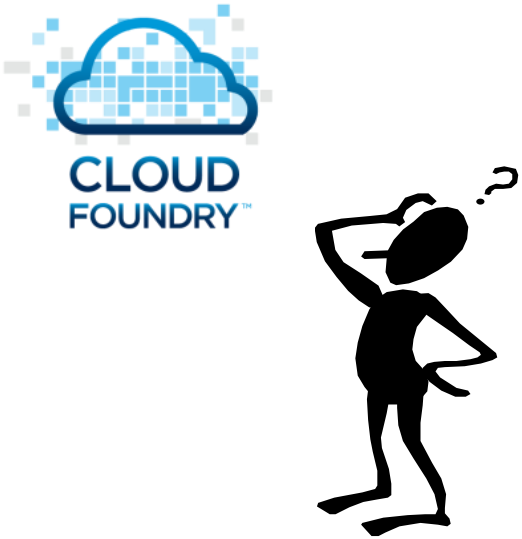
Typical “TODO” List

1. TLS Termination
2. Credential Provisioning
3. JEE Realm Configuration
4. JSSE Truststore Management
5. Warden Container Isolation

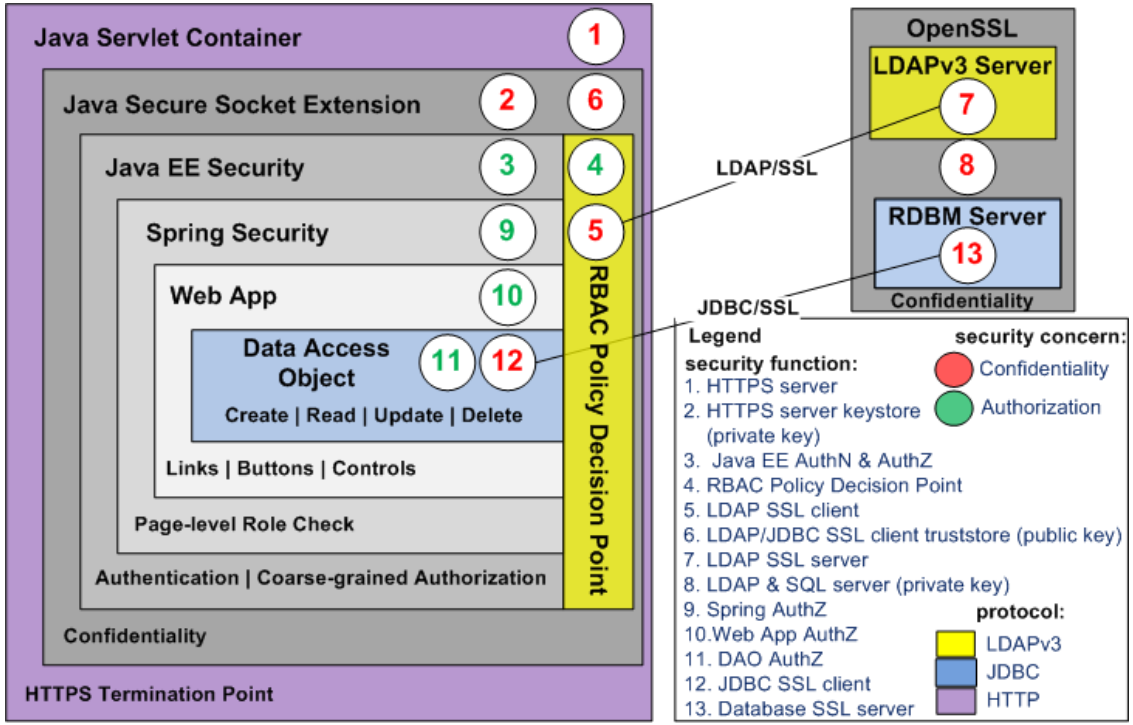


Pivotal

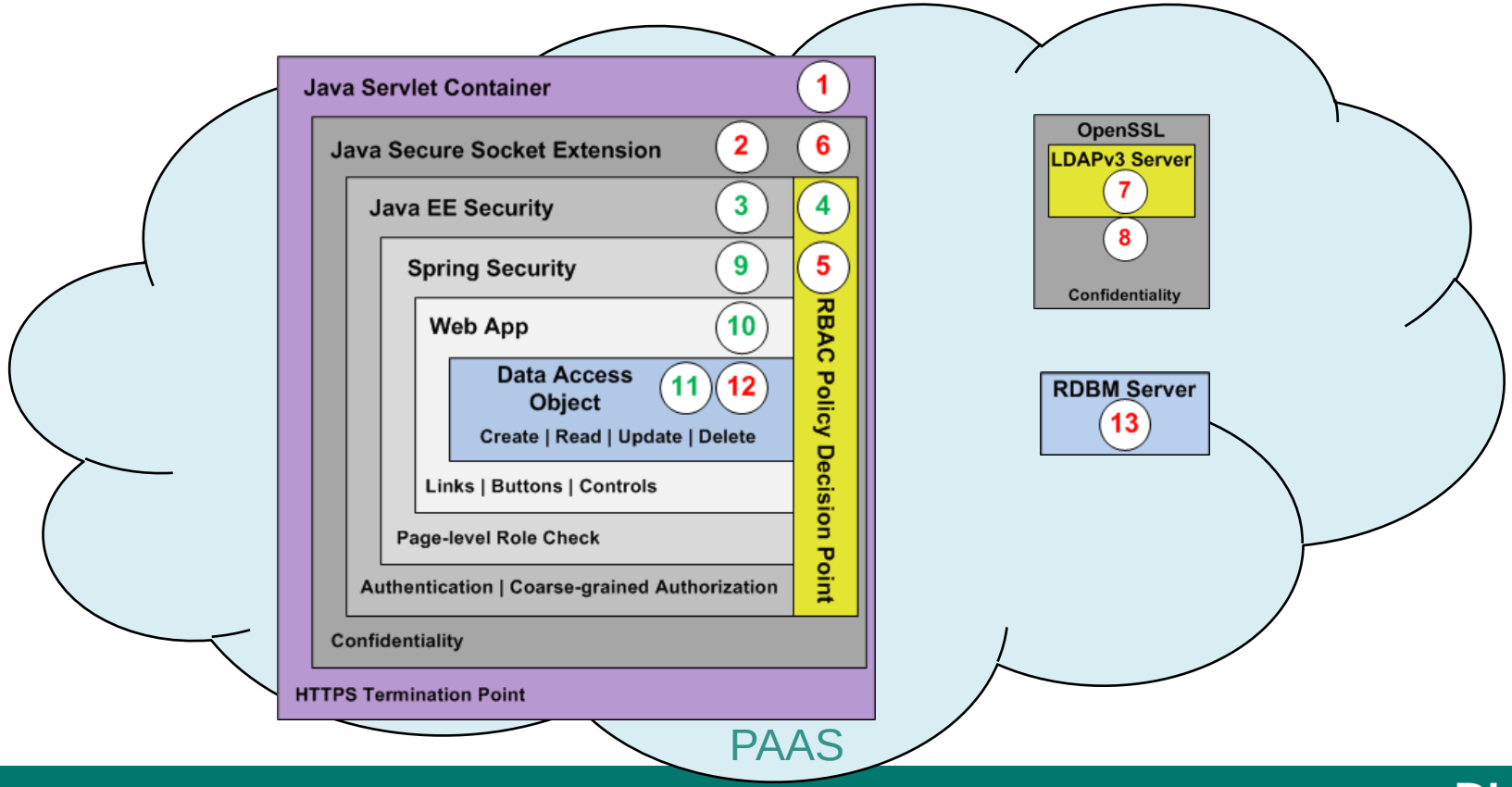
- shell> cf push apache-fortress-demo …



apache-fortress-demo in Cloud Foundry

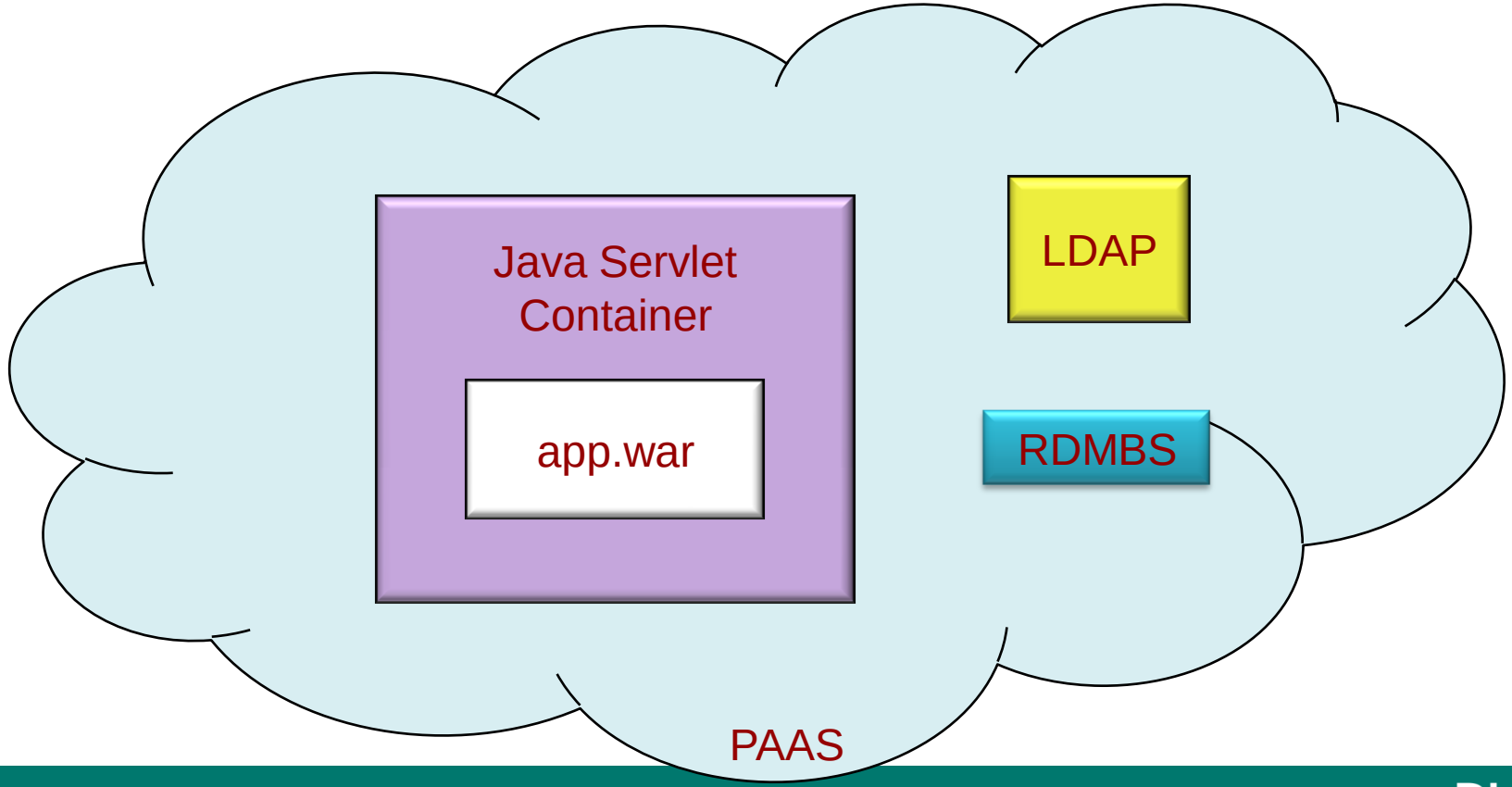


apache-fortress-demo in Cloud Foundry

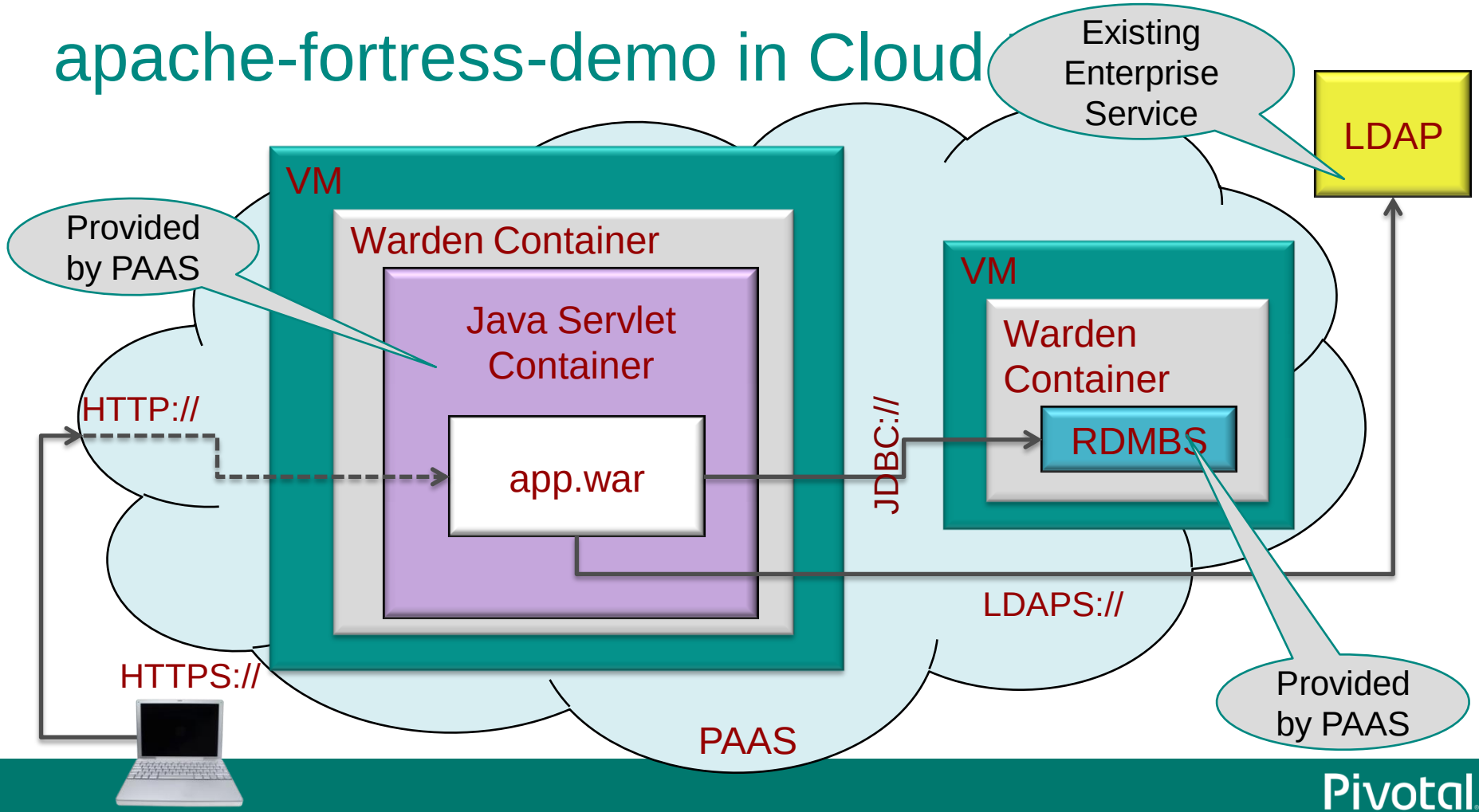


Pivotal

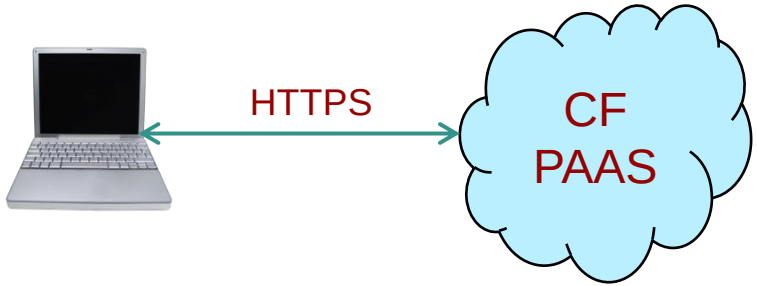
apache-fortress-demo in Cloud Foundry



apache-fortress-demo in Cloud



TLS Termination



- TLS from user's browser terminated at cloud entry point
- Only one certificate for all hosted applications.
- App's IP addr:port not accessible from outside PAAS.

Pivotal

Credential Provisioning

- App's dependencies declared via service bindings
  - "Managed" or "User-Provided".
- Connection strings injected via VCAP_SERVICES
  - Credentials randomly generated.
- No provisioning of hardcoded JDBC credentials
  - e.g., in application-Context.xml or fortress.properties.

Pivotal

cf services

```
shell>  cf create-service  p-mysql  dev-plan  fd2-mysql
shell>  cf create-user-provided-service  \
          page4-oauth2-service  \
          -p “comma, separated, param, names”
shell>  cf bind-service apache-fortress-demo fd2-mysql
shell>  cf bind-service apache-fortress-demo \
          page4-oauth2-service
```

Pivotal

Security Implications

- Strong credentials at target service.
- But, potentially accessible to different staff
  - Privileged operators, not the developers.
- Complicates audit reporting?
  - Or reduces it?

Cloud Foundry Buildpack Changes

- Changes to Tomcat configuration are done via buildpack.
  - JEE Container Realm Configuration
    - Add apache-fortress-realm-proxy jar
  - JSSE Trust Store Management
    - Add your CA certs file to JDK resources

Pivotal

Apache-fortress-demo Java Buildpack

- shell> git clone https://github.com/johnpfield/java-buildpack.git
- shell> cd java-buildpack
- shell> cp ~Downloads/fortress-realm-proxy-1.0-RC41-SNAPSHOT.jar \
  ../java-buildpack/resources/tomcat/lib/
- shell> cp ~/Downloads/mycacerts \
  ../java-buildpack/resources/open_jdk_jre/lib/security/

Pivotal

Building the Java Buildpack

- shell> rvm info
- shell> rvm use 2.1.2
- shell> bundle install
- shell> bundle exec rake package OFFLINE=true
- shell> cf create-buildpack apache-fortress-demo-buildpack \
  build/java-buildpack-offline-bb567da.zip [illegible] enable
- shell> cd ../apache-fortress-demo
- shell> mvn clean package
- shell> cf push apache-fortress-demo –t 90 \
  -p target/apache-fortress-demo.war –b apache-fortress-demo-buildpack

App to deploy

Buildpack to use

Pivotal

Security Tradeoffs with Buildpacks

- Choose from 3 deployment modes
  - Easy, Expert, Offline
- Tighter configuration control, versus latest-and-greatest.
- Individual developers avoid any container security configuration.

Pivotal

Warden Container Isolation

- Applications are deployed inside a Linux Container
- Multiple runtime environments on a single VM host.
- Resource isolation via name-spacing in kernel
- File system isolation via overlayfs
- Think: “chroot on steroids”
- Network isolation via iptables rules

Pivotal

Warden Container Isolation

PAAS (ESX Node)

DEA VM

Hostname: vm-09bf580a-69a0-431c-9741-bb49c4f318b8
VNIC: eth0
Filesystem: /var/vcap/data/warden/depot/
IP: 10.110.57.60
Memory: 4Gb

VNIC: w-17ruu5224qa-0
IP: 10.254.0.1
Filesystem: ./w-17ruu5224qa/tmp/rootfs

VNIC: w-17ruu5334qb-0
IP: 10.254.0.5
Filesystem: ./17wruu5224qb/tmp/rootfs

Warden Container "A"

Hostname: 17ruu5224qa
VNIC: w-17ruu5224-qa-1
Filesystem: /home/vcap
IP: 10.254.0.2
Memory: 1Gb

Warden Container "B"

Hostname: 17ruu5224qb
VNIC: w-17ruu5224-qb-1
Filesystem: /home/vcap
IP: 10.254.0.6
Memory: 1Gb

Pivotal

Application Security Groups

- Provide additional control over container egress.
- Supplementary iptables rules
  - Add specific protocols and ports
- Per deployment, or per Org/Space.
- Staging and/or Running

Pivotal

“Forklifted” Vs. “Cloud Native” Security

Attribute	Forklifted	Cloud Native
Application Security Role	• Singular • Implicit • Invariant	• Multiple • Explicit • Deploy-time Configurable
AuthZ PEP	• Container • Spring • programmatic	• Spring • programmatic
AuthN PEP	• Container based	• Spring
AuthZ PDP	• Synchronous • Statically bound • LDAP/AD via groups, roles	• Runtime negotiation • UAA / OAuth2 via roles, scopes
AuthN PDP	• Single common trust domain	• Arbitrary trust domains
PAP	• Centralized • Pre-condition	• Distributed • On-demand

Pivotal

Wrap Up

Use Case Summary

1. SAML2 RP application for enterprise-centric SSO
2. µService OAuth2 Client Credentials Grant – 2 Party
3. µService OAuth2 Client Credentials Grant – 3 Party
4. Secure RBAC Java Web Application with Tomcat, Spring
5. Same, Forklifted to Cloud Foundry

Pivotal

Lessons Learned (John)

- Assume trust relationships are dynamic
- Expect to fulfill multiple security roles
- Beware of unexpected SOD violations
  - Use ANSI RBAC
- Beware of unwanted transitive trust
- Think Holistically: security of cloud native requires coexisting with installed base.

Pivotal

Closing Thoughts (Shawn)

1. Use TLS across all remote connections
   - *Confidentiality and Integrity*
2. Apply security controls across many layers
   - *Defense in Depth*
3. Never allow users more than they need to do their jobs
   - *Principle of Least Privilege*

Related Sessions

- CON3568 - Federated RBAC: Fortress, OAuth2 (Oltu), JWT, Java EE, and JASPIC
  - October 27, 11:00 am - 12:00 pm | Hilton—Plaza Room B
- CON2324 – A Practical Guide to Role Engineering
  - October 27, 2:30 p.m. | Hilton—Plaza Room B
- CON2325 - RBAC-Enable Your Java Web Applications with Apache Directory Fortress
  - October 29, 1:00 pm - 2:00 pm | Hilton—Plaza Room A

Links

- http://iamfortress.net/2015/09/01/apache-directory-fortress-saml-demo/
- http://iamfortress.net/2015/02/16/apache-fortress-end-to-end-security-tutorial/
- https://github.com/shawnmckinney/
- https://github.com/johnpfield/

Pivotal

Contacts

John

- email: jfield@pivotal.io
- twitter: @architectedsec
- blog: https://johnpfield.wordpress.com

Shawn

- email smckinney@symas.com
- twitter @shawnmckinney
- url http://iamfortress.net/