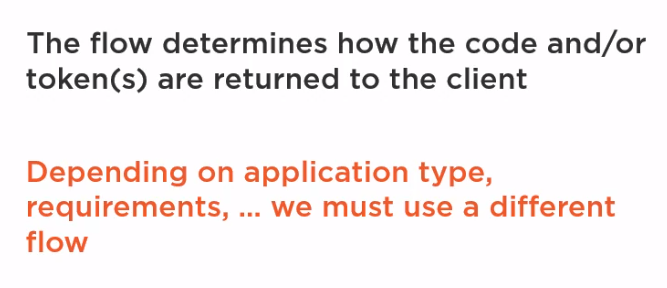
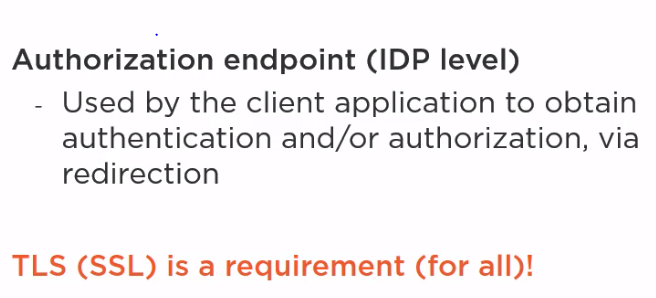
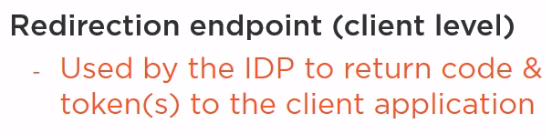
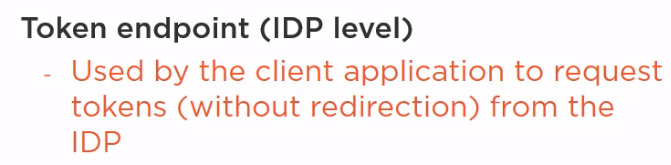
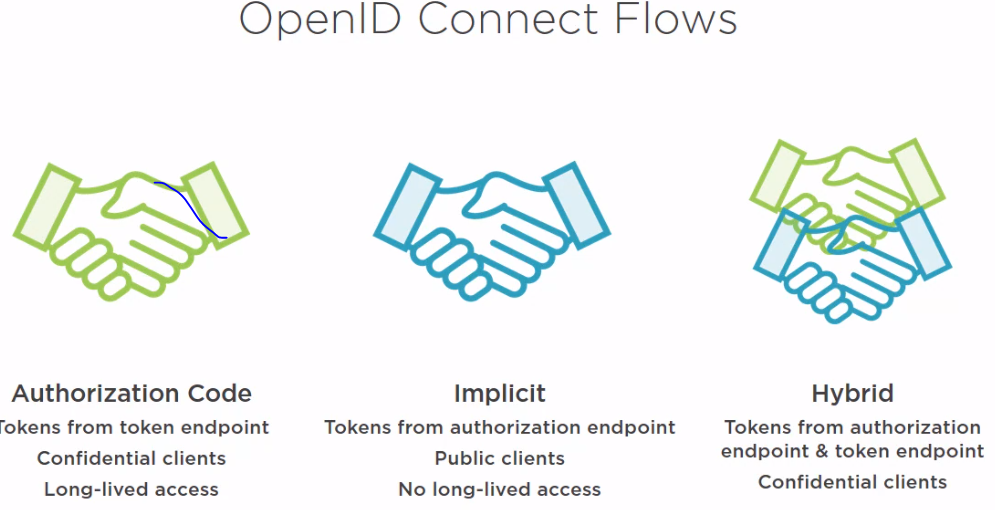
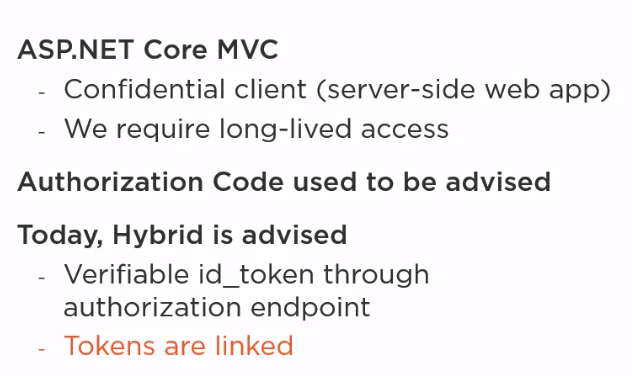
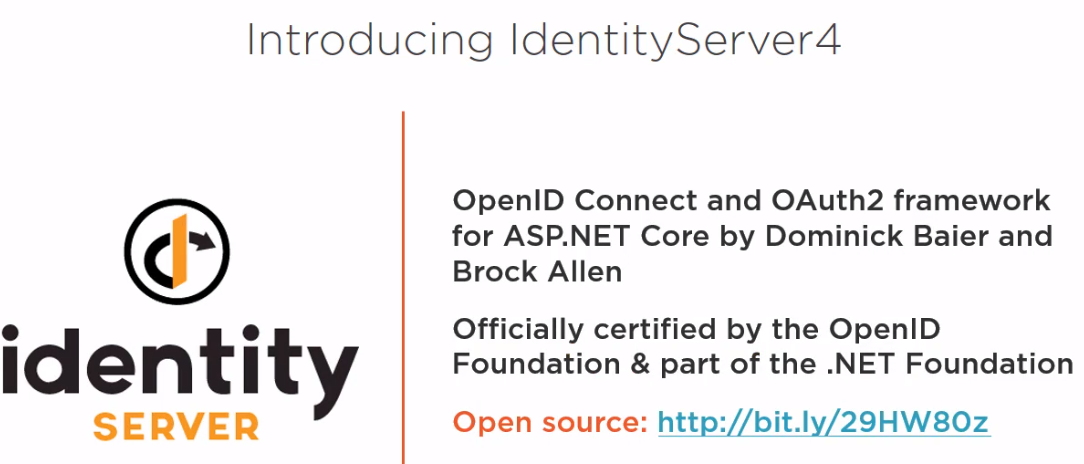
1. 
2. **There are Various End point present inside the openId Connect to handle different type of flows:**  
   Authorization endpoint is used by the client app to obtain authentication for identity tokens and/or authorization for access tokens from the user. This is done by redirecting the client application to the identity provider. So from the authorization endpoint, an authorization code or tokens can be returned to the client application. Important to know is that in OIDC, TLS, or as it's wrongly named SSL, is a requirement. In other words, traffic should always be encrypted.

  
Redirection Endpoint  
We're talking about redirection here. The client redirects to the authorization endpoint at level of the identity provider and the identity provider redirects back to the client. The URI or the IDP redirects the client back to what's called the redirection endpoint  
  
  
Token EndPoint -   
From this endpoint, which lives at level of the identity provider, client applications can programmatically request tokens and that's typically done via an HTTP post without redirection.  
  


1. **Different Type of Flows Present Inside OpenId Connect and OAuth2.0**The first one is the OpenID Connect authorization code flow. It returns an authorization code from the authorization endpoint and tokens from the token endpoint. This authorization code can be seen as a short-lived single use credential used to verify that the user who logged in at level of the identity provider is the same one who started the flow at level of the web app. The authorization code flow is suitable for confidential clients and it allows long-lived access. The implicit flow returns all tokens from the authorization endpoint and there is no authorization code. The token endpoint is induced. It's suitable for public clients. There is no client authentication as public clients can safely store client secrets anyway. Therefore, there is also no long-lived access through the fresh tokens allowed. The implicit flow may be used by confidential clients as well. Lastly, the hybrid flow. This one returns some tokens from the authorization endpoint and others from the token endpoint. It's just a mix of the authorization code and implicit grant or flow.  
   
2. Refresh tokens aren't allowed with an implicit flow, so we won't choose that. And that leaves us with two flows, the authorization code and the hybrid flow. Before OpenID Connect was conceived, the authorization code flow was the flow of choice. For that flow, as we learned, all tokens are returned from the token endpoint, but the hybrid flow has an advantage over that one. It allows us to get an identity token from the authorization endpoint first and we can then verify that before continuing with additional roundtrips to get an access token. Next to that, OpenID Connect links the identity token to an optional access token and these are reasons to choose OpenID Connect over regular OAuth2.  
   
3. 
4. IdentityResources map to scopes that give access to identity related information. Later, we'll also encounter API resources. Those map to scopes that give access to API resources.
5. <http://localhost:53533/.well-known/openid-configuration>-  
   It's this discovery document that's read by other pieces of middleware when we they want to find out where, for example, those endpoints can be found.