Identity@CF

User Account and Authentication

Agenda

- Who we are
- What we build
- Why it’s important
- How to use it

CloudFoundry

- Open Source PaaS (Platform as a service)
  - Multiple framework support - Java, Ruby, Node, Scala
  - Multiple application services – MySQL, Postgres, Mongo, Redis, Rabbit
- Cloudfoundry.org
- https://github.com/cloudfoundry
- Cloudfoundry.com – Cloudfoundry service hosted by Vmware
- Other cloudfoundry instances are hosted by AppFog, ActiveState and Tier3

Who we are

- Identity team
  - Dale Olds
  - Dave Syer
  - Luke Taylor
  - Joel D'sa
  - Vidya Valmikinathan

vcap-dev@cloudfoundry.org

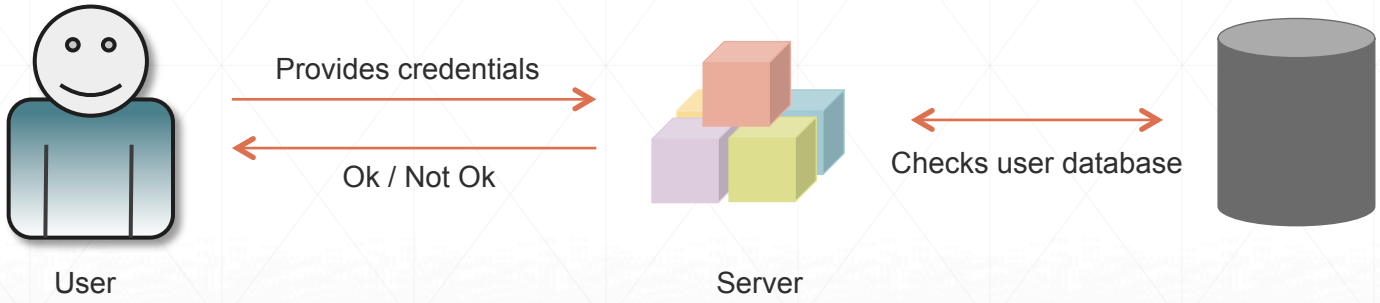
What we build

- **UAA** – User authentication and authorization server
- **Spring Security Oauth2** – Spring project that supports UAA features
- **uaac** – Command line api client for the UAA
- **Authentication servers** to support
  - External authentication sources (google, yahoo, github, linkedin),
  - Enterprise identity - SAML2

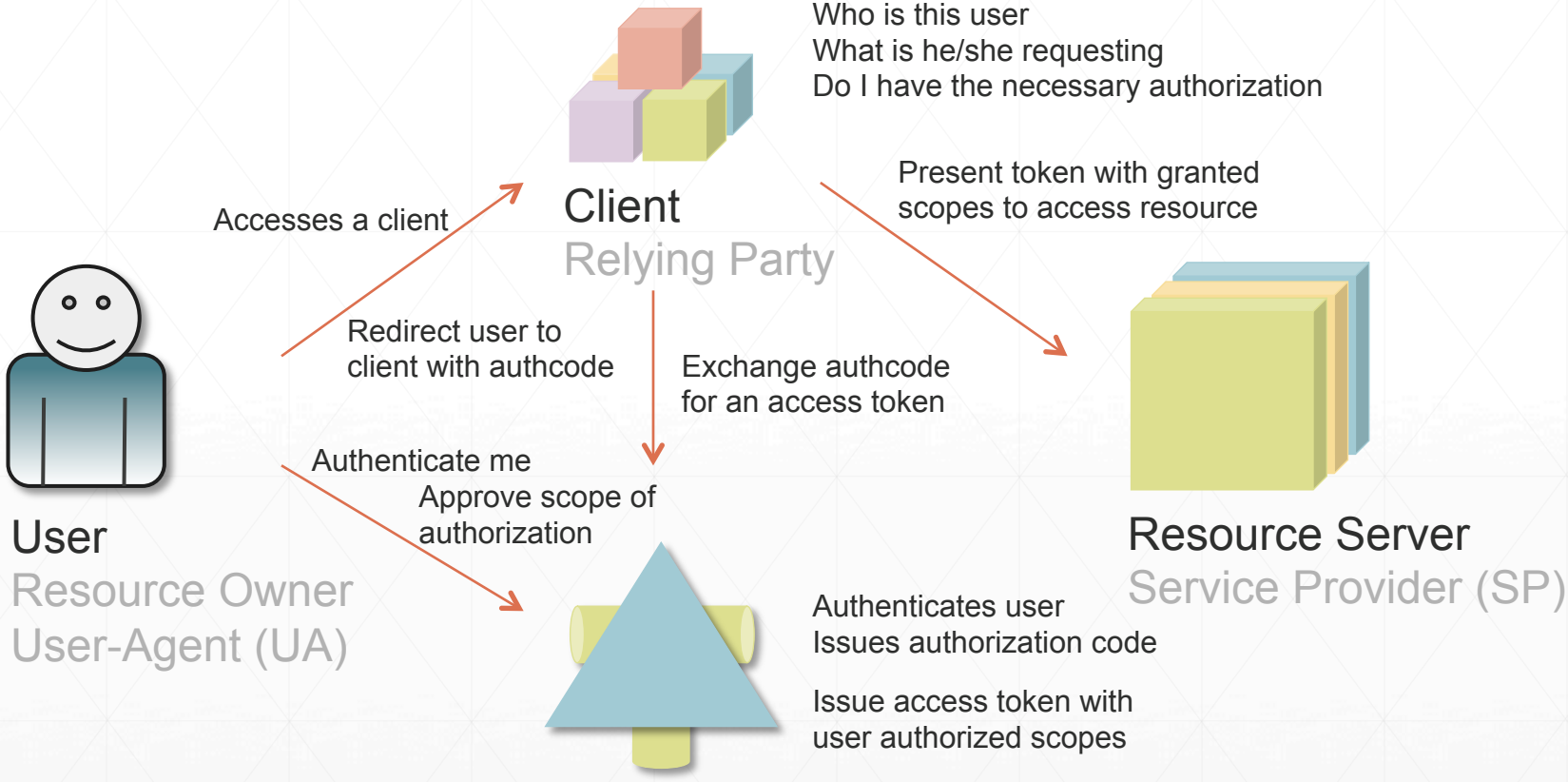
Why is it important

- **Higher degree of trust** – Credentials are accepted only by a trusted source
- **Standards based** – Consistent, proven API, process and interactions that users are comfortable with
- Trustworthy interactions between the **user and the platform**
- Trustworthy interactions **between components**
- Simple third party participation to **extend the platform**

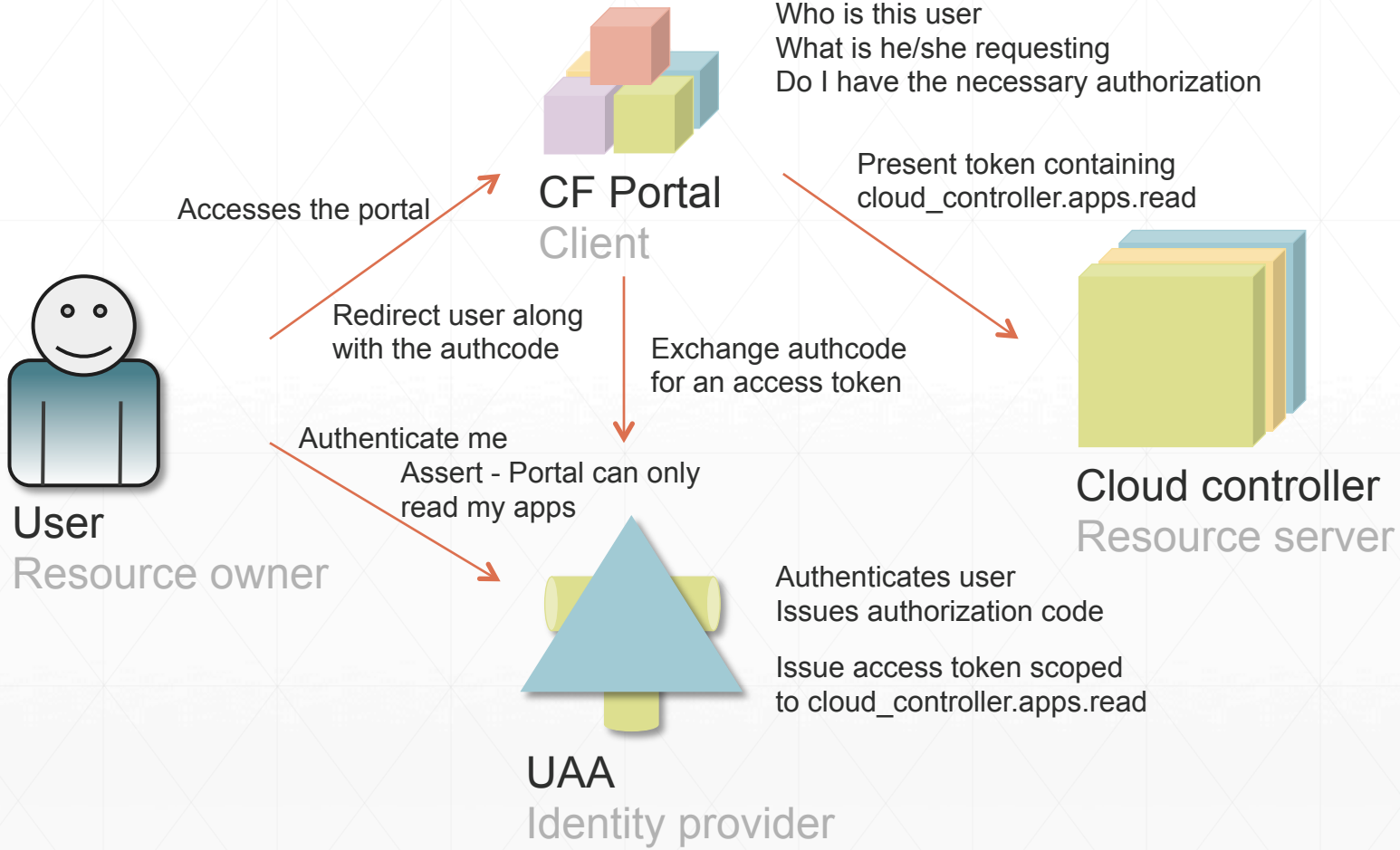
Traditional approach to authentication



Oauth2 authorization code flow

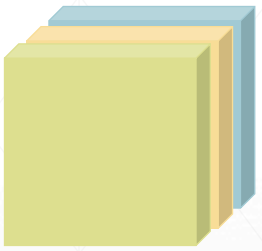


Oauth2 for Cloud Foundry



Oauth2 for Cloud Foundry

CF Portal
Client



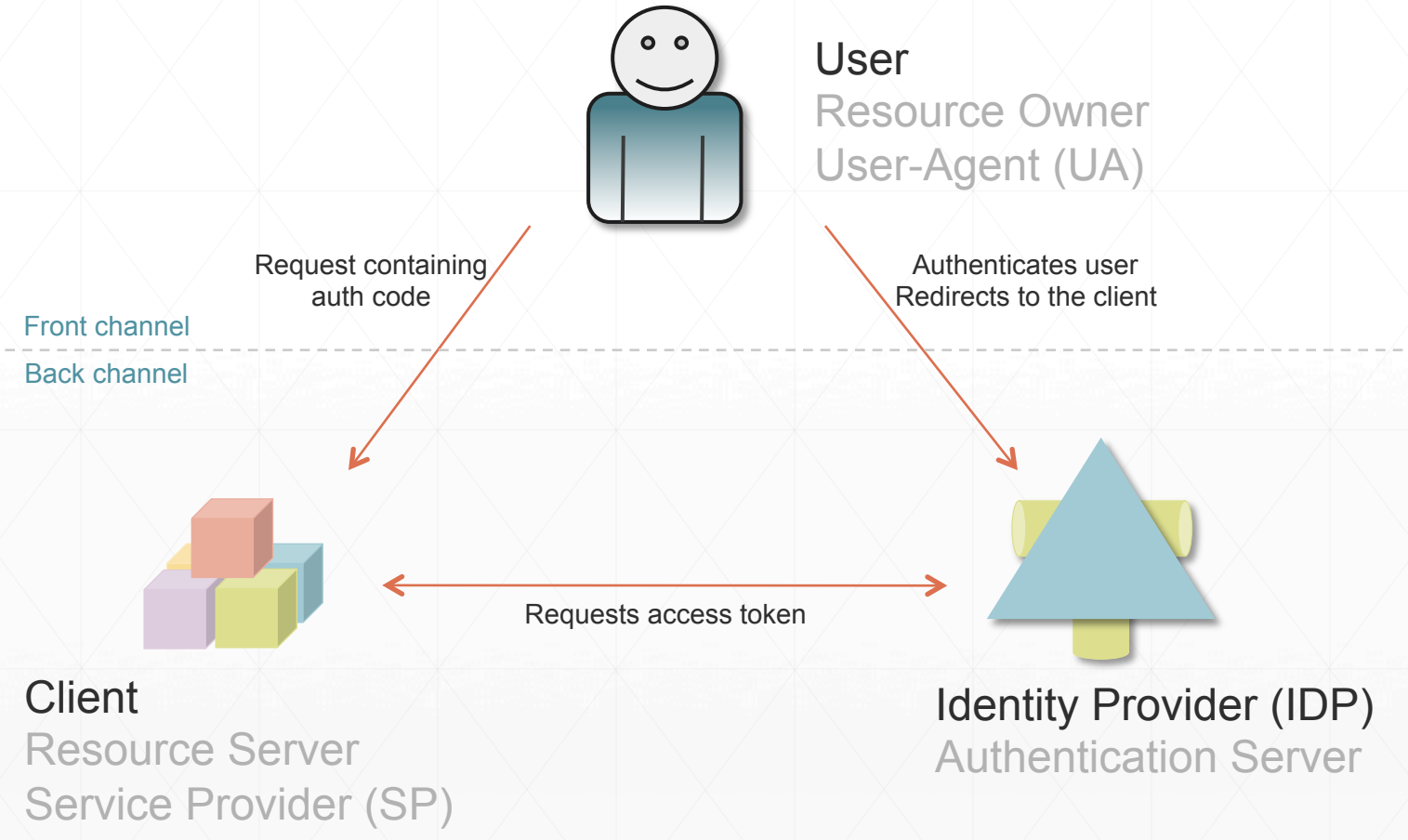
Cloud controller
Resource server

vmc
STS
Wavemaker
3rd parties (Appsecute etc.)

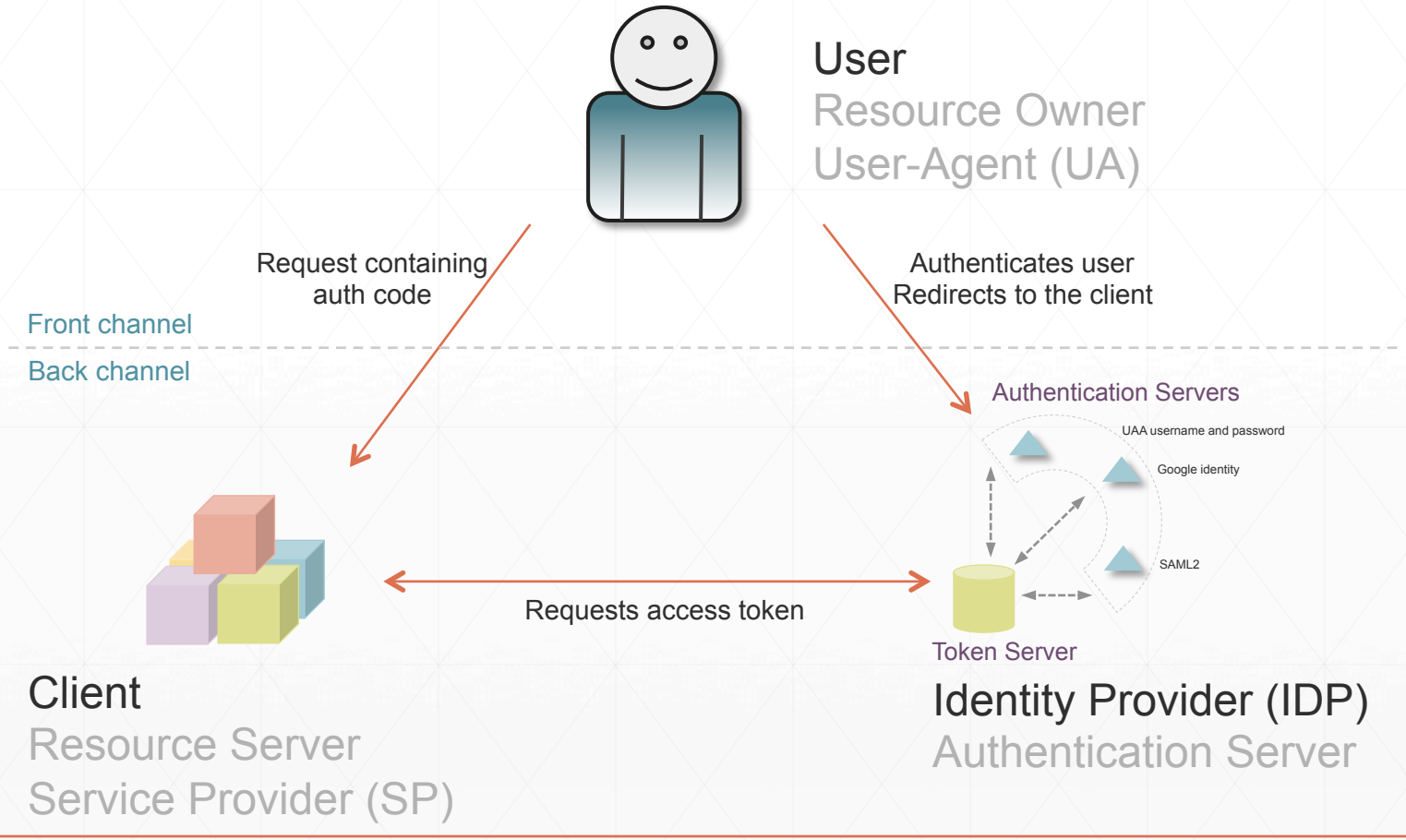
Dashboard
Service Gateways
Health Manager

Collector
Dashboard
Director

Oauth2 authentication



Oauth2 authentication



UAA Features

- Authenticates users from multiple sources
- Presents a single standard protocol for consumers
- OpenID Connect and Oauth2 – delegated authorization
- SCIM – user management

Demo

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Consumer of the user's identity
- Provides a service or needs access to a resource
- Can act by itself or on behalf of users
- Clients are registered with the UAA
  - client id and secret
  - authorized grant type
    - authorization_code
    - client_credentials
    - implicit, owner_password
  - scopes and authorities
  - redirect uri
  - token expiry

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Set of strings
- Each string represents a permission that client can request on behalf of a user
- Examples:
  - cloud_controller.admin
  - password.write
  - openid
- Requestable scopes configured during client registration
- Subset of the scopes granted to the client must be authorized by a user (authorization code flow)

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Set of strings
- Represents a permission that a client can be granted when it acts on it's own
- Examples:
  - scim.write
  - portal.users.read
- Configured during client registration

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Client credentials - client id and secret
- Client id is added to authorization requests
- Credentials are used to authenticate token requests

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- The user agent is redirected from the UAA to the client along with authorization codes or credentials
- Redirect URI for a client are registered with the UAA to prevent fraudulent redirections
- Request URI must match the registration

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Internal users are stored in the UAA database
- Users are provisioned using the SCIM API
- UAA has the ability to consume external identities
- Clients can also be users

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- UAA users are authenticated using username (email address) and password
- Authentication is represented by an “access token” (bearer token) that contains the set of the user’s scopes
- Flexibility to support any form of authentication
  - External user databases like LDAP
  - External authentication protocols, Incoming OpenID, SAML
- Goal is to fully implement OpenID Connect

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Authorization information is contained in an Oauth2 bearer token
- Client acting as itself
  - Uses a client credentials grant to request authorization
  - Scopes granted are the same as the registered “authorities”
- Client acting on behalf of the user
  - Uses an authorization code grant to request authorization
  - Scopes granted

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- Token is a standard JWT (JSON web token)
- Three parts separated by periods
- header.content.signature
- Production UAA tokens are signed using a shared secret. This is changing to use public key signing.


```
{
   "exp":1349467969,
   "user_name":"jdsa@vmware.com",
   "scope":[
      "cloud_controller.read",
      "cloud_controller.write",
      "openid"
   ],
   "email":"jdsa@vmware.com",
   "aud":[
      "openid",
      "cloud_controller"
   ],
   "jti":"9d82c1c2-94cd-4433-a8e5-19549dccaed2",
   "user_id":"cf9d5fdb-6433-4c41-b61c-cc9fda937620",
   "client_id":"vmc"
}
```

Oauth2 with the UAA

- Clients
  - Scopes
  - Authorities
  - Client credentials
  - Redirect URI
- Users
- User Authentication
- Authorization
- Tokens
  - Signatures

- All tokens are signed using a shared secret.
- Public key signing is currently available.

SCIM

- JSON API for user management
- Simple cloud identity management (now system for inter-domain identity management)

The future

- Full support for ID tokens (OpenID Connect)
- Expanded SCIM support
- IdaaS

Q&A

Addidional topics

- password strength (aaS)