Pivotal

Security in the world of microservices

Madhav Sathe
msathe@pivotal.io
Oct 2018

Safe Harbor Statement

The following is intended to outline the general direction of Pivotal's offerings. It is intended for information purposes only and may not be incorporated into any contract. Any information regarding pre-release of Pivotal offerings, future updates or other planned modifications is subject to ongoing evaluation by Pivotal and is subject to change. This information is provided without warranty or any kind, express or implied, and is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions regarding Pivotal's offerings. These purchasing decisions should only be based on features currently available. The development, release, and timing of any features or functionality described for Pivotal's offerings in this presentation remain at the sole discretion of Pivotal. Pivotal has no obligation to update forward looking information in this presentation.

Agenda

- Challenges
- Standards and protocols
- PCF built for security
- Demo
- Bonus - Some patterns

The fundamentals

- Identity
  - Who you are
  - E.g. a person, web application, mobile application or a microservice
- AuthN/Authentication
  - Establish the identity (using credentials)
- AuthZ/Authorization
  - What resources you can access, what actions you can perform
- Federated identity
  - E.g. when I login to Pivotal sites using Pivotal Identity I can access my account on Salesforce (without having to login to Salesforce separately)
- Delegated authorization
  - Limited amount of access given typically to an application on behalf of someone

Pivotal

Microservices deliver business agility?

Right 

Microservices deliver faster time to market?

Right 

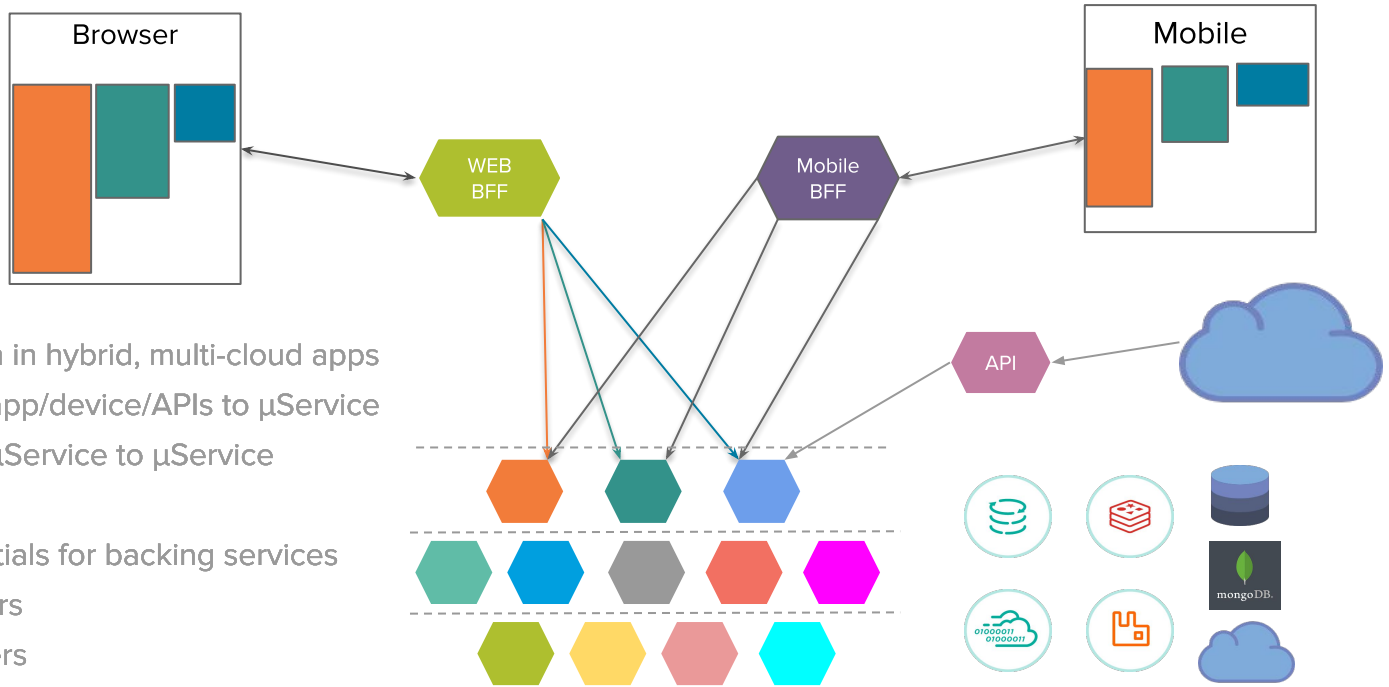
Microservices make security simple?

Well... 

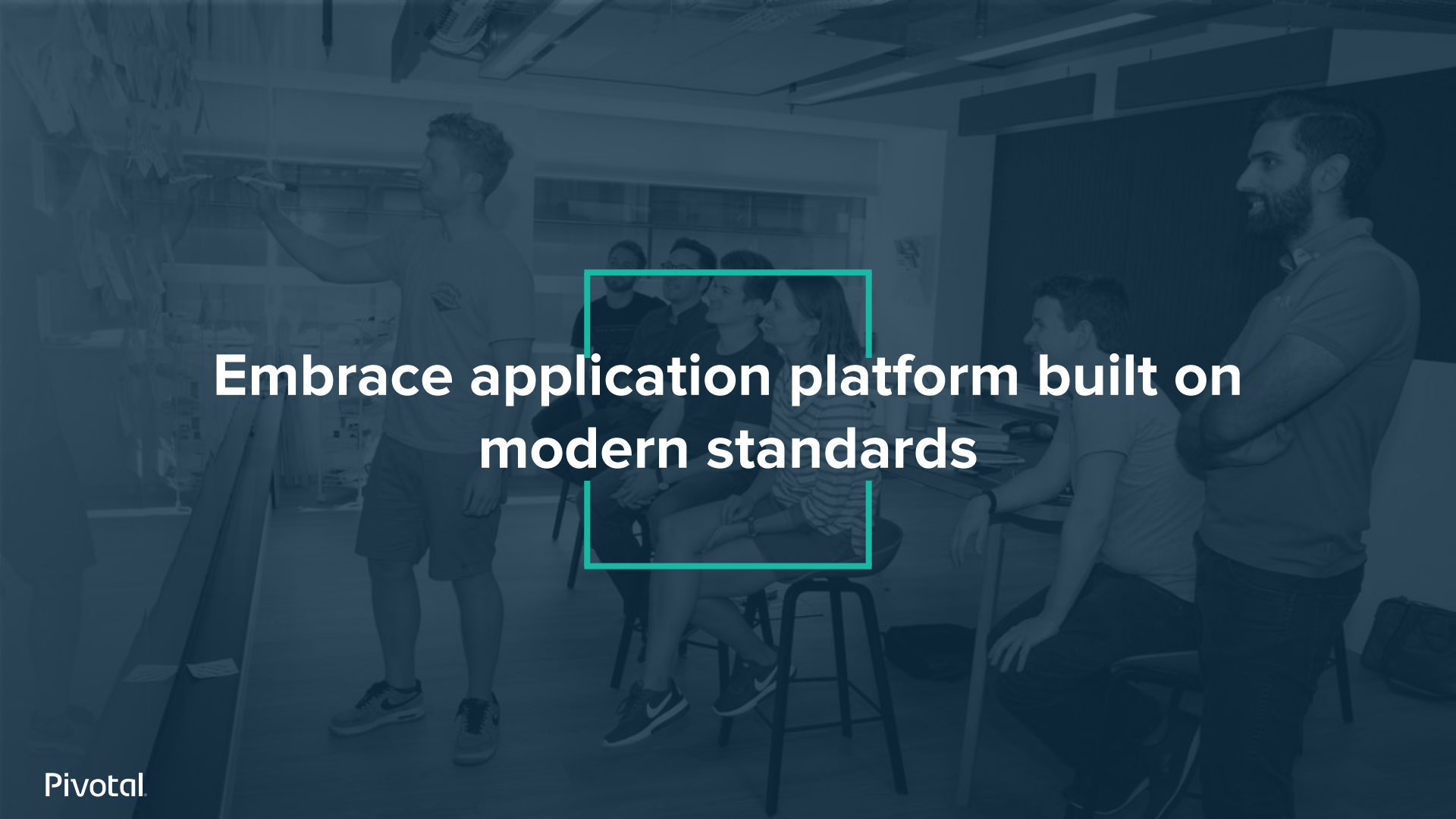
In the high speed, agile world of APIs & microservices

security is a moving target

Developer Challenges in µServices Architecture



> Identity federation in hybrid, multi-cloud apps
> AuthN/AuthZ for app/device/APIs to µService
> AuthN/AuthZ for µService to µService
> Single Sign On
> Managing credentials for backing services
> Creating containers
> Securing containers
> Runtime CVEs
> Threat vectors such as DDOS, SQL injection, etc



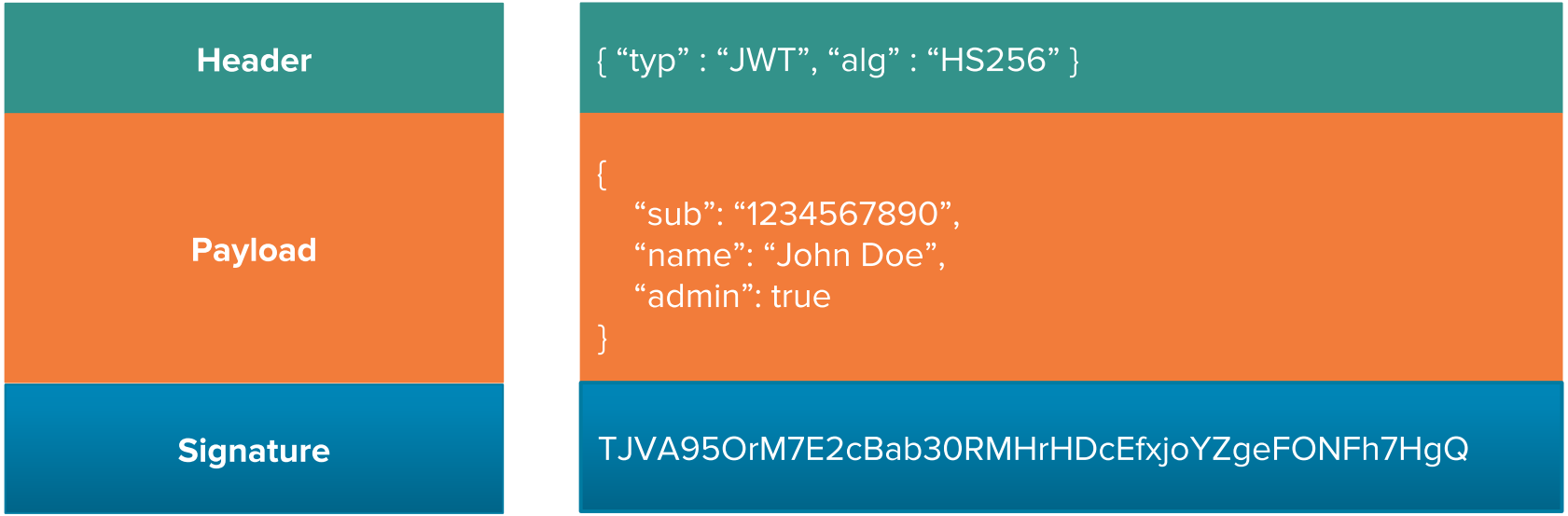
Pivotal

Key standards and protocols

- **JWT** - JSON Web Token
  - Compact
  - URL safe
  - Base-64 encoded
  - Self contained
  - Used along with other JOSE standards - JWA, JWK, JWS and JWE
- **OAuth2.0**
  - Delegated authorization
- **OpenID Connect**
  - OAuth2.0 + Identity layer

JWT https://jwt.io/

Standard used by OpenID Connect to share asserted identity of the user (ID Token) and (optionally) by OAuth2.0 to authorize delegated access (by value Access Token)



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOn

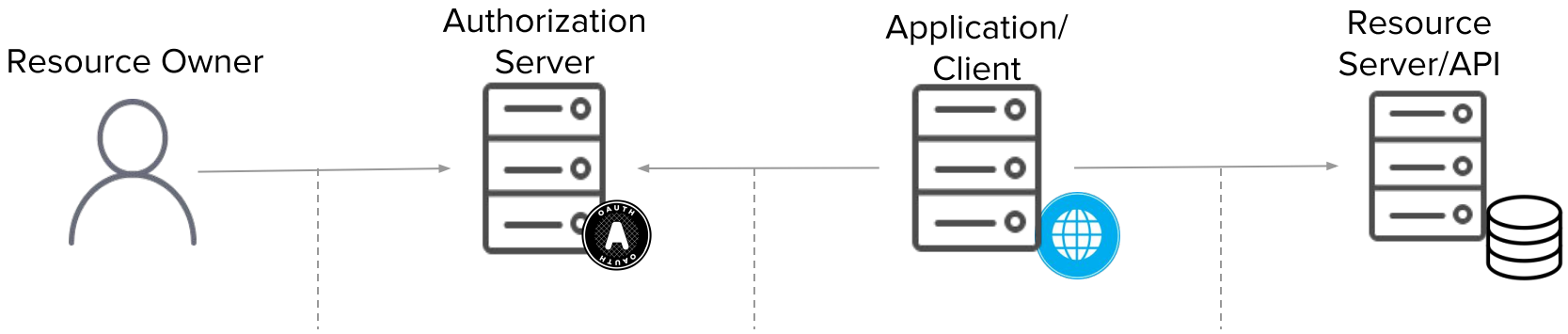
RydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

OAuth 2.0

Protocol to provide delegated access control to applications

RFC	Title	Purpose
6749	The OAuth 2.0 Authorization Framework	**how a token can be obtained**
6750	The OAuth 2.0 Authorization Framework: Bearer Token Usage	**how to make HTTP requests with the token once it is obtained**

Meet the actors



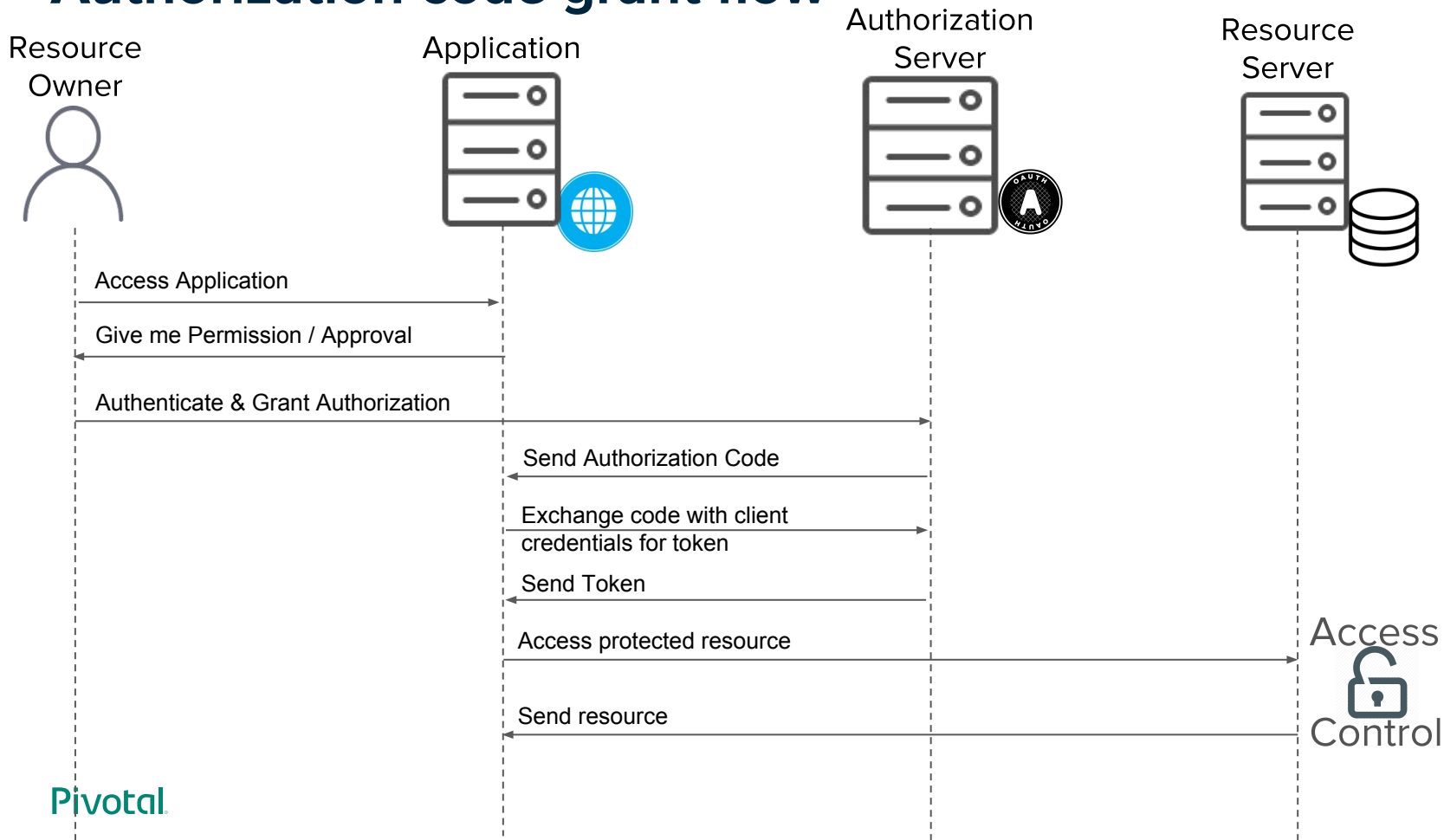
OAuth 2.0 Grant types: application types

- **Authorization code**: web application
  - Mostly used for server side apps such as web apps
  - Most common use of OAuth
- **Password**: Native Mobile, Desktop, or Command Line App
  - RO shares credentials with the client app → client app is trusted by RO
- **Implicit**: Single-Page JavaScript App
  - Client secret that is not guaranteed to be confidential
- **Client credentials**: service to service
  - When client app is acting on its own behalf

User is authenticated

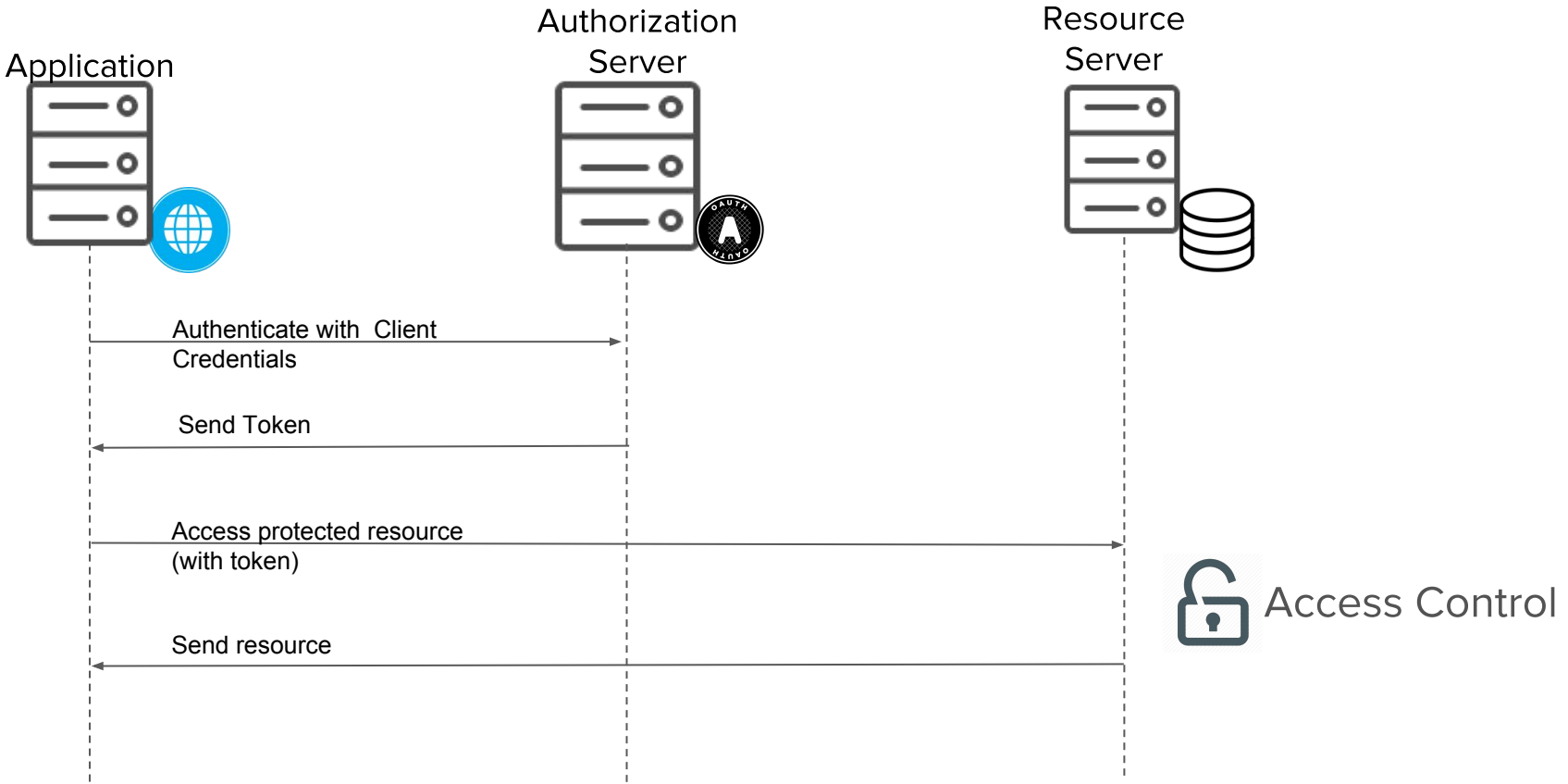
User is not authenticated

Authorization code grant flow

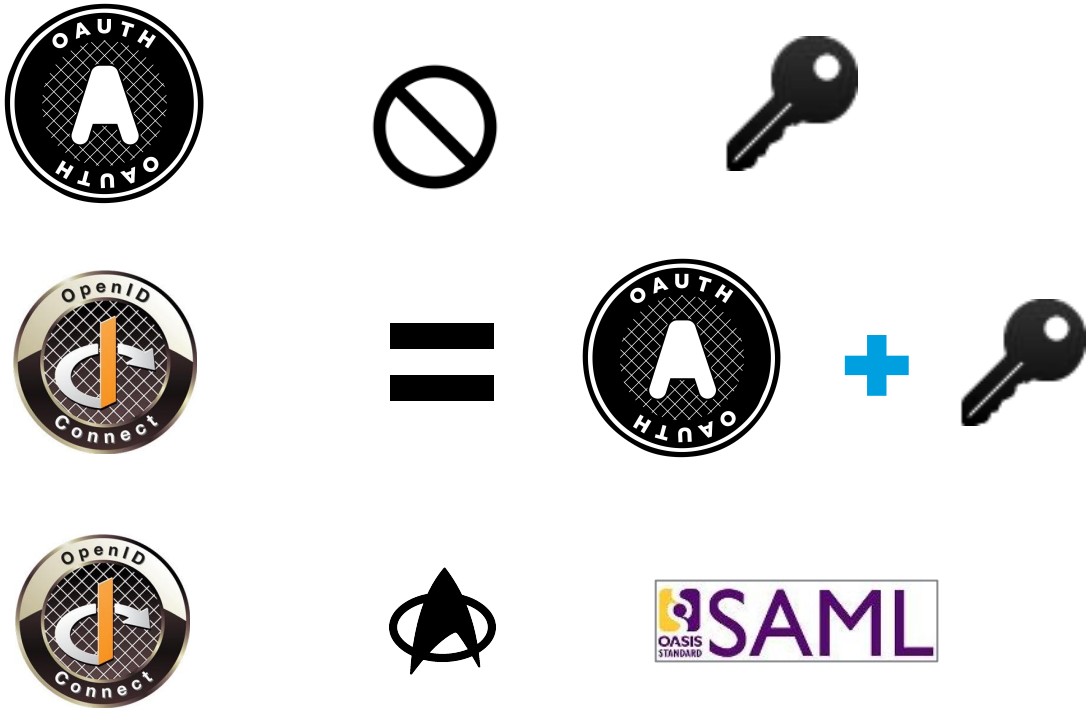


Pivotal

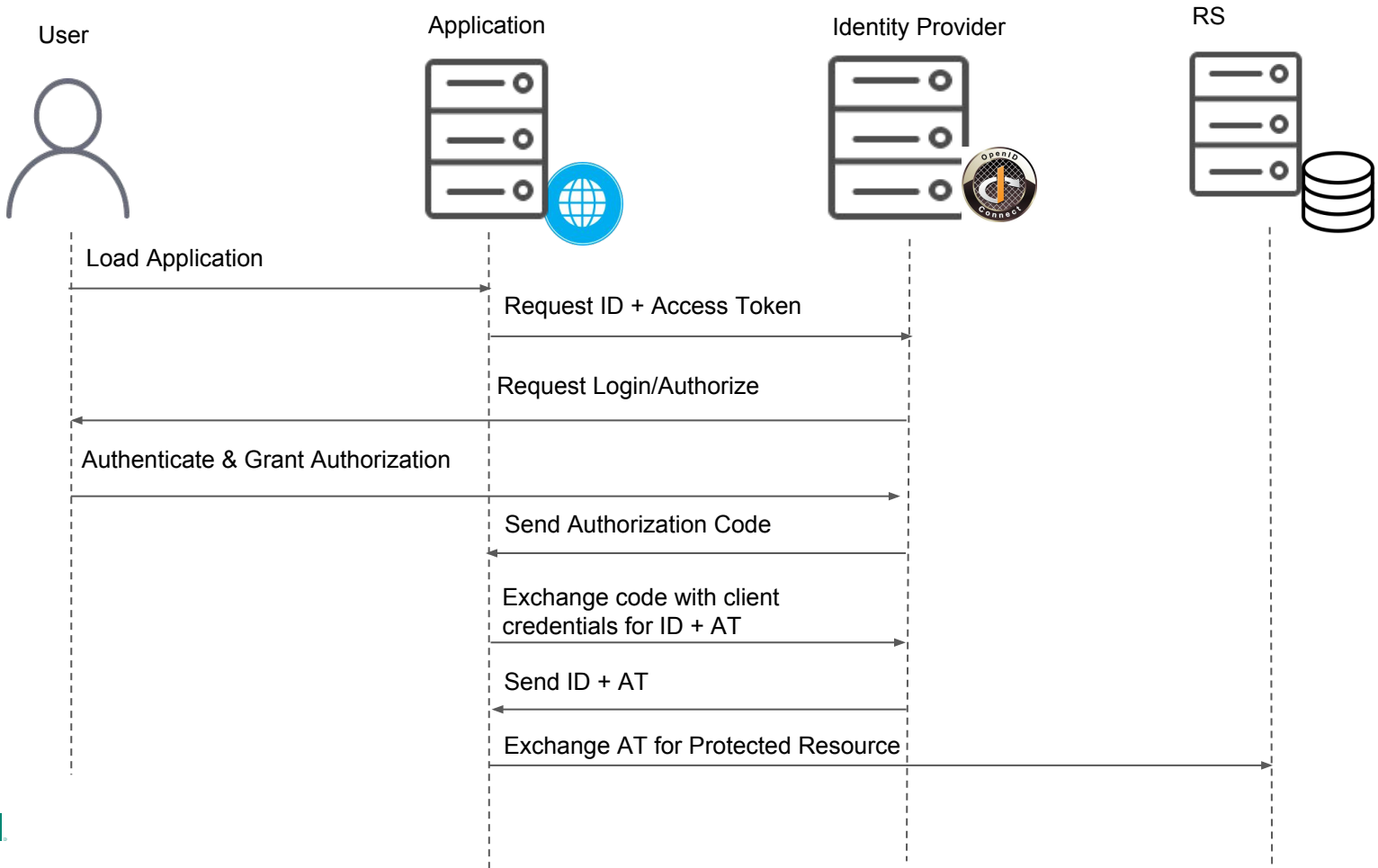
Client credentials grant flow



OpenID Connect 1.0



Typical OIDC flow



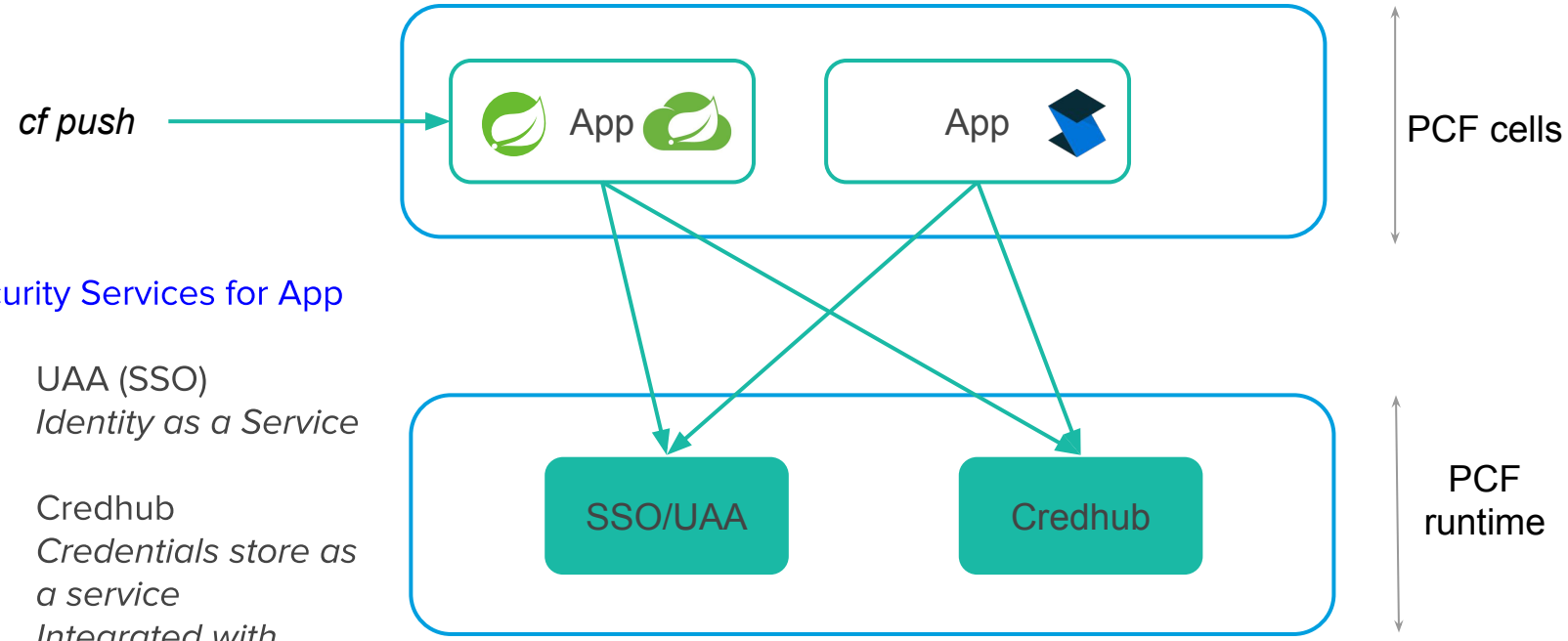
Types of tokens

- Access token
  - Generated by Auth Server for Client to access the Resource Server/API/microService
  - Goes in the Authorization header as a Bearer token
  - Opaque token
    - Randomly generated string
    - Resource Server can validate with Auth server
    - Risk mitigation - can be revoked
  - JWT
    - Contains user information and scopes
    - Self validated, saves roundtrip to Auth Server
    - Bummer - can’t be revoked
- Refresh token
  - Token used by client to get a new Access Token
- ID token
  - OIDC server
  - Always JWT
  - Contains user identity and claims
  - Meant to be used by Client
  - Self validated, saves roundtrip to Auth Server

Enterprise ready security

Push security out of the application code on to the platform

A platform with security services for your apps...



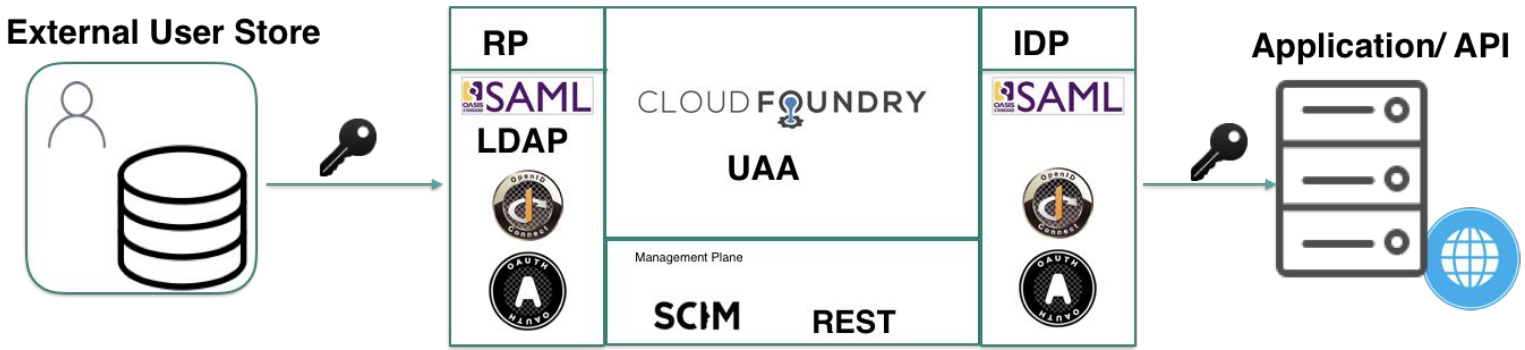
- Security Services for App
  - UAA (SSO)
    Identity as a Service
  - Credhub
    Credentials store as a service
    Integrated with service brokers

- Identity Proxy
- User AuthN/AuthZ
- Service-to-Service AuthN/AuthZ
- Credential Generation
- Credential Storage
- Credential Rotation

UAA - User Account and Authentication Service

OAuth2 server bundled/integrated within PCF that can be used for centralized identity management

Its primary role is as an OAuth2 provider, issuing tokens for client applications



UAA - OIDC Certified IDP

https://openid.net/certification/

OpenID Foundation ▾ Intellectual Property ▾ Current Workir

OpenID Connect FAQ and Q&As Workshops ▾

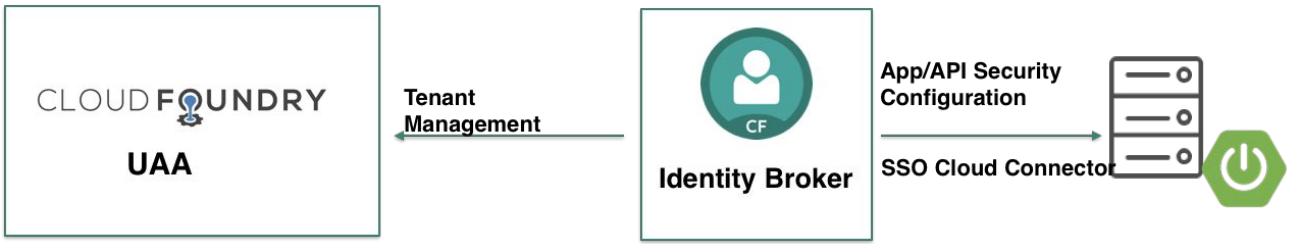
PayPal	Login with PayPal	
Peercraft ApS	Peercraft	19-Jan-2016
Ping Identity	PingFederate Summer 2015 Release	10-Apr-2015
Ping Identity	PingFederate 9.1.1	
Pivotal	Pivotal Cloud Foundry 2.2 UAA	17-Jul-2018
Privacy Vaults Online (PRIVO)	PRIVO-Lock	23-Oct-2015
ProSiebenSat.1 Media	7Pass ^2.0.0	7-Aug-2017
Recruit	Recruit ID	16-May-2018

UAA Basics

- Multi-tenant IDP
- Users : Internal or External (LDAP, SAML, OpenID Connect)
- Permissions
  - Groups associated with Users
  - Groups associated with Applications
  - On User's Behalf - As Scopes
  - On Application's Behalf - As Authorities
- External Group Mappings
  - Derive UAA Group from External Groups
  - LDAP, SAML, OpenID Connect
- External User Attributes
  - LDAP, SAML, OpenID Connect
  - In ID_Token and /UserInfo

SSO Service

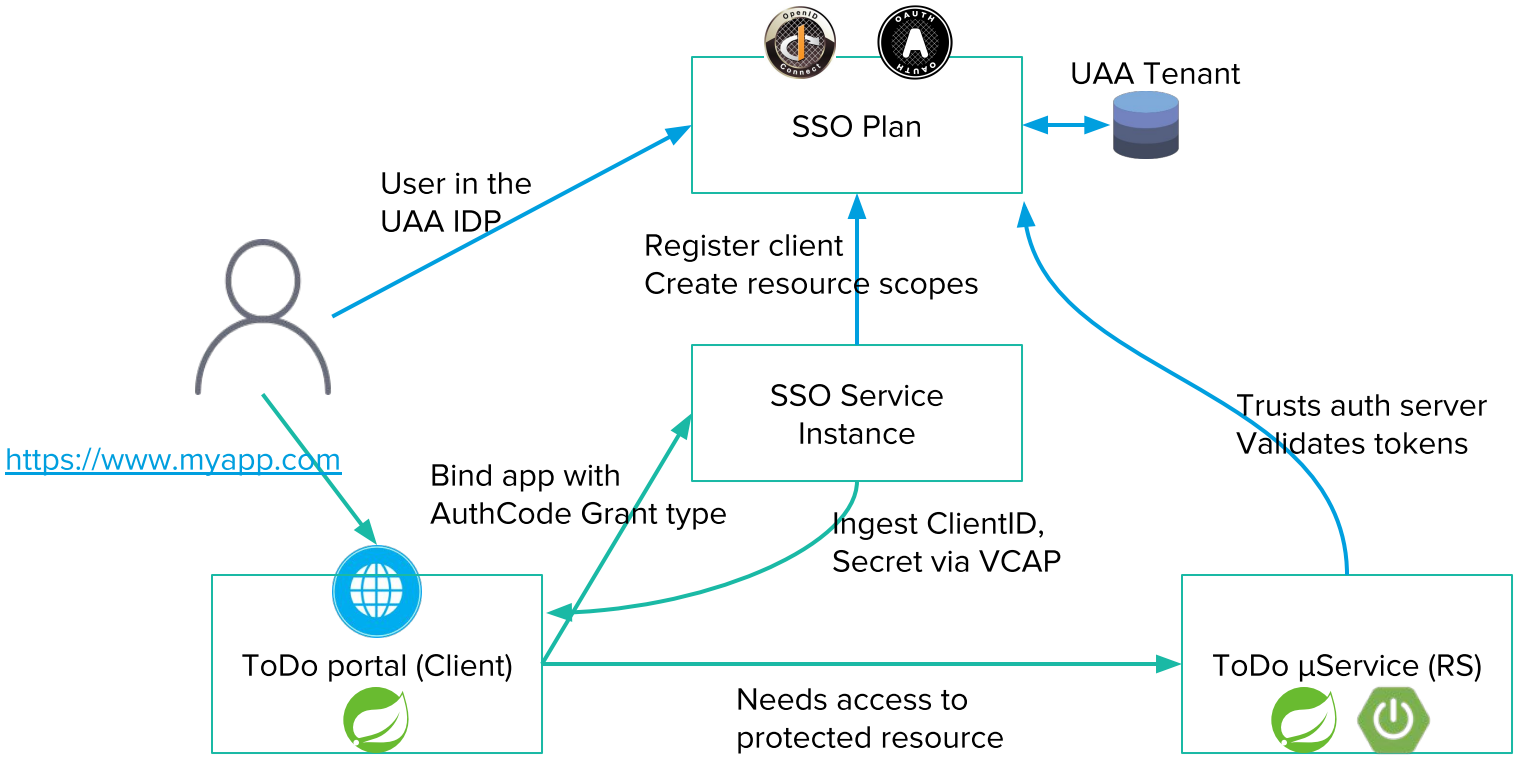
- Secure apps with minimal developer overhead
- Provide UAA as a service
- Self-service dashboard for admins
- SAML IDP integration is hard, SSO service broker makes it easy for security admins



Personas

- Security admin
  - Create plans
  - Configure backing IDP
  - Map IDP users and attributes
- Platform operator
  - Create SSO service instance
  - Create Admin user
  - Create application users
- Application developer
  - Application manifest
  - Spring Boot/Security annotations
  - SSO service binding
  - Managing application scopes

Pivotal

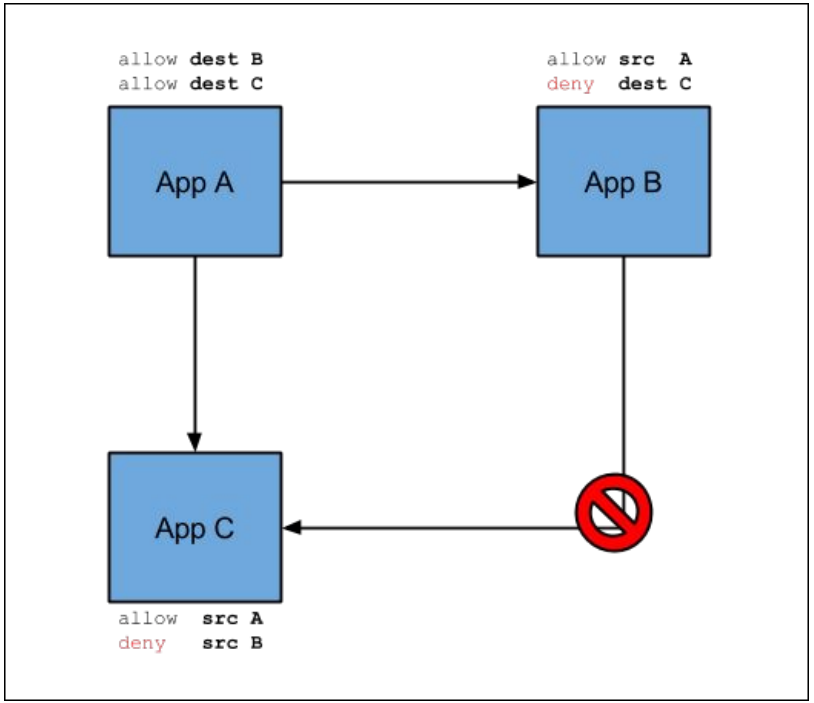

SSO Plan
UAA Tenant
User in the
UAA IDP
Register client
Create resource scopes
SSO Service
Instance
Trusts auth server
Validates tokens
https://www.myapp.com
Bind app with
AuthCode Grant type
Ingest ClientID,
Secret via VCAP
ToDo portal (Client)
ToDo µService (RS)
Needs access to
protected resource

Tokens are not security

Facebook access token breach announced on Sep 25

Container-to-Container Networking

- C2C enables direct communication between application containers on Cloud Foundry
- Fine grained policies for µService to µService access
- Policies can be defined via cf cli so easy to incorporate into your pipelines no need for tickets to configure firewalls
- Provides DNS based service discovery
- https://docs.cloudfoundry.org/concepts/understand-cf-networking.html



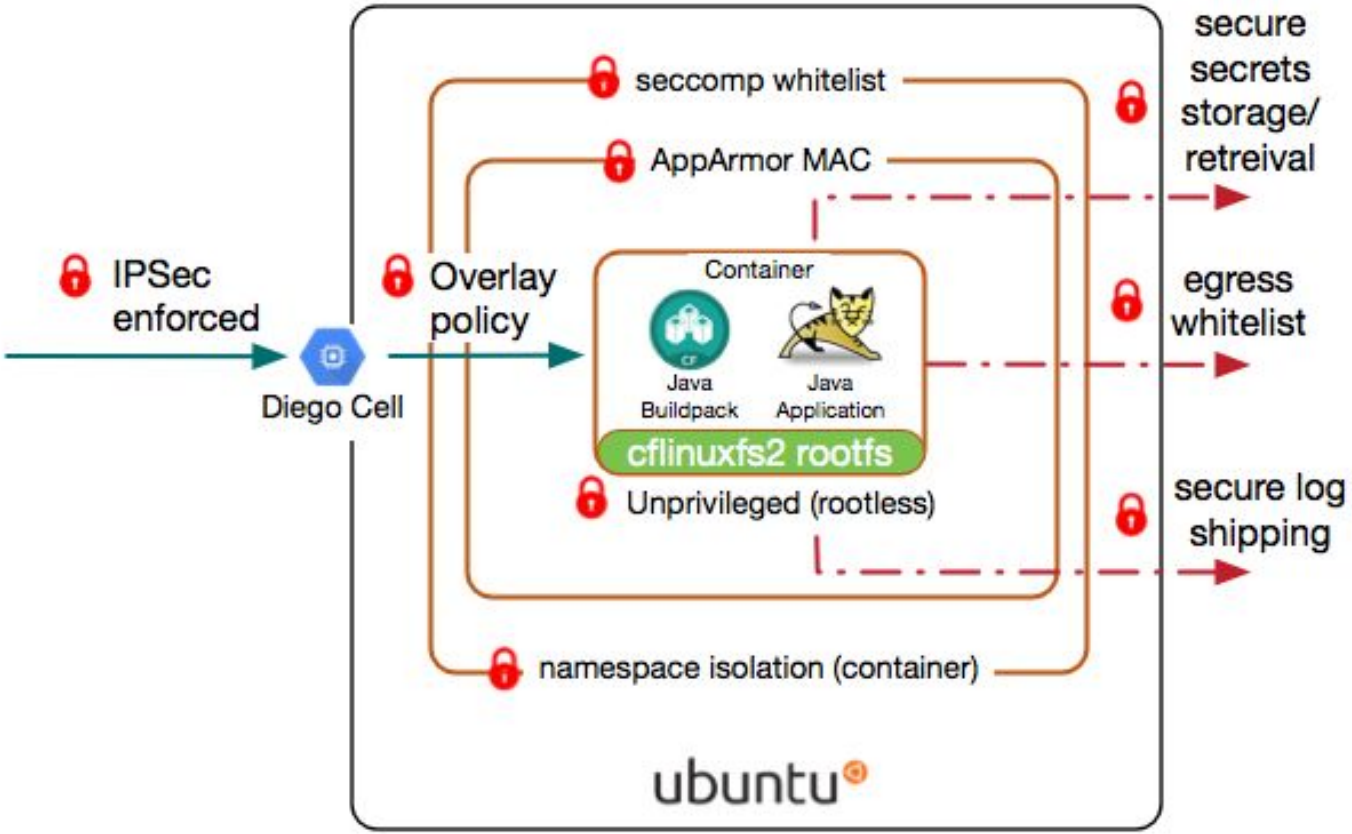
What about security of the container?

PCF creates a natively secured container for every application instance

Multiple layers of overlapping container security provides **defense in depth**

- Complete isolation for containers using namespaces + pivot_root
- Unprivileged containers by default
- Cgroups to restrict resource usage and access control
- Dropped capabilities
- AppArmor as Mandatory Access Control layer
- Seccomp filtering to block harmful system calls
- Vetted and hardened OS to reduce attack surface
- Vetted and fine tuned RootFS
- **All of the above with ZERO developers/operations overhead**

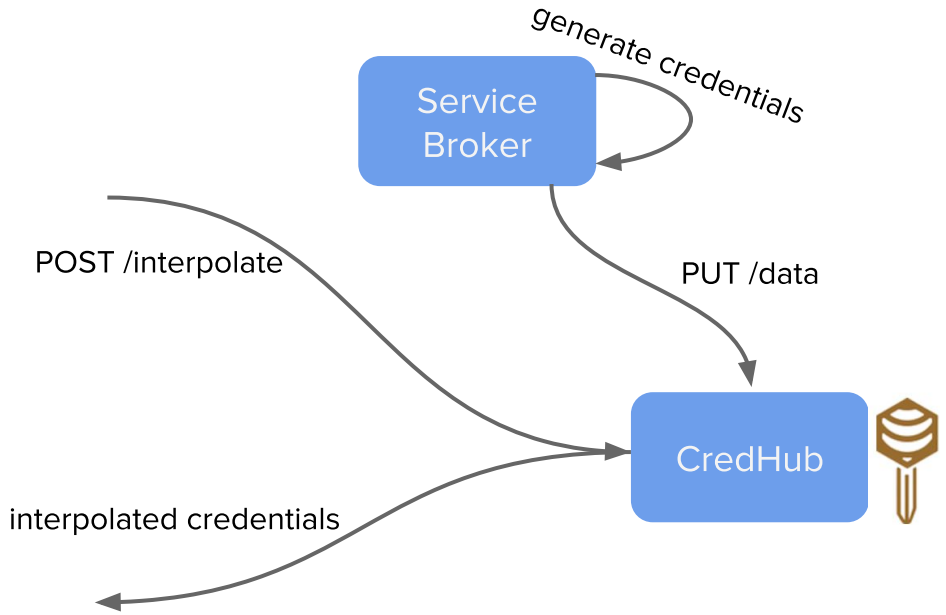
PCF Locks Down Application Containers



CredHub - Cradle to Grave Credentials Management

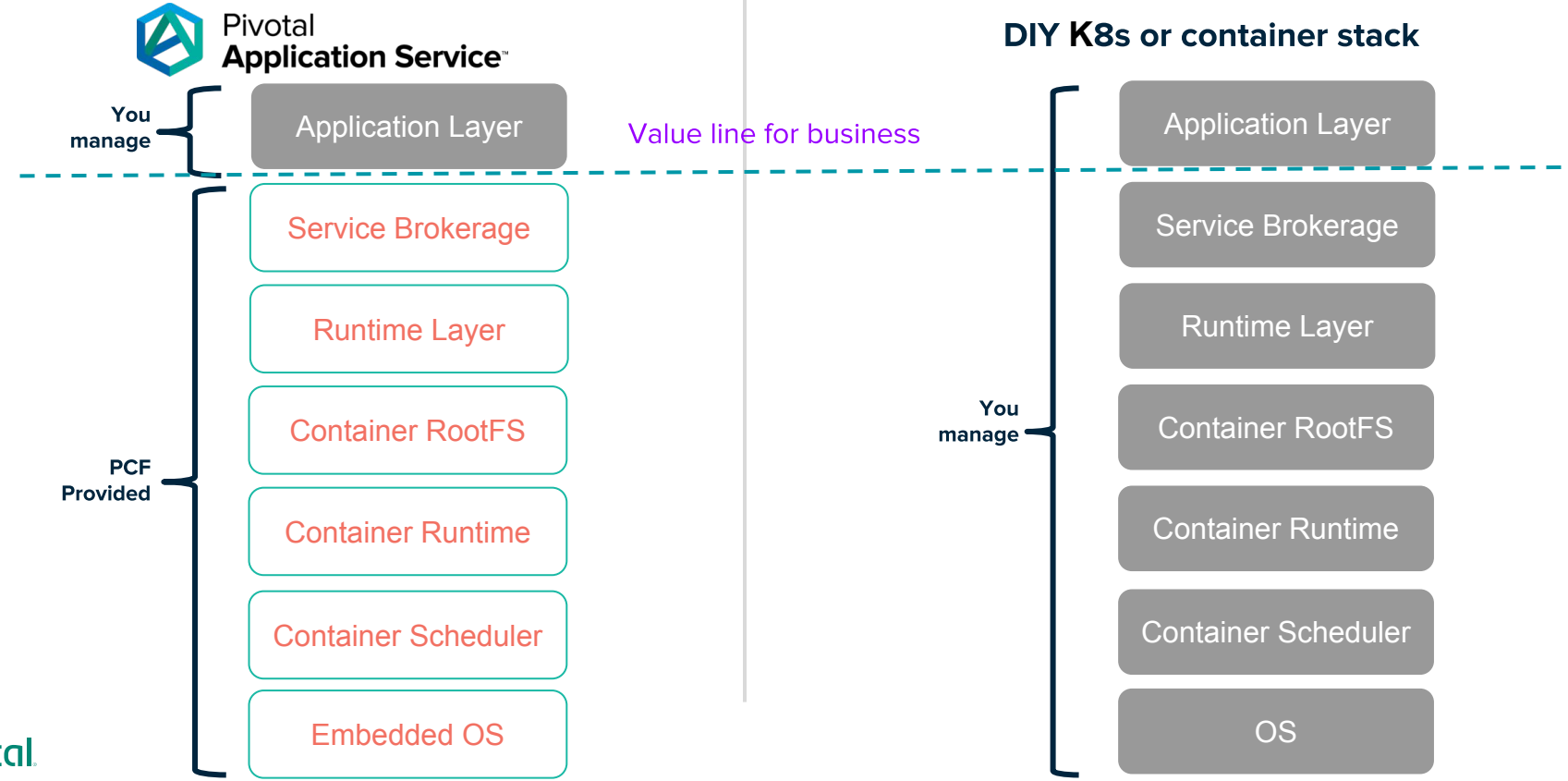
```
“VCAP_SERVICES”: {
  “my-service”: [{
    “credentials”: {
      “credhub-ref”: “/c/my-broker/1111/2222/credentials”
    },
  }]
}
```

```
“VCAP_SERVICES”: {
  “service-name”: [{
    “credentials”: {
      “uri”: “https://service-6yQVNrhZVP.example.com”,
      “username”: “VofTuQk2BH”,
      “password”: “fRqah7Wygi”
    },
  }]
}
```



PCF creates & interpolates credentials → no developer overhead, mitigate risk of accidental leaks

Decide what you want to own from ‘security’ perspective



Microservices Security Patterns

Pivotal

Work with your InfoSec to design and develop secured µServices

API Gateway in DMZ

- Use cases
  - Centralized authentication
  - Address common threats
    - Input validation
    - SQL injection
  - Mitigating DDOS attacks
- Key benefits
  - Separate the cross cutting concerns
  - Let security experts handle it and iterate over it without impact developer productivity
- Things to remember
  - Use Opaque token as this is exposed on the wild web
  - Downstream/on platform service can retrieve ID Token for the given opaque token
  - Make sure expiry of ID token doesn’t last more than opaque token

API as a µService facade

- Use cases
  - Token exchange e.g. for accessing legacy apps
  - Enforce rate limiting
  - Coarse grained access control
  - Gather performance & usage metrics
  - Response aware business metrics
- Key benefits
  - Let security experts handle and iterate over security constructs
  - Application developers can focus on business value
  - Support hybrid use cases
  - Avoid tokens proliferation
- Things to remember
  - Use PCF C2C networking policies to block traffic to downstream apps
  - Use PCF's internal domain for downstream apps
  - Can use mTLS for more secured authentication
  - Ensure µService doesn't allow other access routes

µService handles AuthN/AuthZ

- Use cases
  - Each microservice needs to implement authentication
  - Fine grained authorization
  - Don’t want extra hop of the API facade
  - Developers want to implement all authorization closer to the business logic
  - Don’t have API platform that can implement cross-cutting concerns and implementing a DIY API gateway is more costly than desired benefits
- Benefits
  - Better latency
  - No learning curve to implement custom API gateway and then maintain it
- Things to remember
  - Use PCF C2C networking policies to block traffic to downstream apps
  - Can use mTLS for more secured authentication
  - Block all other callers
  - Use Access Tokens for fine grained access control

Greenfield Apps Using Legacy IDP

- Mix of legacy and greenfield Apps/APIs OR ecosystem of multiple IDPs
  - Legacy API expects SAML assertion from an external IDP
  - Greenfield API expects OAuth token from UAA
- Flow
  - External IDP registers an OAuth client
  - grant_type = “urn:ietf:params:oauth:grant-type:saml2-bearer” or “urn:ietf:params:oauth:grant-type:jwt-bearer”
  - Scopes -> Relevant scopes needed in access token
  - Request UAA Access Token
    - Pass client id and secret
    - JWT or SAML assertion
    - Response = UAA access token

Transforming How The World Builds & Runs Software