Using the signing certificate in the Identity microservice

Let's update our Identity microservice to use the signing certificate we created in the previous lesson.

In Play.Identity

1. Update IdentitySettings:

```
public class IdentitySettings
{
    …
    public string PathBase { get; init; }
    public string CertificateCerFilePath { get; init; }
    public string CertificateKeyFilePath { get; init; }
}
```

2. Update Startup:

```
public class Startup
{
    private const string AllowedOriginSetting = "AllowedOrigin";
    private readonly IHostEnvironment environment;

    public Startup(IConfiguration configuration, IHostEnvironment environment)
    {
        Configuration = configuration;
        this.environment = environment;
    }
    …
    public void ConfigureServices(IServiceCollection services)
    {
        …
        services.AddMassTransitWithMessageBroker(Configuration, retryConfigurator =>
        {
            …
        });

        AddIdentityServer(services);

        services.AddLocalApiAuthentication();
        …
    }
    …
    private void AddIdentityServer(IServiceCollection services)
    {
```

```
    var identityServerSettings =
Configuration.GetSection(nameof(IdentityServerSettings)).Get<IdentityServerSettings>();

    var builder = services.AddIdentityServer(options =>
    {
      options.Events.RaiseSuccessEvents = true;
      options.Events.RaiseFailureEvents = true;
      options.Events.RaiseErrorEvents = true;
    })
    .AddAspNetIdentity<ApplicationUser>()
    .AddInMemoryApiScopes(identityServerSettings.ApiScopes)
    .AddInMemoryApiResources(identityServerSettings.ApiResources)
    .AddInMemoryClients(identityServerSettings.Clients)
    .AddInMemoryIdentityResources(identityServerSettings.IdentityResources);


    if (environment.IsDevelopment())
    {
      builder.AddDeveloperSigningCredential();
    }
    else
    {
      var identitySettings =
Configuration.GetSection(nameof(IdentitySettings)).Get<IdentitySettings>();
      var cert = X509Certificate2.CreateFromPemFile(
        identitySettings.CertificateCerFilePath,
        identitySettings.CertificateKeyFilePath);
      builder.AddSigningCredential(cert);
    }
  }
}
```

3. Bump container image version in README and in identity.yaml

4. Build and push the image

5. Apply identity.yaml

In Postman

6. Do a GET on the jwks_uri

7. Notice it's a different, bigger value than before

8. Copy the value to a temporal location

9. Destroy the Identity pod
10. Do a GET on the jwks_uri again
11. Notice the value did not change
12. Get a new access token
13. Decode the token
14. Notice kid matches the one in the jwks_uri
15. Do a GET on /users. It should work.
16. Commit and push

In the next module you will deploy your other microservices to the K8s cluster and see the full system working in production.