Multi-Cloud End-User Identity and Access Management with Zero Redeploys and Code

Anton Aleksandrov
Chief Architect,
IBM Cloud Application Identity

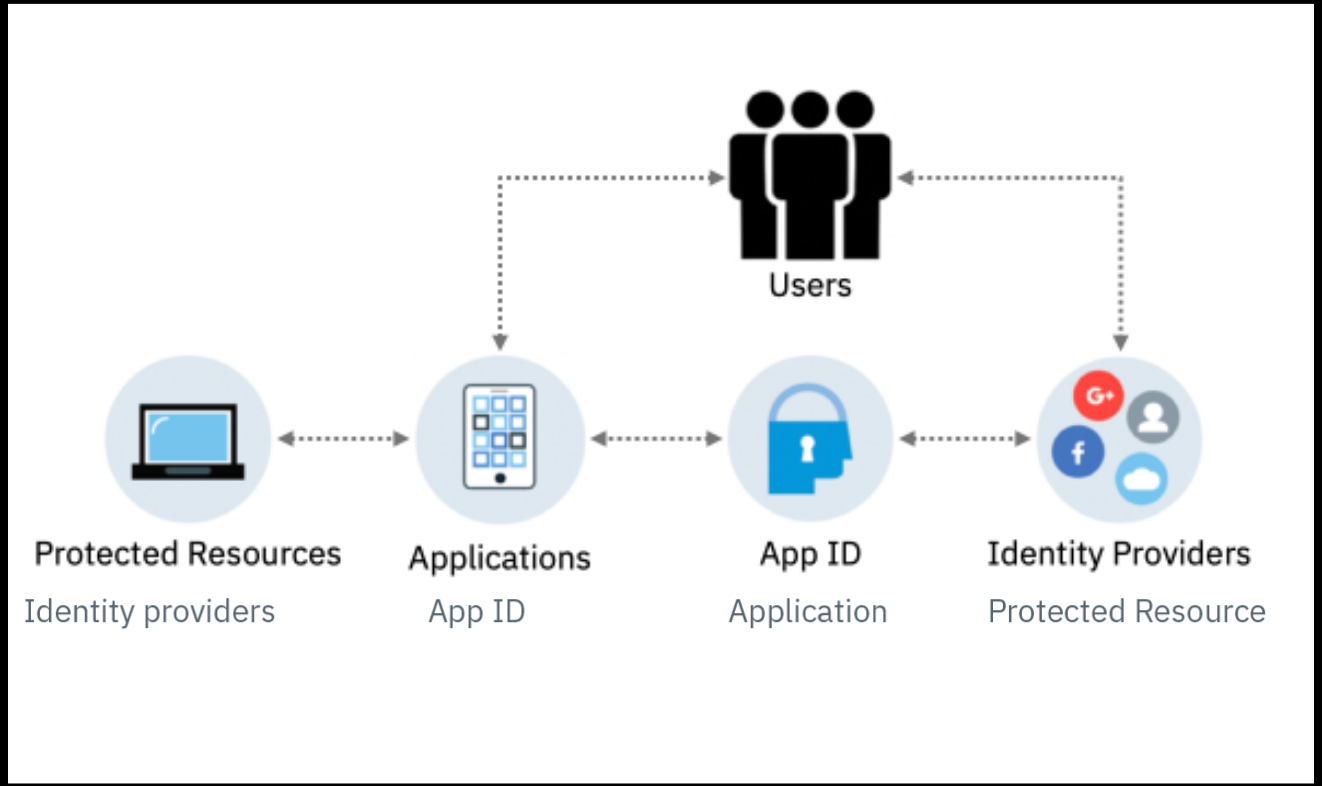
IBM Cloud

Disclaimer

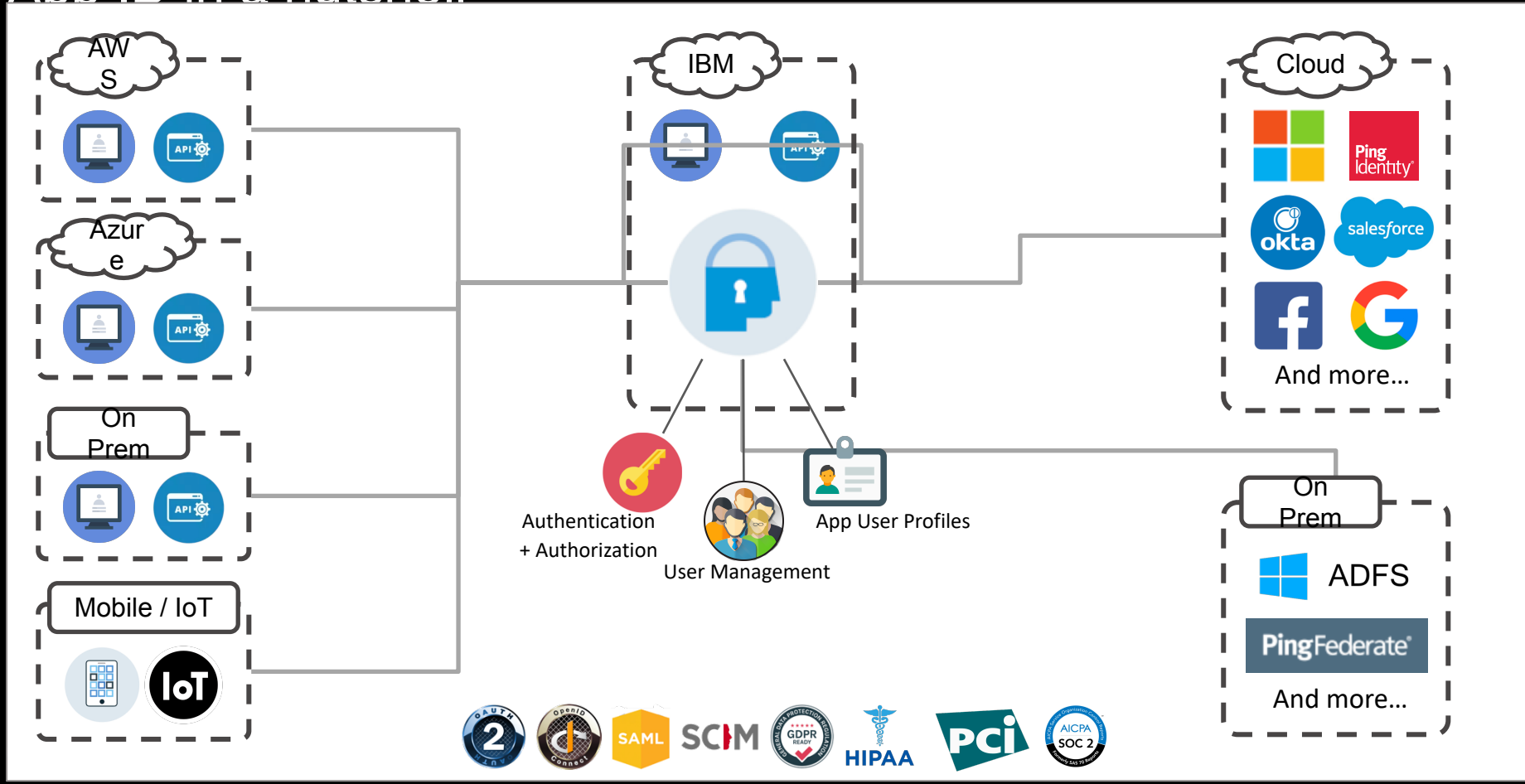
- **Notice: Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsibility for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations. The products, services, and other capabilities described herein are not suitable for all client situations and may have restricted availability. IBM does not provide legal, accounting or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.**
- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

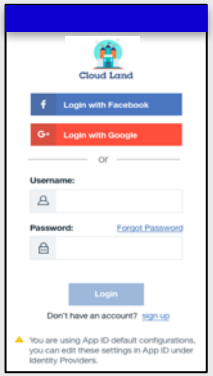
App ID in a nutshell



App ID in a nutshell



App ID Value to Developers of any app on any cloud



Add authentication to your mobile and web apps and protect your APIs and back-ends running on cloud. Add email/password-based sign-up and sign-in with App ID's scalable user registry - Cloud Directory, or social log-in. For employee apps, use SAML 2.0 federation for enterprise sign-in. For all app users, enrich their profiles with additional info so you can build engaging experiences.

Authentication / Authorization

- Email/Password, Social sign-in, Enterprise sign-in
- Use Client SDKs for iOS and Android, Server SDKs for node.js and Swift, REST APIs called from any language, a customizable sign-in widget, and App ID starter app
- Secure your apps running on Kubernetes or your managed APIs with no code changes
- Open standards based (OAUTH2, OIDC, SCIM, SAML 2.0)

User Management

- Sign-in, sign-up, email verification, change password, forgot password
- Default UI and flows, or replace with your own branding and custom flows
- Default email templates you can customize and brand

Application User Profile Management

- Store end user data, like app preferences, to personalize app experiences
- Leverage data from user repositories, or add your own custom attributes
- Continuity for users who start out anonymously and sign-in later

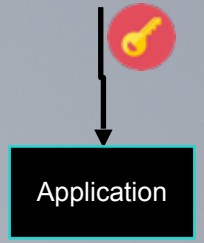
Know your friends

- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**

- OAuth2- https://tools.ietf.org/html/rfc6749
- OIDC - https://openid.net/developers/specs
- Bearer Token - https://tools.ietf.org/html/rfc6750
- JWT - https://tools.ietf.org/html/rfc7519

Know your friends

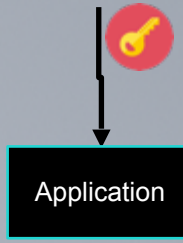
- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**



- **OAuth2- https://tools.ietf.org/html/rfc6749**
- **OIDC - https://openid.net/developers/specs**
- **Bearer Token - https://tools.ietf.org/html/rfc6750**
- **JWT - https://tools.ietf.org/html/rfc7519**

Know your friends

- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**

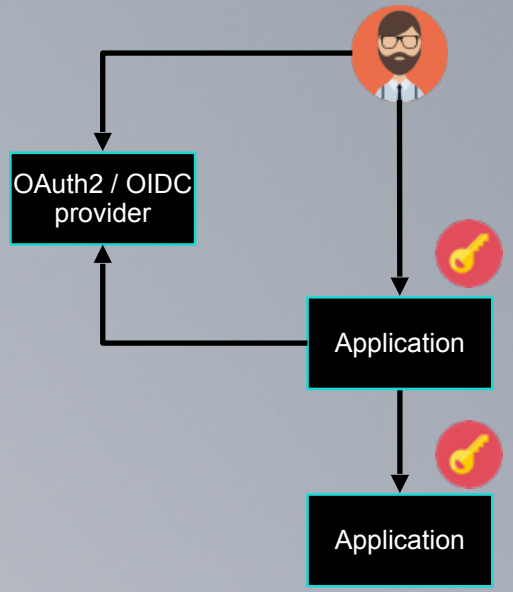


- OAuth2- https://tools.ietf.org/html/rfc6749
- OIDC - https://openid.net/developers/specs
- Bearer Token - https://tools.ietf.org/html/rfc6750
- JWT - https://tools.ietf.org/html/rfc7519


```
{
    iss: "appid-oauth.ng.bluemix.net",
    exp: 1542231804,
    aud: "d5966856-a4d1-4e0c-86d3-f6b45a66818a",
    sub: "35e32cac-2a26-4cdd-b24e-1e3b3d704e55",
    amr: [
        "cloud_directory"
    ],
    iat: 1542228203,
    tenant: "f30e77ae-599c-4833-98b9-4fb7728caa1e",
    scope: "openid appid_default appid_readprofile appid_readuserattr
}
```

Know your friends

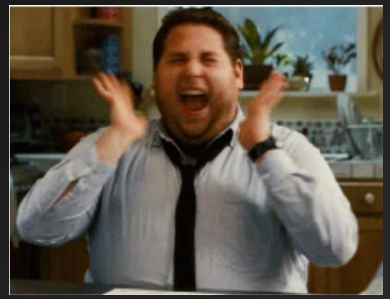
- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**

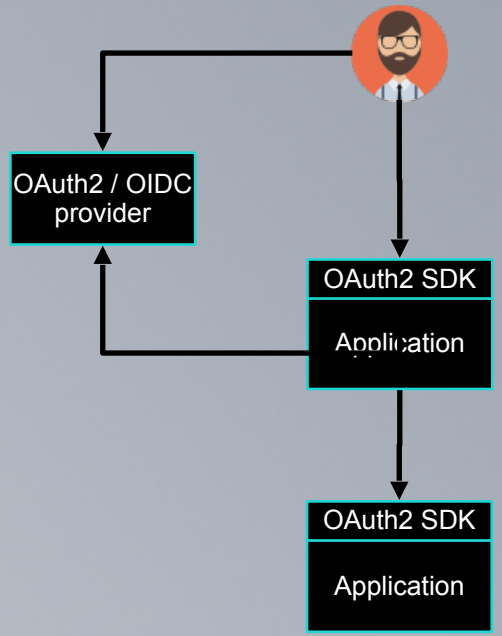


- OAuth2- https://tools.ietf.org/html/rfc6749
- OIDC - https://openid.net/developers/specs
- Bearer Token - https://tools.ietf.org/html/rfc6750
- JWT - https://tools.ietf.org/html/rfc7519

Know your friends

- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**





- OAuth2- https://tools.ietf.org/html/rfc6749
- OIDC - https://openid.net/developers/specs
- Bearer Token - https://tools.ietf.org/html/rfc6750
- JWT - https://tools.ietf.org/html/rfc7519

Configuration

```
passport.use(new BearerStrategy(
  function(token, done) {
    User.findOne({ token: token }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      return done(null, user, { scope: 'read' });
    });
  }
));
```

The verify callback for bearer tokens accepts the `token` as an argume
can be passed, which will be set by Passport at `req.authInfo`. This i
token, and can be used when making access control checks.

Protect Endpoints

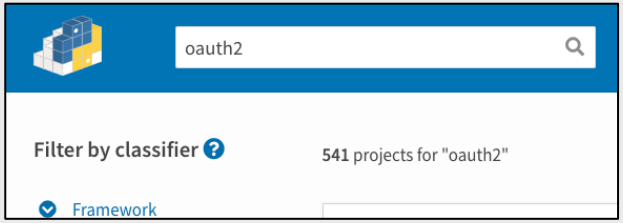
```
app.get('/api/me',
  passport.authenticate('bearer', { session: false }),
  function(req, res) {
    res.json(req.user);
  });
```

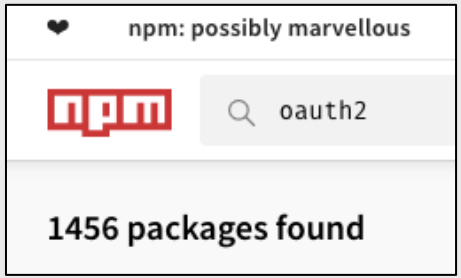
SocialApplication

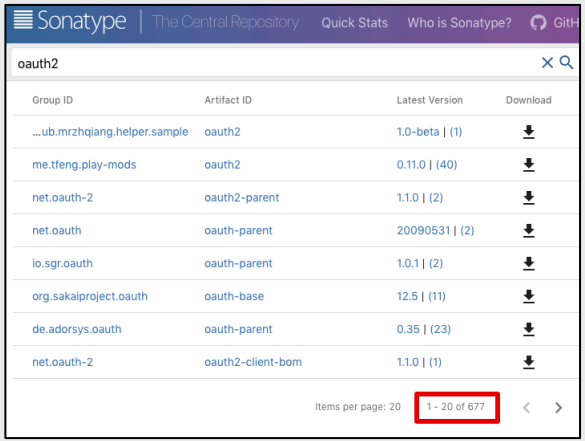
```
@SpringBootApplication
@EnableOAuth2Sso
@RestController
public class SocialApplication extends WebSecurityConfigurerAdapter

  ...

  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http
      .antMatcher("/**")
      .authorizeRequests()
        .antMatchers("/", "/login**", "/webjars/**", "/error**")
        .permitAll()
      .anyRequest()
        .authenticated();
  }

}
```

```
consumer_key = 'my_key_from_twitter'
consumer_secret = 'my_secret_from_twitter'

request_token_url = 'https://api.twitter.com/oauth/request_token'
access_token_url = 'https://api.twitter.com/oauth/access_token'
authorize_url = 'https://api.twitter.com/oauth/authorize'

consumer = oauth.Consumer(consumer_key, consumer_secret)
client = oauth.Client(consumer)

# Step 1: Get a request token. This is a temporary token that is used for
# having the user authorize an access token and to sign the request to obta
# said access token.

resp, content = client.request(request_token_url, "GET")
if resp['status'] != '200':
    raise Exception("Invalid response %s." % resp['status'])

request_token = dict(urlparse.parse_qsl(content))
```

```
// Requests an OAuthToken using a "code" type
func GetOauthToken(r *http.Request) (*oauth2.Token, error

        log.Println("Getting auth token.")

        ctx := context.Background()

        if ctx == nil{
                return nil, errors.New("Could not get con
        }

        if r.URL.Query().Get(STATE) != STATE {
                return nil, errors.New("State value did n
        }

        // Exchange code for OAuth token
        oauth2Token, oauth2TokenError := conf.Exchange(ct
        if oauth2TokenError != nil {
                return nil, errors.New("Failed to exchang
        }

        return oauth2Token, nil
}

// Requests a user profile, using a bearer token
func GetUserProfile(r *http.Request, token oauth2.Token)

        log.Println("Getting user profile.")

        ctx := context.Background()

        if ctx == nil{
                return nil, errors.New("Could not get con
        }

        // Getting now the userInfo
        client := conf.Client(ctx, &token)

        // Get request using /userinfo url
        userinfoResponse, userinfoError := client.Get(str
        if userinfoError != nil {
                return nil, errors.New("Failed to obtain
        }

        defer userinfoResponse.Body.Close()
```


oauth2
Filter by classifier
541 projects for "oauth2"
Framework


npm: possibly marvellous
npm
oauth2
1456 packages found

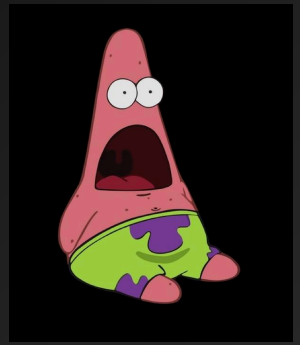

Sonatype
The Central Repository
Quick Stats
Who is Sonatype?
oauth2
Group ID
Artifact ID
Latest Version
Download
...ub.mrzhqiang.helper.sample
oauth2
1.0-beta | (1)
me.tfeng.play-mods
oauth2
0.11.0 | (40)
net.oauth-2
oauth2-parent
1.1.0 | (2)
net.oauth
oauth-parent
20090531 | (2)
io.sgr.oauth
oauth-parent
1.0.1 | (2)
org.sakaiproject.oauth
oauth-base
12.5 | (11)
de.adorsys.oauth
oauth-parent
0.35 | (23)
net.oauth-2
oauth2-client-bom
1.1.0 | (1)
Items per page: 20
1 - 20 of 677


Libraries for Token Signing/Verification
FILTER BY
Warning: Critical vulnerabilities in JSON Web Token libraries with asymmetric keys. Learn more.
.NET
Python
Node.js
Java
MINIMUM VERSION 1.0.1
MINIMUM VERSION 4.2.2
Sign
Verify
iss check
sub check
aud check
exp check
nbf check
iat check
jti check
HS256
HS384
HS512
RS256
RS384
RS512
ES256
ES384
ES512
View Repo

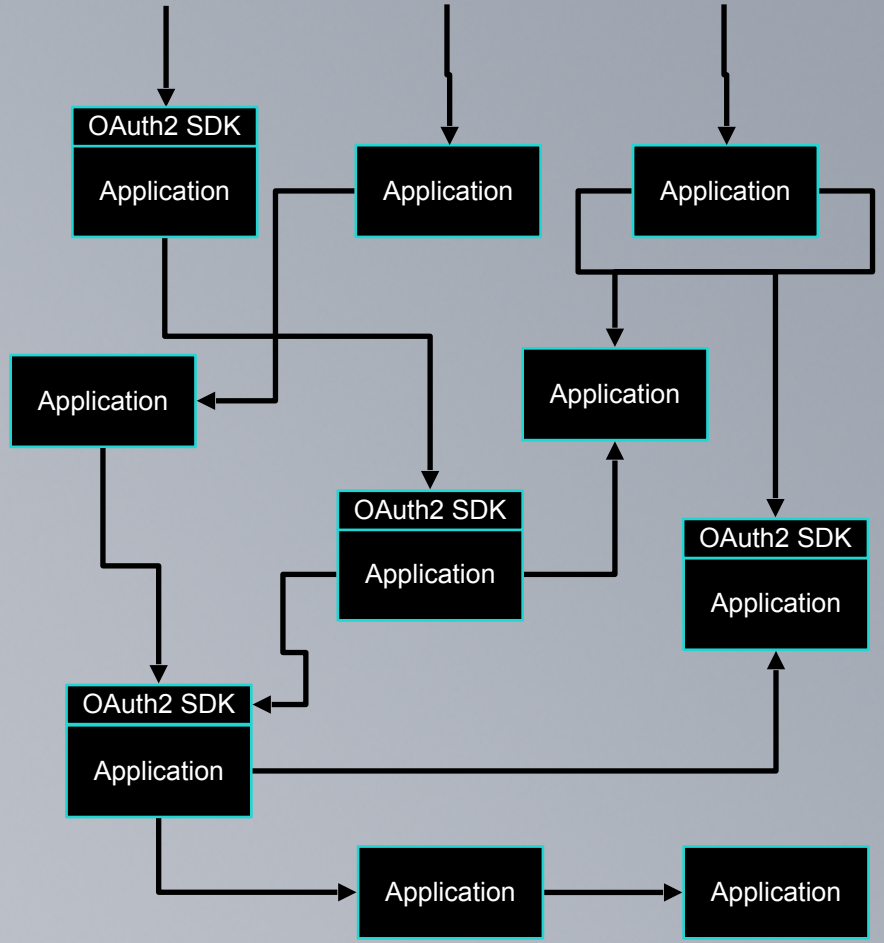

Go
Groovy
Haskell
Haxe
Rust
Sign
Verify
iss check
sub check
aud check
exp check
nbf check
iat check
jti check
HS256
HS384
HS512
RS256
RS384
RS512
ES256
ES384
ES512
View Repo
nickvellios
gbrlsnchs
Pascal de Kloe
Seth Moeckel
Kaleidos
Stefan Saasen
Fraser Tweedale
Kevin Leung
GildedHonour

Know your friends

- Authorize with **OAuth2**
- Authenticate with **OIDC**
- Access with **Bearer Token**
- Get insights from **JWT**



- OAuth2- https://tools.ietf.org/html/rfc6749
- OIDC - https://openid.net/developers/specs
- Bearer Token - https://tools.ietf.org/html/rfc6750
- JWT - https://tools.ietf.org/html/rfc7519



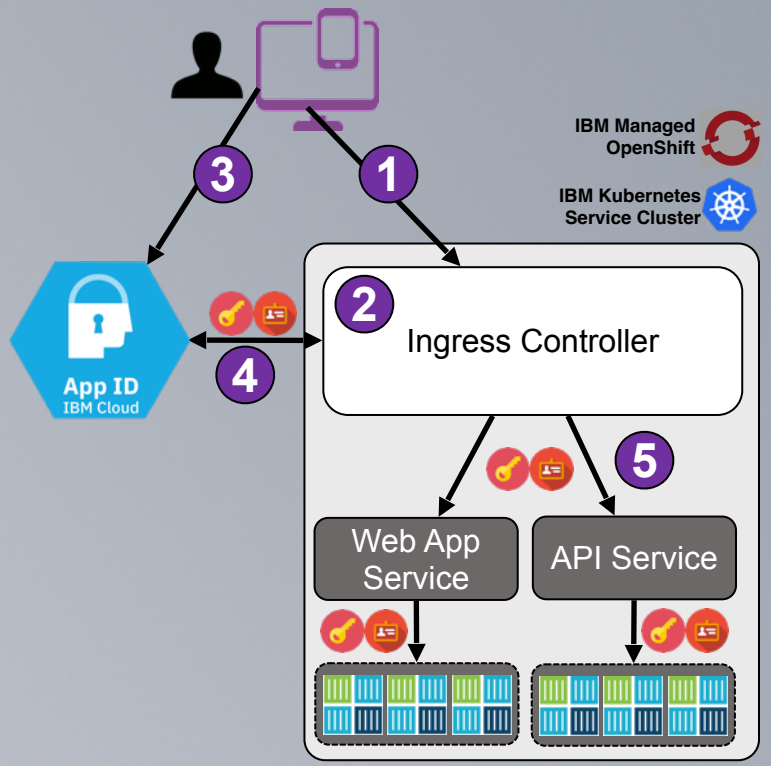
Consistently enforce policy-driven authentication

Secure Apps on Kubernetes / Open Shift – No code change is required

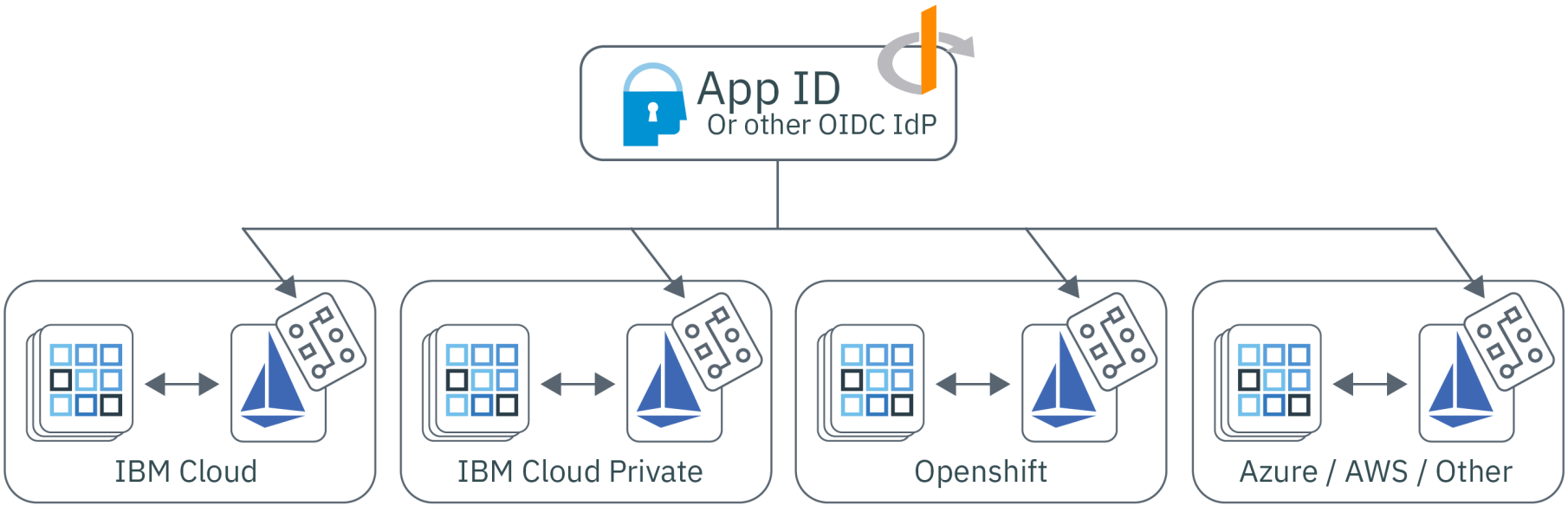
Consistently enforce policy-driven authentication to your apps, *μ*Services and APIs using declarative security.

Security is enforced, using simple annotations, at the Application Edge Gateway (Ingress) to ensure user authentication/autherization and token validation.

1. Request is sent to Web app or API
2. Ingress Controller either validates supplied tokens (API flow) or starts a 3-leg OIDC authentication process (Web app flow)
3. User Authenticates with App ID
4. App ID access and identity tokens are received and validated by Ingress Controller
5. Request containing access and identity tokens is forwarded to Kubernetes pods



App Identity and Access Istio Mixer Adapter



App Identity and Access Istio Mixer Adapter

```yaml
apiVersion: "security.cloud.ibm.com/v1"
kind: Policy
metadata:
    name: my-sample-backend-policy
    namespace: sample-namespace
spec:
    targets:
    - serviceName: <kubernetes-service-name-to-protect>
        paths:
        - prefix: /api/files
            method: ALL
            policies:
            - policyType: jwt
                config: my-oidc-provider-config
                rules: // optional
                - claim: iss
                    match: ALL
                    source: access_token
                    values:
                    - <expected-issuer-id>
                - claim: scope
                    match: ALL
                    source: access_token
                    values:
                    - files.read
                    - files.write
```

App Identity and Access Istio Mixer Adapter

Policy targets

Policy rules

```yaml
apiVersion: "security.cloud.ibm.com/v1"
kind: Policy
metadata:
    name: my-sample-backend-policy
    namespace: sample-namespace
spec:
    targets:
    - serviceName: <kubernetes-service-name-to-protect>
        paths:
        - prefix: /api/files
            method: ALL
            policies:
            - policyType: jwt
                config: my-oidc-provider-config
                rules: // optional
                - claim: iss
                    match: ALL
                    source: access_token
                    values:
                    - <expected-issuer-id>
                - claim: scope
                    match: ALL
                    source: access_token
                    values:
                    - files.read
                    - files.write
```


https://youtu.be/z1kCjxOw9Vs

Thank you

Anton Aleksandrov
Chief Architect,
IBM Cloud Application Identity Service
—
antona@us.ibm.com