

Securing Angular



Agenda

- Authentication / Authorization Basics
- JSON Web Tokens
- Authentication in Angular
- Authentication in ASP.NET Core

Authentication / Authorization Basics

Authorization

- Authorization is the function of specifying access rights/privileges to resources
- Authorization takes place when
 - Angular wants to consume an Api
 - Authentication between Angular + Api can be achieved using JWT
 - The user wants to navigate from one page (Route) to another
 - Typically done by using Route Guards

Authentication

- Authentication is the process of verifying that "you are who you say you are"
- Authentication can be done using
 - AuthO
 - JWT (Json Web Token)
- AuthO is supported by almost all cloud services like
 - Facebook
 - Google
 - Microsoft (Azure AD, Office 365)

JSON Web Tokens

JSON Web Tokens

- An open, industry standard RFC 7519 method for representing claims securely between two parties
- Can be sent through a URL, POST parameter, or inside an HTTP header
- Contains all the required information about the user
- Doku @ <https://jwt.io/>

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWVhbnR5dWV9Ij0wLCJpYXN0ZWQiOiJ1b3RlbnQ1IiwiaWF0Ij01NjM0NTY3ODkwfQ
```

Decoded

EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "sub": "1234567890",  "name": "John Doe",  "admin": true}
```

VERIFY SIGNATURE

HMACHSHA256(
base64UrlEncode(header) + ".", +
base64UrlEncode(payload),

secret

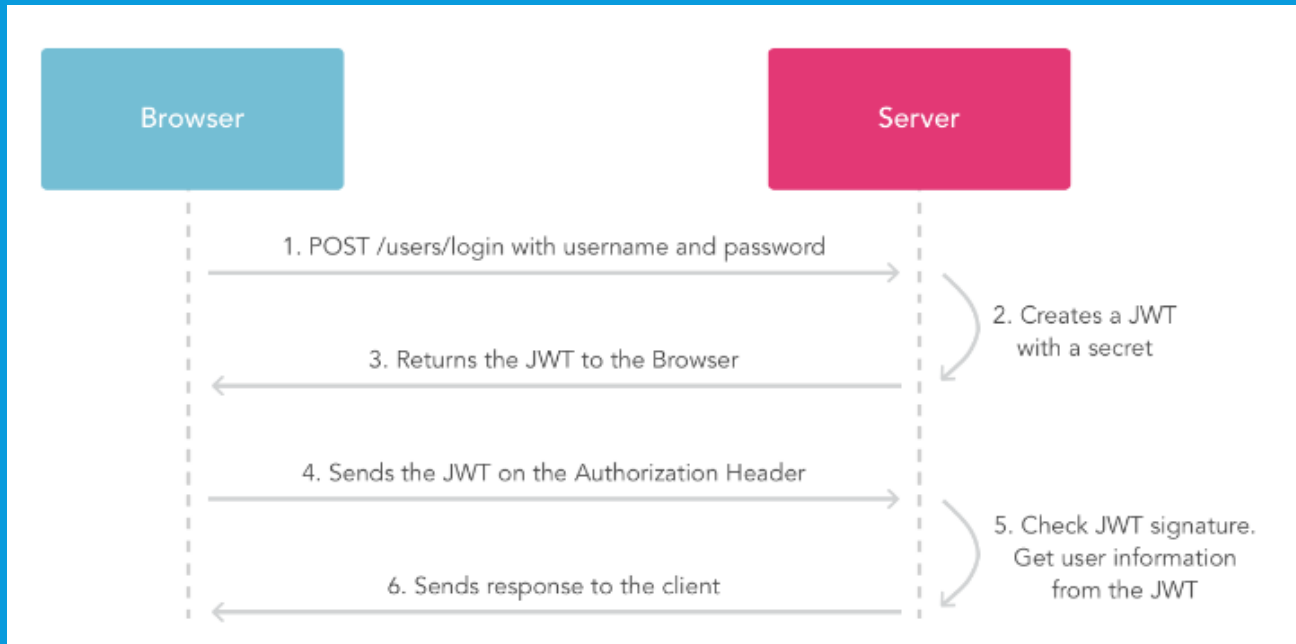
)

secret

 base64 encoded

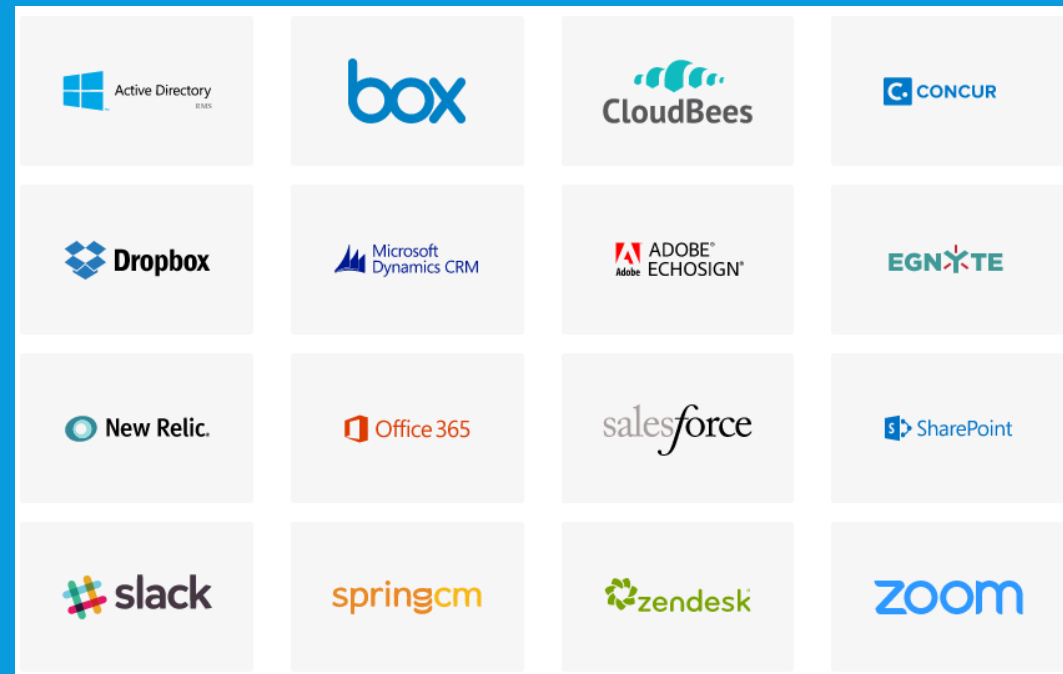
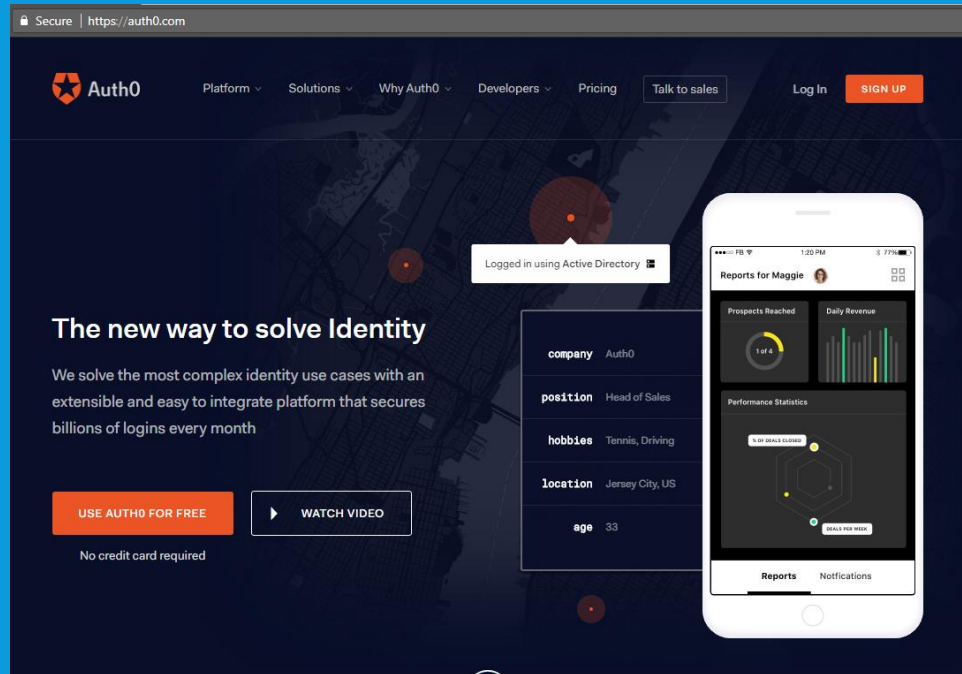
How do JSON Web Tokens work?

- Adds "Authorization: Bearer <token>" to header of Request



Auth0

- Auth0 provides free JSON Web Token Authentication
- Provides integrations to other Single Sign On Providers



Authentication in Angular

Authentication in SPAs

- SPAs use Tokens instead of Sessions / Cookies

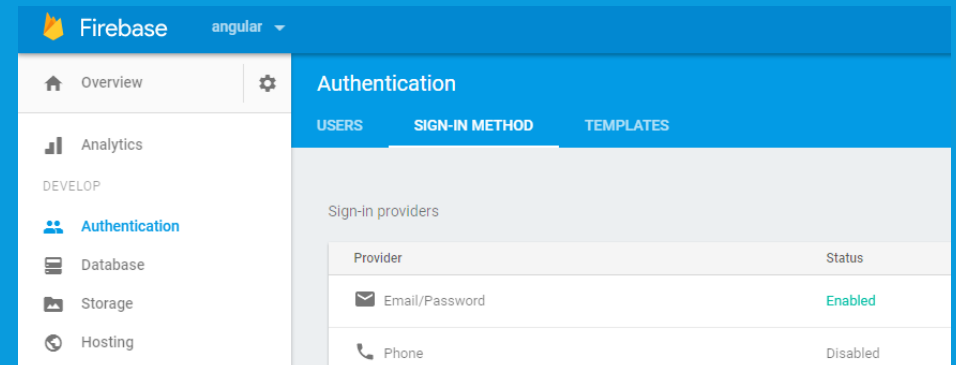
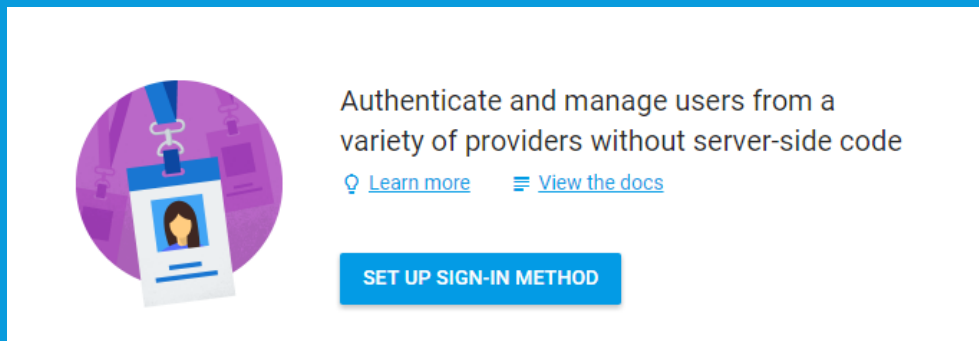


SignIn / SignUp Form

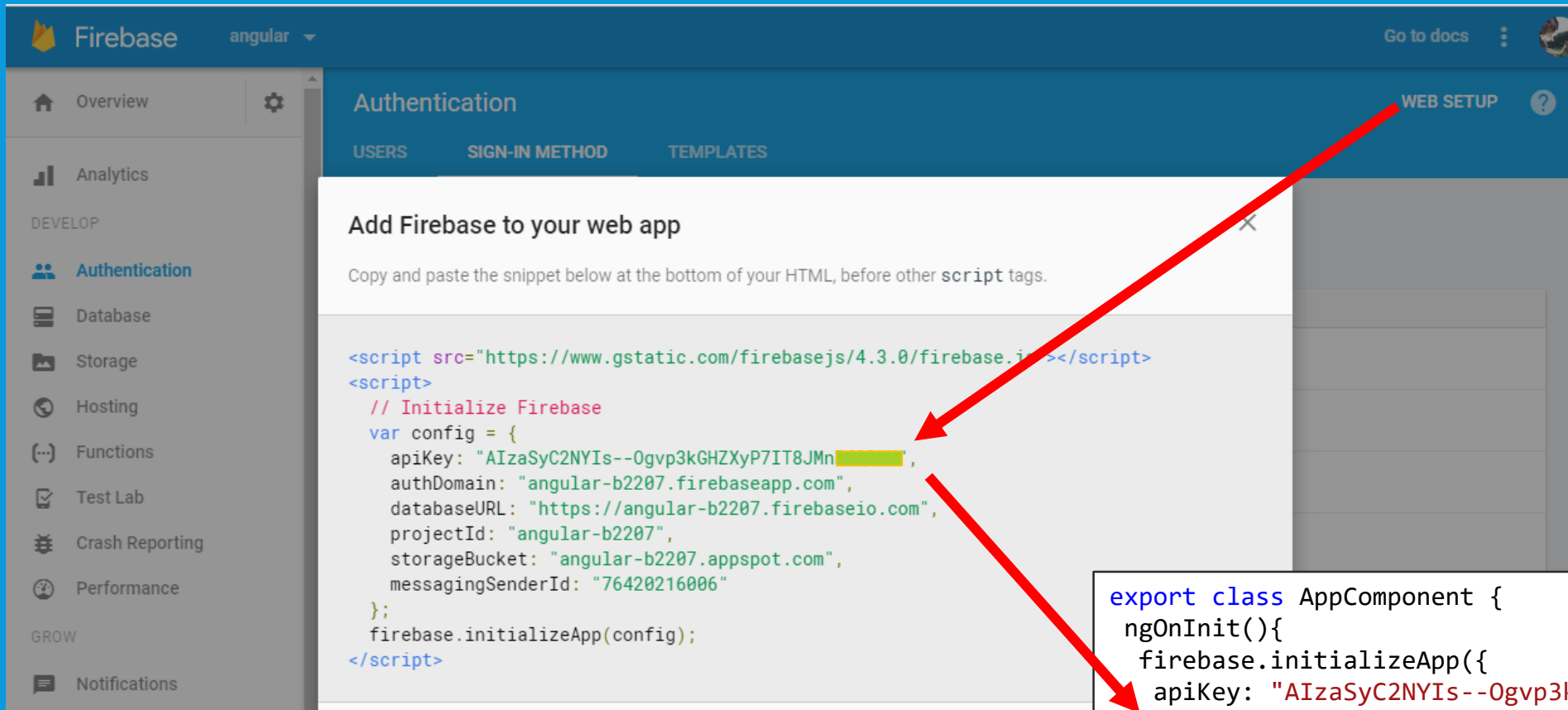
- Create two Components for
 - Signing in
 - Signing up
- Add them to Module
- Create Routes
- Install a JWT compatible SDK – ie Firebase
- Setup AuthService & AuthGuard

Firebase

- Firebase provides an easy to understand approach to get into using Tokens
- Does not mean that this has to be your primary Auth Method
- We are configuring Social, Windows, ... later on



Firebase Keys



The screenshot shows the Firebase console interface. On the left is a sidebar with navigation links: Overview, Analytics, DEVELOP (Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance), and GROW (Notifications). The main area is titled 'Authentication' and has tabs for 'USERS', 'SIGN-IN METHOD', and 'TEMPLATES'. A 'WEB SETUP' button is in the top right. A modal window titled 'Add Firebase to your web app' is open, containing the instruction: 'Copy and paste the snippet below at the bottom of your HTML, before other script tags.' Below the instruction is a code snippet for initializing Firebase. A red arrow points from the 'WEB SETUP' button to the code snippet, and another red arrow points from the 'apiKey' value in the snippet to a separate code block.

Authentication

USERS SIGN-IN METHOD TEMPLATES

WEB SETUP

Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other script tags.

```
<script src="https://www.gstatic.com/firebasejs/4.3.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyC2NYIs--Ogvp3kGHZxYP7IT8JMnRfQaaaa",
    authDomain: "angular-b2207.firebaseio.com",
    databaseURL: "https://angular-b2207.firebaseio.com",
    projectId: "angular-b2207",
    storageBucket: "angular-b2207.appspot.com",
    messagingSenderId: "76420216006"
  };
  firebase.initializeApp(config);
</script>
```

```
export class AppComponent {
  ngOnInit(){
    firebase.initializeApp({
      apiKey: "AIzaSyC2NYIs--Ogvp3kGHZxYP7IT8JMnRfQaaaa",
      authDomain: "angular-b2207.firebaseio.com"
    })
  }
}
```

Auth Service

- Implement an AuthService providing the following methods:
 - signupUser()
 - signinUser()
 - logout()
 - getToken()
 - isAuthenticated()

```
signupUser(email: string, password: string) {  
  firebase.auth().createUserWithEmailAndPassword(email, password)  
    .catch(  
      error => console.log(error)  
    )  
}
```

Auth Guard

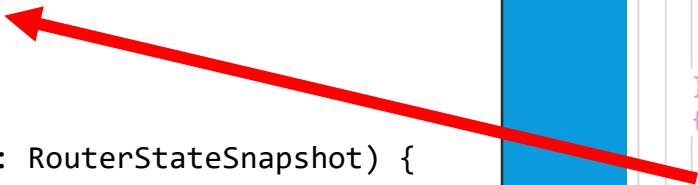
- Remember: AuthGuard is used to protect certain Routes

```
@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService) {}

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    return this.authService.isAuthenticated();
  }
}
```

```
const appRoutes: Routes = [
  { path: '',
    component: DemosComponent,
    children: [
      { path: 'promise', component: PromiseComponent },
    ]
  },
  { path: 'vouchers',
    component: VouchersComponent,
    canActivate: [AuthGuard]
  },
  { path: 'vouchers/:id',
    component: VoucherComponent,
    canActivate: [AuthGuard]
  },
  { path: 'accounts',
    component: AccountsComponent,
    canActivate: [AuthGuard],
    data: { title: 'Accounts' }
  },
  { path: 'signup',
    component: SignupComponent
  },
  { path: 'signin',
    component: SigninComponent
  }
];
```



Authentication in ASP.NET Core

Authorization

- Authorization Types
 - Simple Authorization – using Authorize attribute
 - Claim / Token based Authorization
- Ressources to Protect
 - Angular App
 - Web Api [Controller]

Authorize

- The [Authorize]-Attribute is used to require authorization for controllers
- Implemented in Microsoft.AspNetCore.Authorization
- Can be configured globally in Startup.cs -> Configure Services
- Can be set on
 - an individual method
 - a class

```
[Authorize]
[HttpGet]
public IEnumerable<Voucher> Get()
{
    var vouchers = ctx.Vouchers.OrderByDescending(v => v.Date).ToList();
    return vouchers;
}
```

Role based Authorization

- Role based authorization checks are declarative on
 - a controller
 - an action of an controller
- If your specify more than one attribute the user must fulfill all role memberships

```
[Authorize(Roles = "HRManager,Finance")]  
[HttpGet]  
public IEnumerable<Voucher> Get()  
{  
    var vouchers = ctx.Vouchers.OrderByDescending(v => v.Date).ToList();  
    return vouchers;  
}
```

Windows Authentication

- Windows Authentication can be enabled in
 - Project Properties for testing purposes
 - web.config in IIS or IIS Express

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <security>
      <authentication>
        <windowsAuthentication enabled="true" />
        <anonymousAuthentication enabled="false" />
      </authentication>
    </security>
  </system.webServer>
</configuration>
```

Application Configuration: N/A Platform: N/A

Profile: IIS Express [New... Delete]

Launch: IIS Express

☒ Launch URL:

☐ Use Specific Runtime: Version: 1.0.0-rc1-update1 Platform: .NET Framework Architecture: x86

Environment Variables:

Name	Value
Hosting:Environment	Development

[Add] [Remove]

Web Server Settings

App URL: http://localhost:8086

☐ Enable SSL

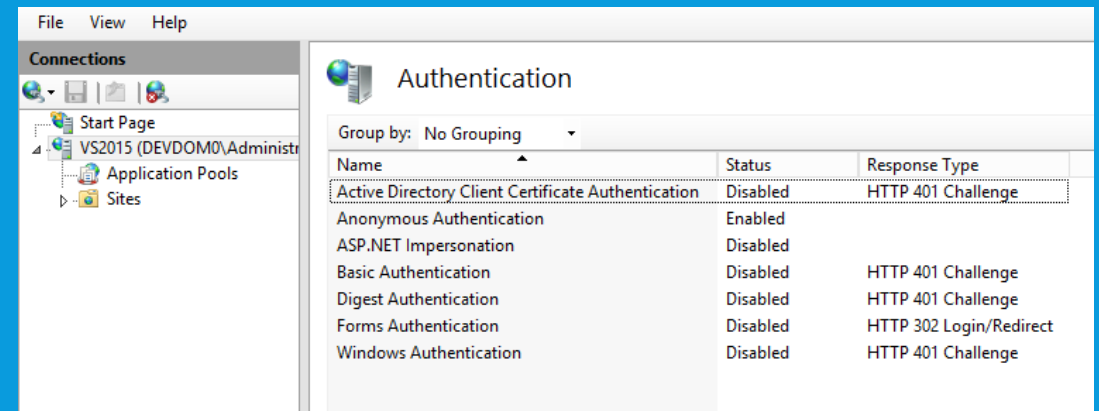
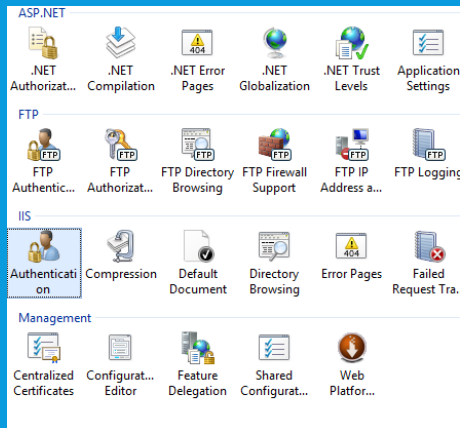
URL: [Copy](#)

☒ Enable Anonymous Authentication

☒ Enable Windows Authentication

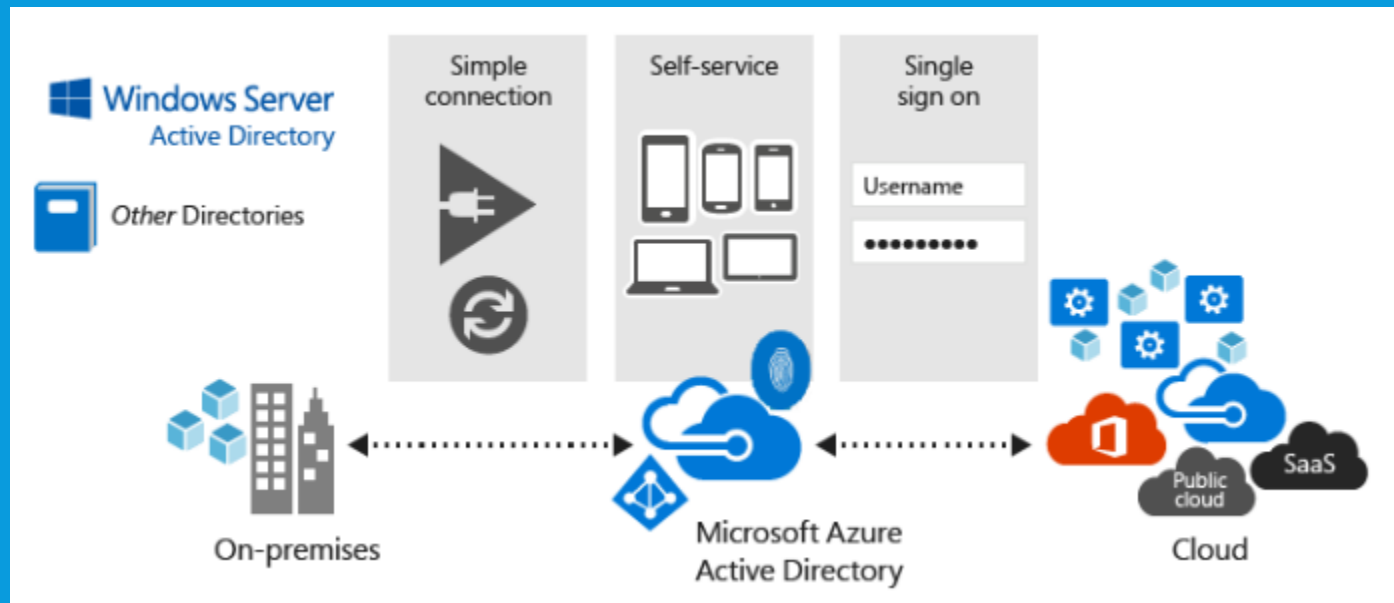
Basic Authentication

- ASP.NET Security will not include Basic Authentication middleware
- If needed it can be configured using IIS
- For testing there is a a github middleware project at <https://github.com/aspnet/Security>



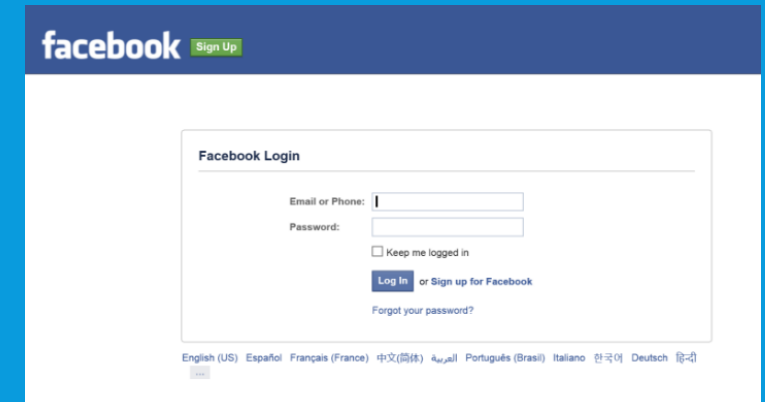
Azure Active Directory

- Azure AD is Microsoft's multi-tenant cloud based directory and identity management service.
- Azure AD provides an affordable, easy to use solution to give employees and business partners single sign-on (SSO) access to thousands of cloud SaaS Applications



OAuth 2.0

- OAuth 2.0 is an open standard for authorization, allowing users to log in to third party websites using their external accounts without exposing their password
- Technically OAuth 2.0 is an authorization protocol
- Supported by
 - Microsoft,
 - Google,
 - Facebook, ...



Use JWT in ASP.NET Core

- Manual Configuration
 - Configure .NET Core for Identity
 - Enable Json Web Tokens (JWT)
- Use Identity Server 4
 - Easy to understand / implement
 - No need to do everything on your own
 - Available for .NET Core
 - Home @ <http://www.identityserver.io/> - Documetation @ <http://docs.identityserver.io/en/release/>