Tell don't ask - thanks to SignalR Services

Mark Allibone

Expert Software Engineer

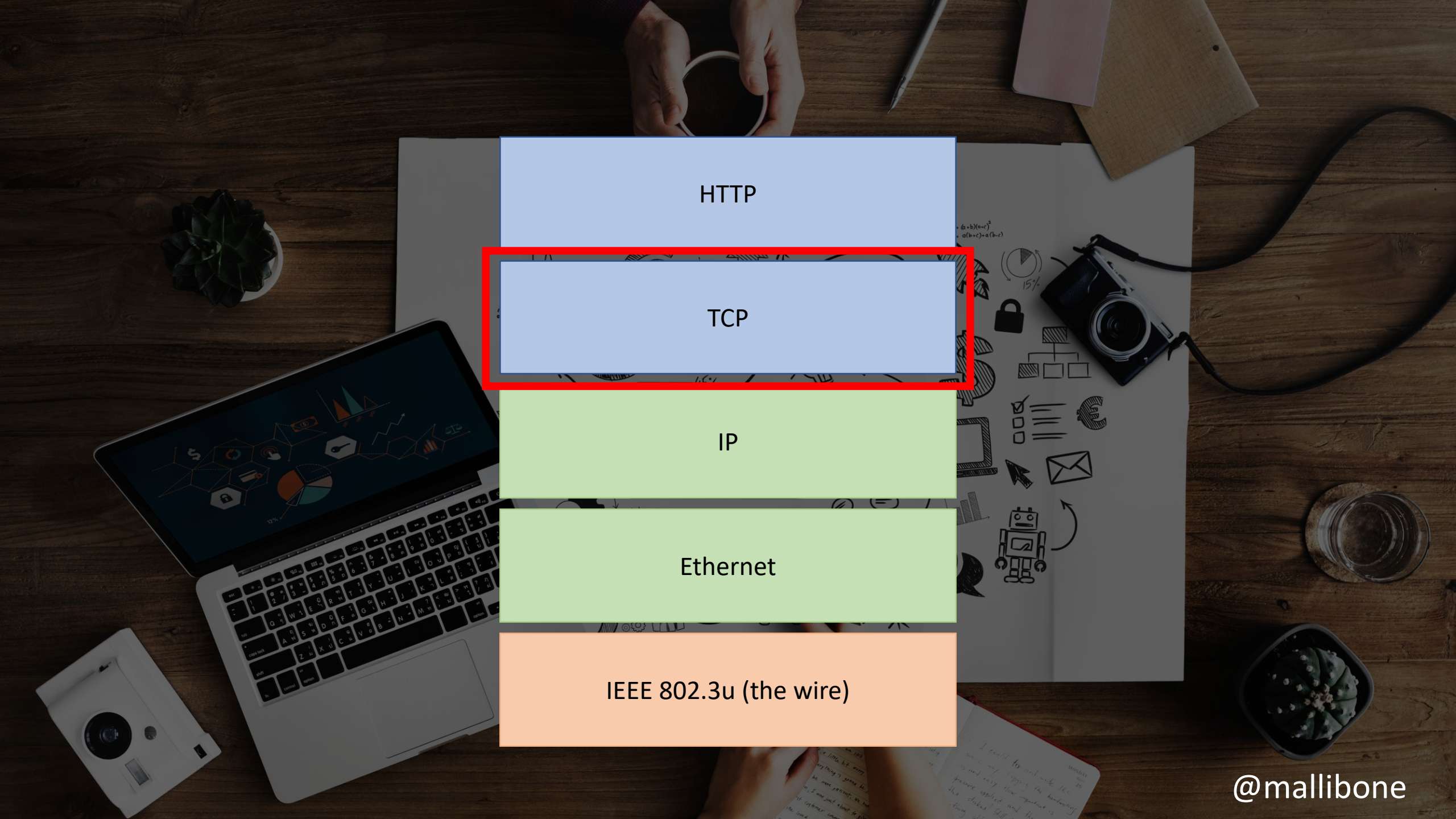
Rey Technology

@mallibone

Photo by **Daria Shevtsova** from **Pexels**

@mallibone


HTTP
@mallibone


HTTP
TCP
IP
Ethernet
IEEE 802.3u (the wire)
@mallibone

TCP Sockets

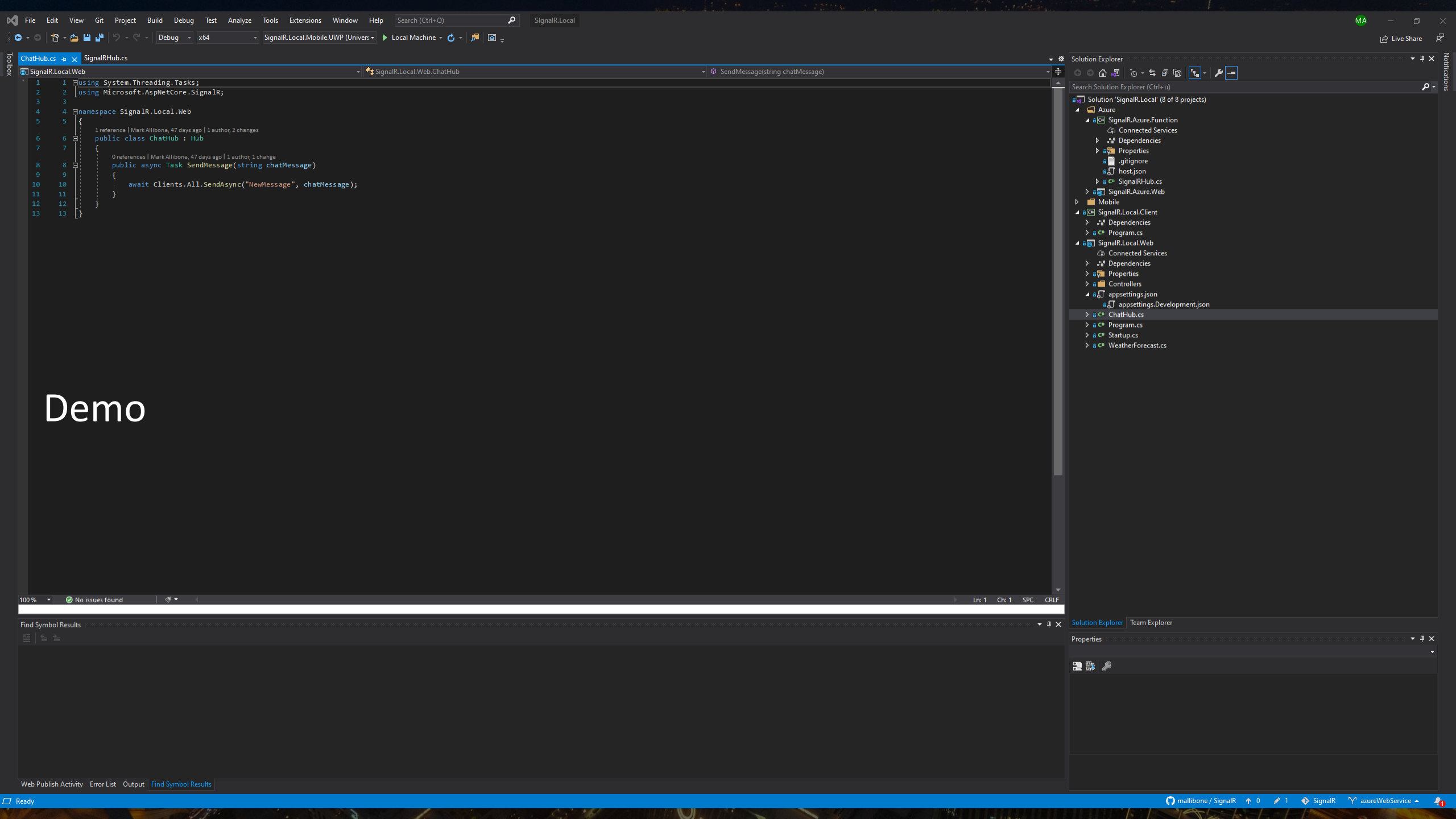
@mallibone

TCP Sockets → Web Sockets

@mallibone

TCP Sockets → Web Sockets → SignalR

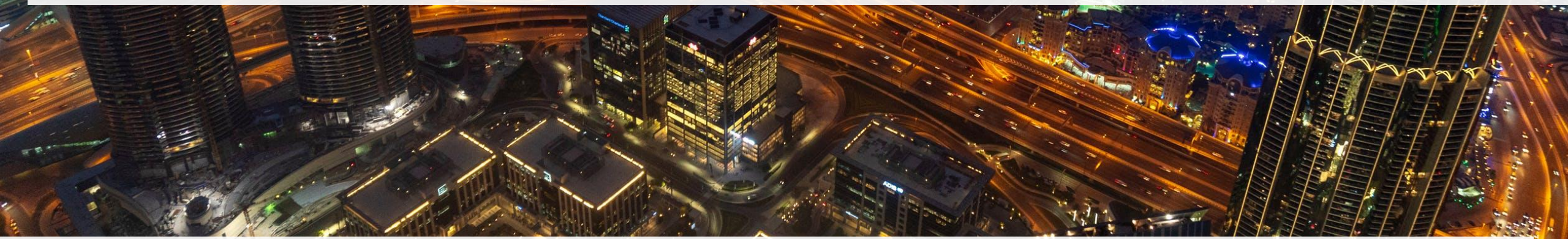
@mallibone


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
SignalR.Local
Debug
x64
SignalR.Local.Mobile.UWP (Univers
Local Machine
Live Share
ChatHub.cs
SignalRHub.cs
SignalR.Local.Web
SignalR.Local.Web.ChatHub
SendMessage(string chatMessage)
using System.Threading.Tasks;
using Microsoft.AspNetCore.SignalR;
namespace SignalR.Local.Web
{
1 reference | Mark Allibone, 47 days ago | 1 author, 2 changes
public class ChatHub : Hub
{
0 references | Mark Allibone, 47 days ago | 1 author, 1 change
public async Task SendMessage(string chatMessage)
{
await Clients.All.SendAsync("NewMessage", chatMessage);
}
}
}
Demo
100 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'SignalR.Local' (8 of 8 projects)
Azure
SignalR.Azure.Function
Connected Services
Dependencies
Properties
.gitignore
host.json
SignalRHub.cs
SignalR.Azure.Web
Mobile
SignalR.Local.Client
Dependencies
Program.cs
SignalR.Local.Web
Connected Services
Dependencies
Properties
Controllers
appsettings.json
appsettings.Development.json
ChatHub.cs
Program.cs
Startup.cs
WeatherForecast.cs
Solution Explorer
Team Explorer
Properties
Find Symbol Results
Web Publish Activity
Error List
Output
Find Symbol Results
Ready
mallibone / SignalR
0
1
SignalR
azureWebService

Demo Recap

- Integrated in ASP.NET Core
- Client via NuGet (.Net Standard)
- Local Development can be tricky
- SignalR can fall back to other methods if WebSockets are disabled

”How does it Scale?”



It depends™

Option 1: Vertical Scaling

Servericon made by prettycons from www.flaticon.com

@mallibone

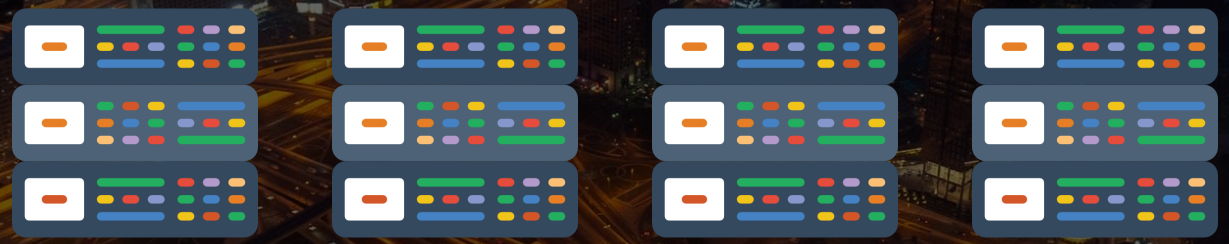
Option 1: Vertical Scaling

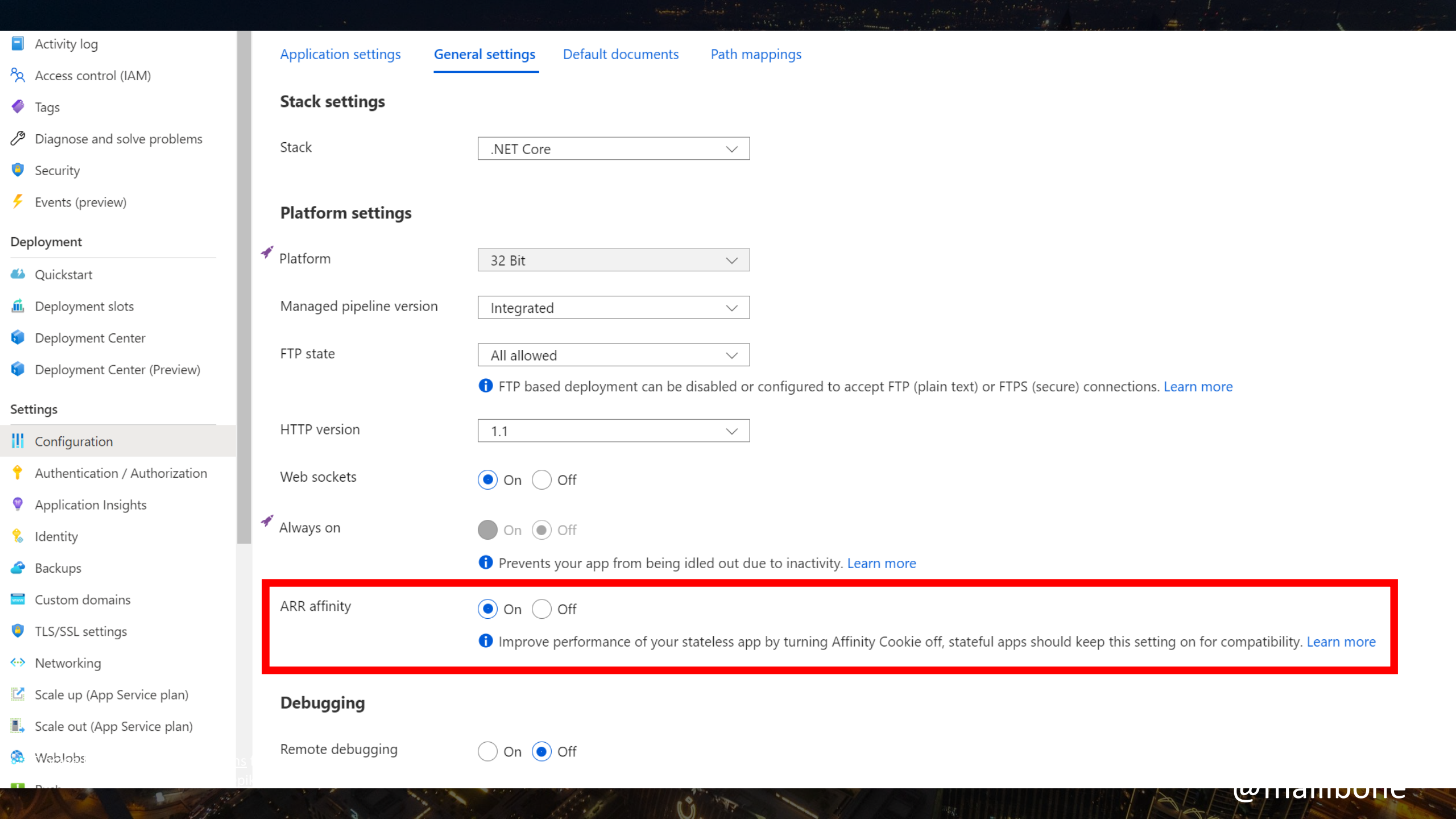
https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#app-service-limits

@mallibone

Option 2: Horizontal Scaling

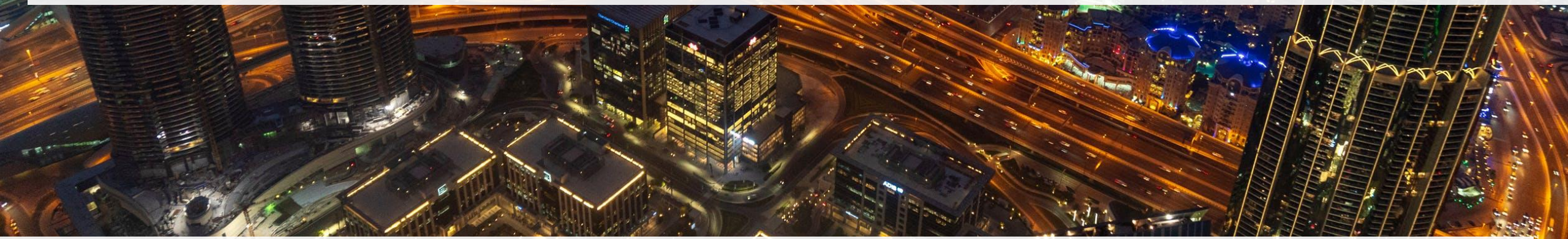
Option 2: Horizontal Scaling



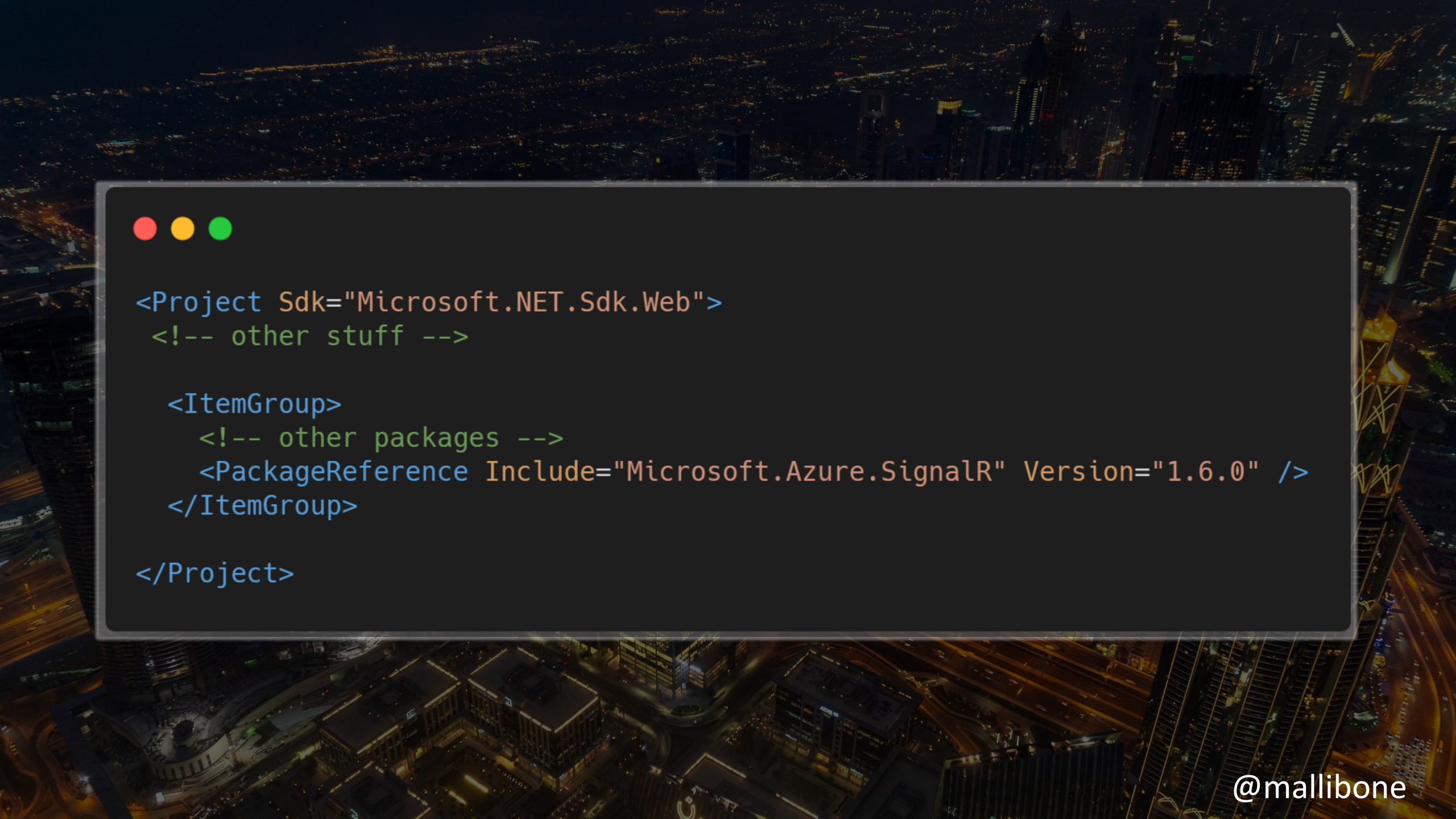

Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Security
Events (preview)
Deployment
Quickstart
Deployment slots
Deployment Center
Deployment Center (Preview)
Settings
Configuration
Authentication / Authorization
Application Insights
Identity
Backups
Custom domains
TLS/SSL settings
Networking
Scale up (App Service plan)
Scale out (App Service plan)
Webjobs
Application settings
General settings
Default documents
Path mappings
Stack settings
Stack
.NET Core
Platform settings
Platform
32 Bit
Managed pipeline version
Integrated
FTP state
All allowed
FTP based deployment can be disabled or configured to accept FTP (plain text) or FTPS (secure) connections. Learn more
HTTP version
1.1
Web sockets
On
Off
Always on
On
Off
Prevents your app from being idled out due to inactivity. Learn more
ARR affinity
On
Off
Improve performance of your stateless app by turning Affinity Cookie off, stateful apps should keep this setting on for compatibility. Learn more
Debugging
Remote debugging
On
Off

@mallibone

«There is an Azure Service for that.™»

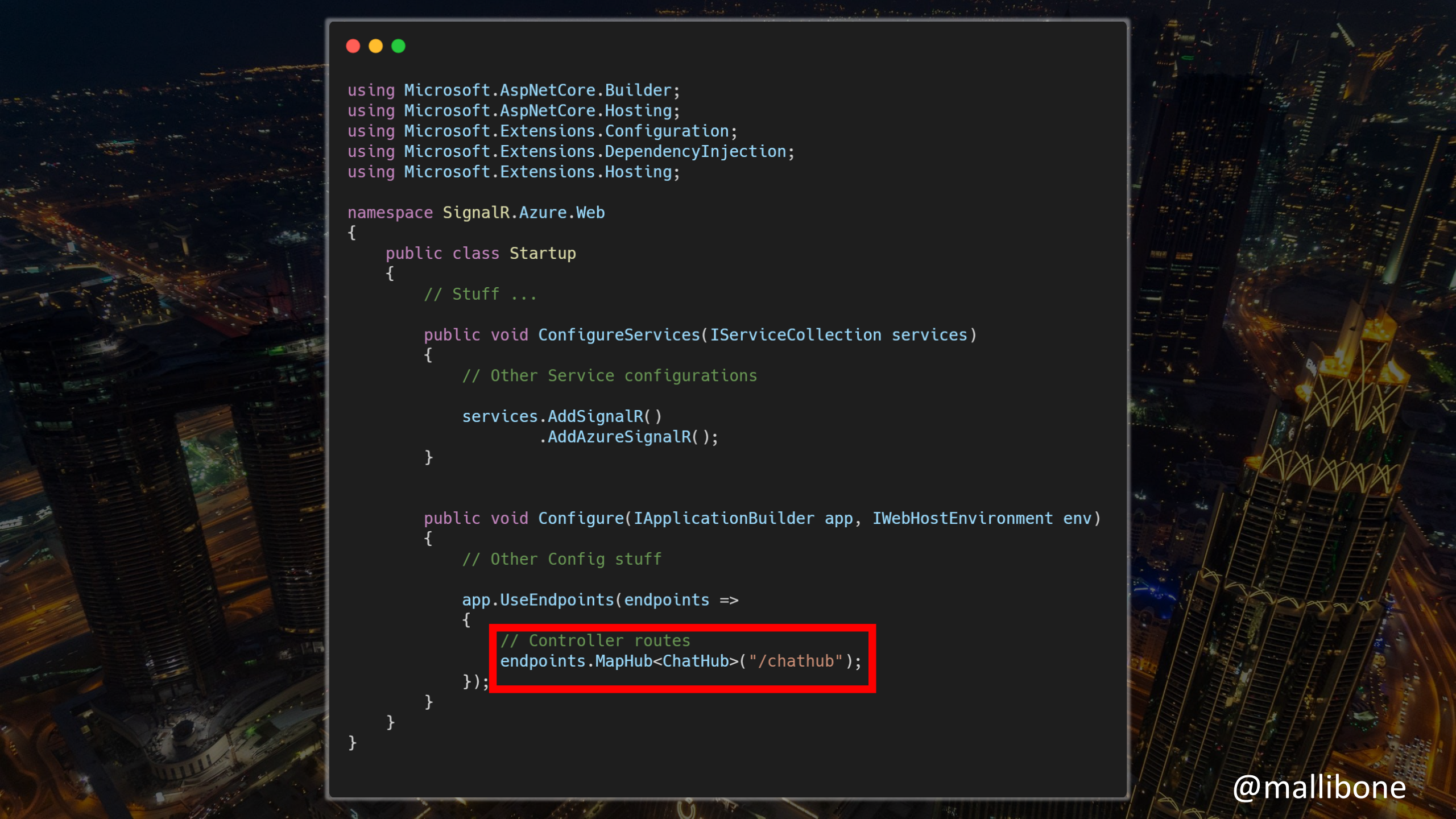


@mallibone



```
<Project Sdk="Microsoft.NET.Sdk.Web">
 <!-- other stuff -->

  <ItemGroup>
    <!-- other packages -->
    <PackageReference Include="Microsoft.Azure.SignalR" Version="1.6.0" />
  </ItemGroup>

</Project>
```

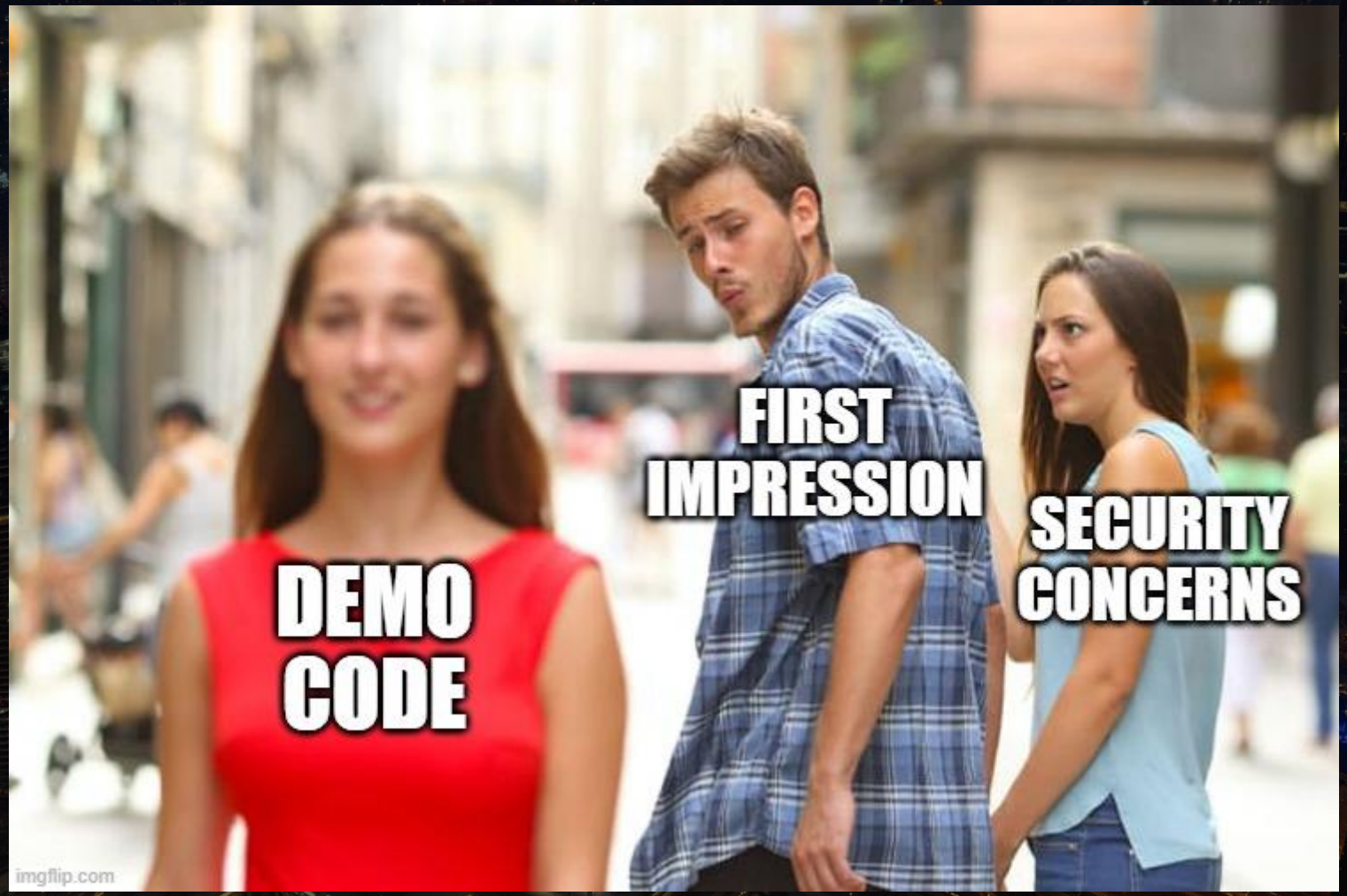
```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace SignalR.Azure.Web
{
    public class Startup
    {
        // Stuff ...

        public void ConfigureServices(IServiceCollection services)
        {
            // Other Service configurations

            services.AddSignalR()
                    .AddAzureSignalR();
        }


        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            // Other Config stuff

            app.UseEndpoints(endpoints =>
            {
                // Controller routes
                endpoints.MapHub<ChatHub>("/chathub");
            });
        }
    }
}
```



```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace SignalR.Azure.Web
{
    public class Startup
    {
        // Stuff ...

        public void ConfigureServices(IServiceCollection services)
        {
            // Other Service configurations

            services.AddSignalR()
                    .AddAzureSignalR();
        }


        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            // Other Config stuff

            app.UseEndpoints(endpoints =>
            {
                // Controller routes
                endpoints.MapHub<ChatHub>("/chathub");
            });
        }
    }
}
```

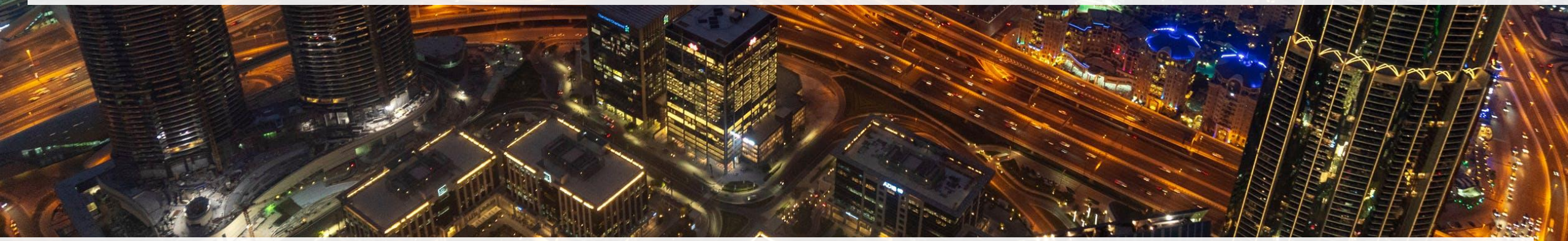
That is all...

@mallibone



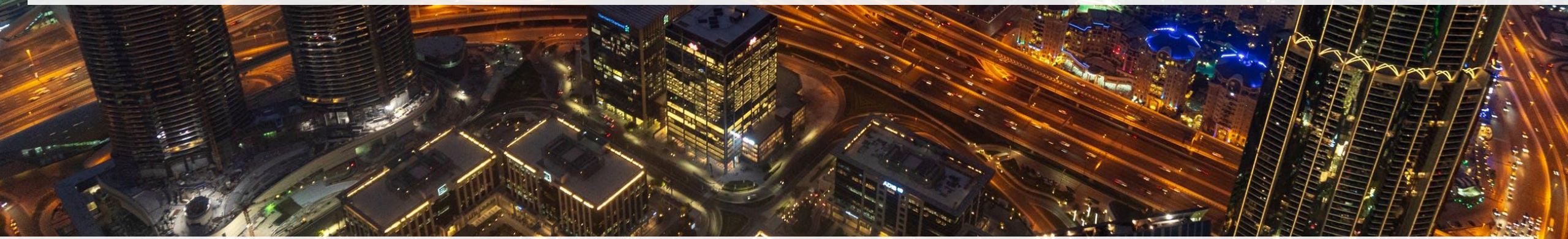
@mallibone

Cross-origin resource sharing (CORS)



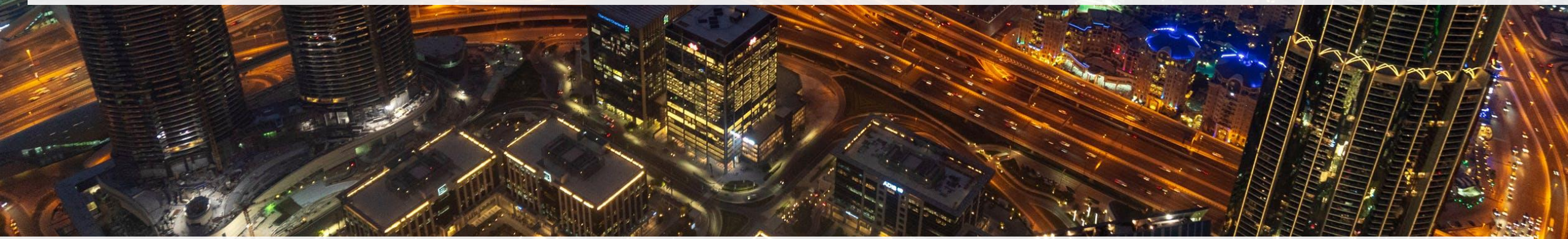

public class Startup
{
// Startup Configurations ...
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
// Middleware configurations ...
app.UseCors(builder =>
{
builder.WithOrigins("https://gnabber.com")
.AllowAnyHeader()
.WithMethods("GET", "POST")
.AllowCredentials();
});
app.UseRouting();
app.UseEndpoints(endpoints =>
{
endpoints.MapHub<ChatHub>("/chathub");
});
// ... even more middleware configuraitons ...
}
}
@mallibone

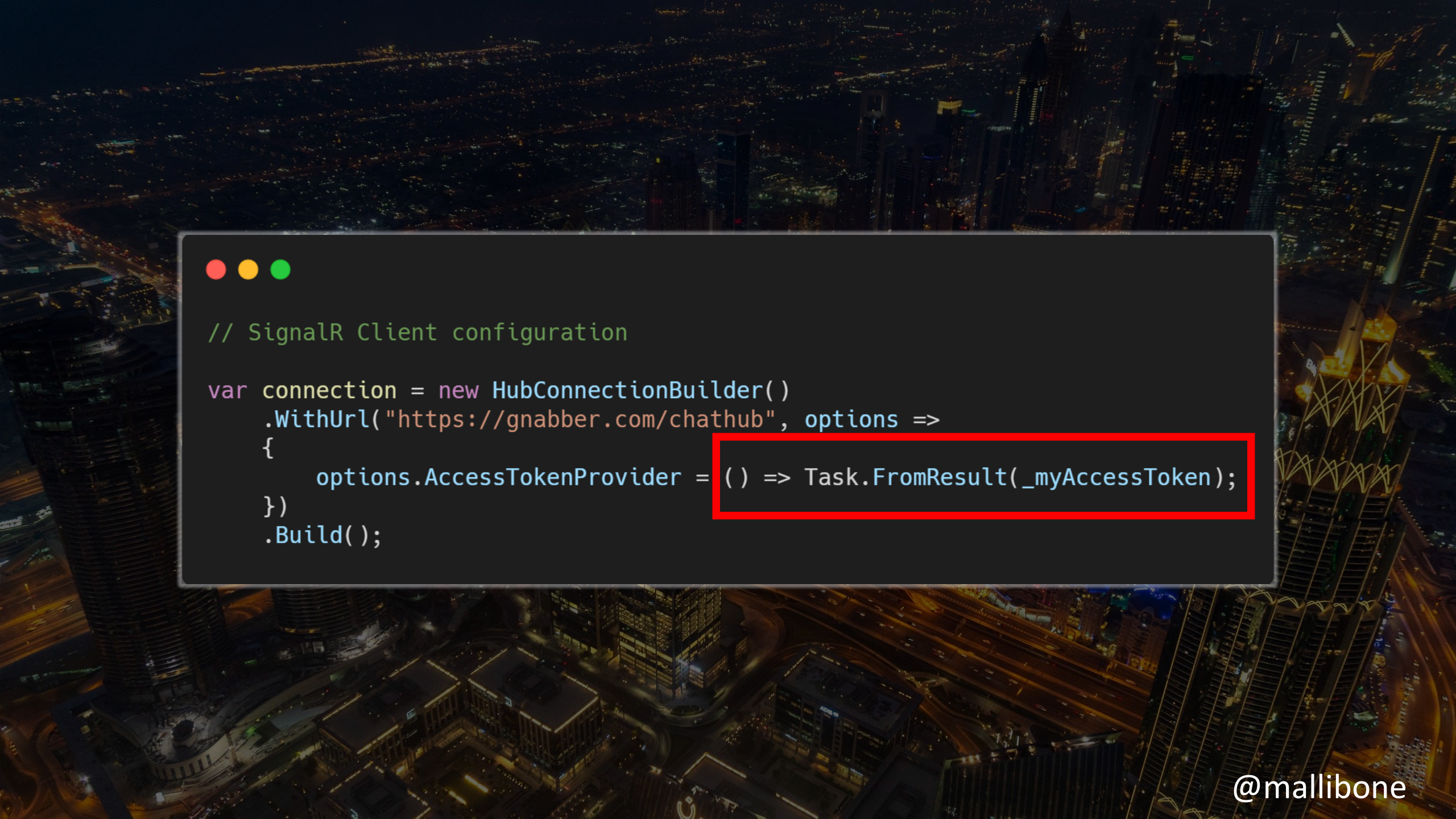
Authentication and Authorization



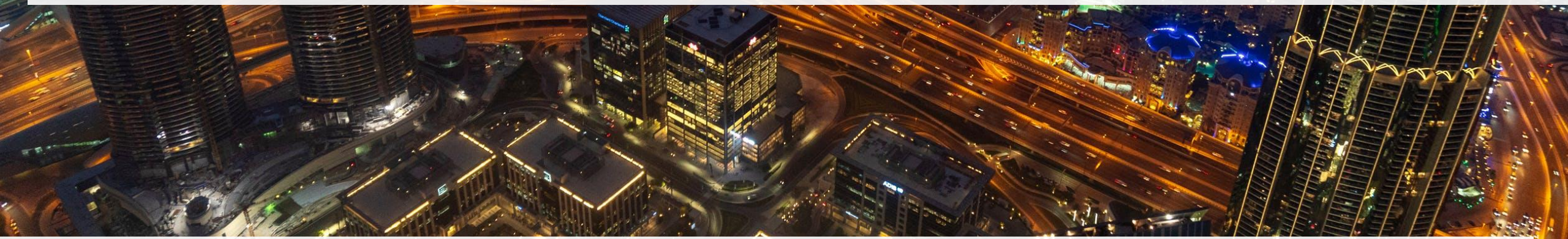
```
public class Startup
{
    // Startup Configurations ...

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        // Middleware config ...

        app.UseAuthentication();
        app.UseAuthorization();

        // ... more middleware Config
    }
}
```

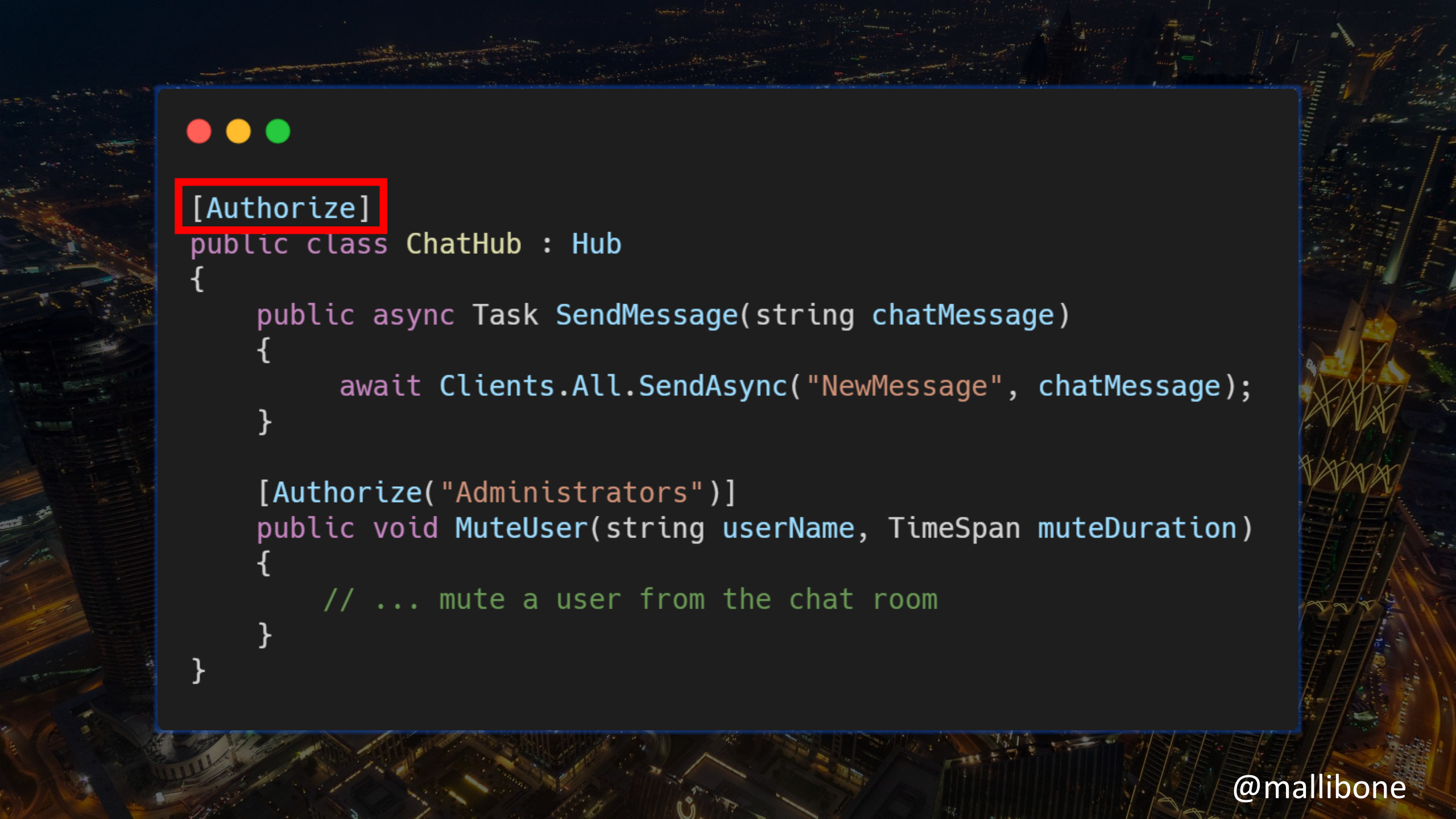
Token based Authentication

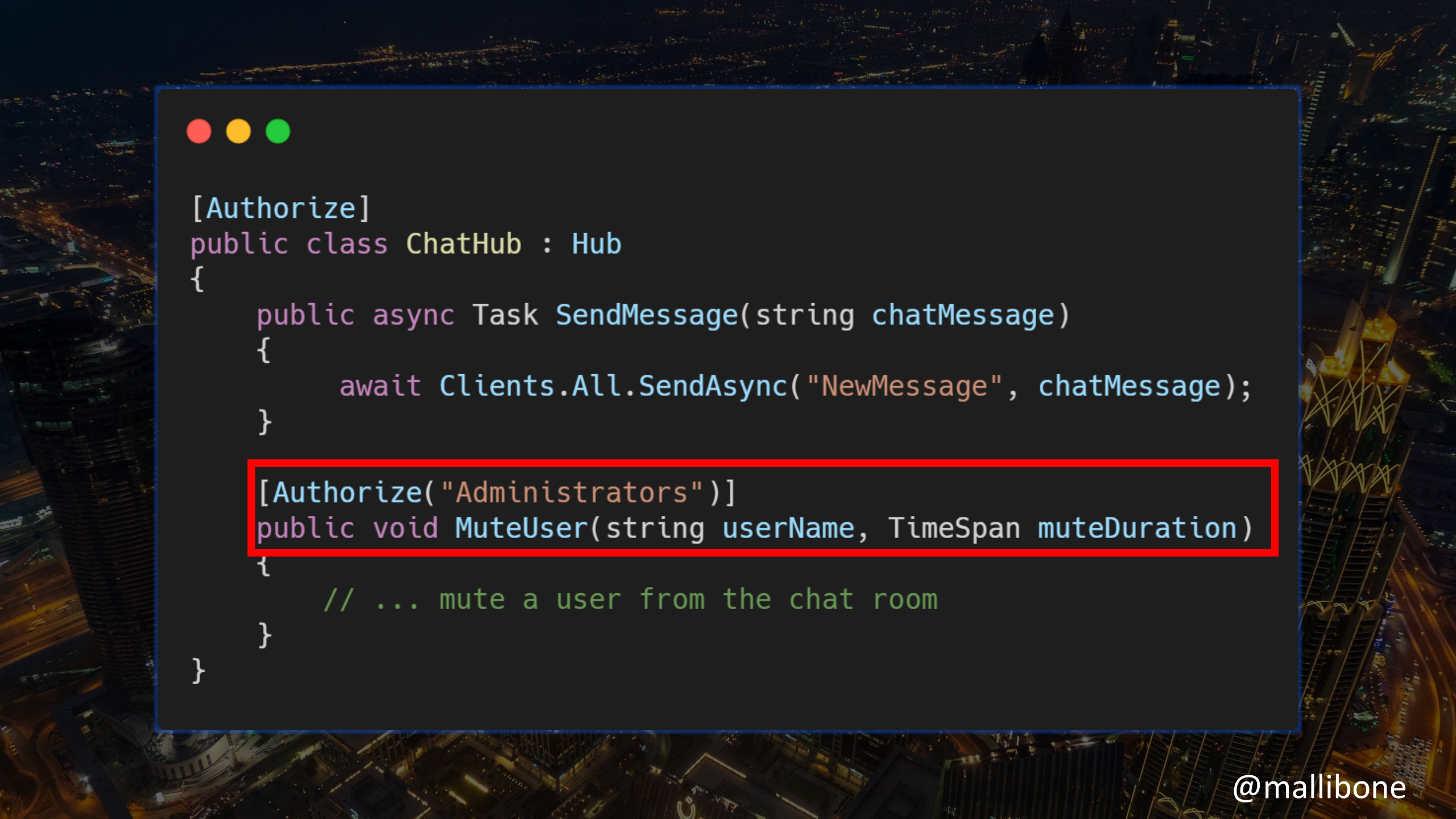



// SignalR Client configuration
var connection = new HubConnectionBuilder()
.WithUrl("https://gnabber.com/chathub", options =>
{
options.AccessTokenProvider = () => Task.FromResult(_myAccessToken);
})
.Build();
@mallibone

Authentication and Authorization

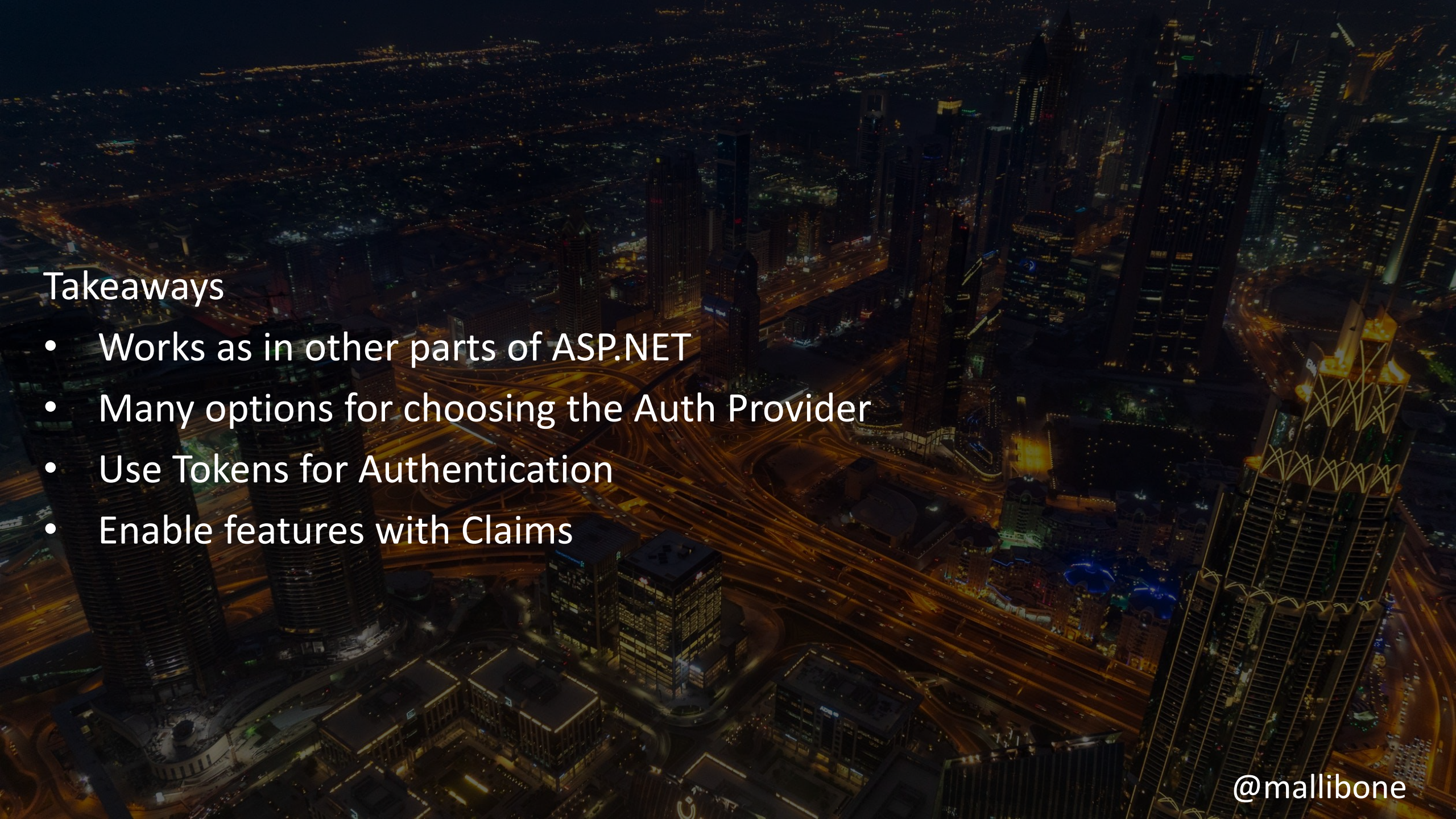



[Authorize]
public class ChatHub : Hub
{
public async Task SendMessage(string chatMessage)
{
await Clients.All.SendAsync("NewMessage", chatMessage);
}
[Authorize("Administrators")]
public void MuteUser(string userName, TimeSpan muteDuration)
{
// ... mute a user from the chat room
}
}
@mallibone



```
[Authorize]
public class ChatHub : Hub
{
    public async Task SendMessage(string chatMessage)
    {
        await Clients.All.SendAsync("NewMessage", chatMessage);
    }

    [Authorize("Administrators")]
    public void MuteUser(string userName, TimeSpan muteDuration)
    {
        // ... mute a user from the chat room
    }
}
```

@mallibone



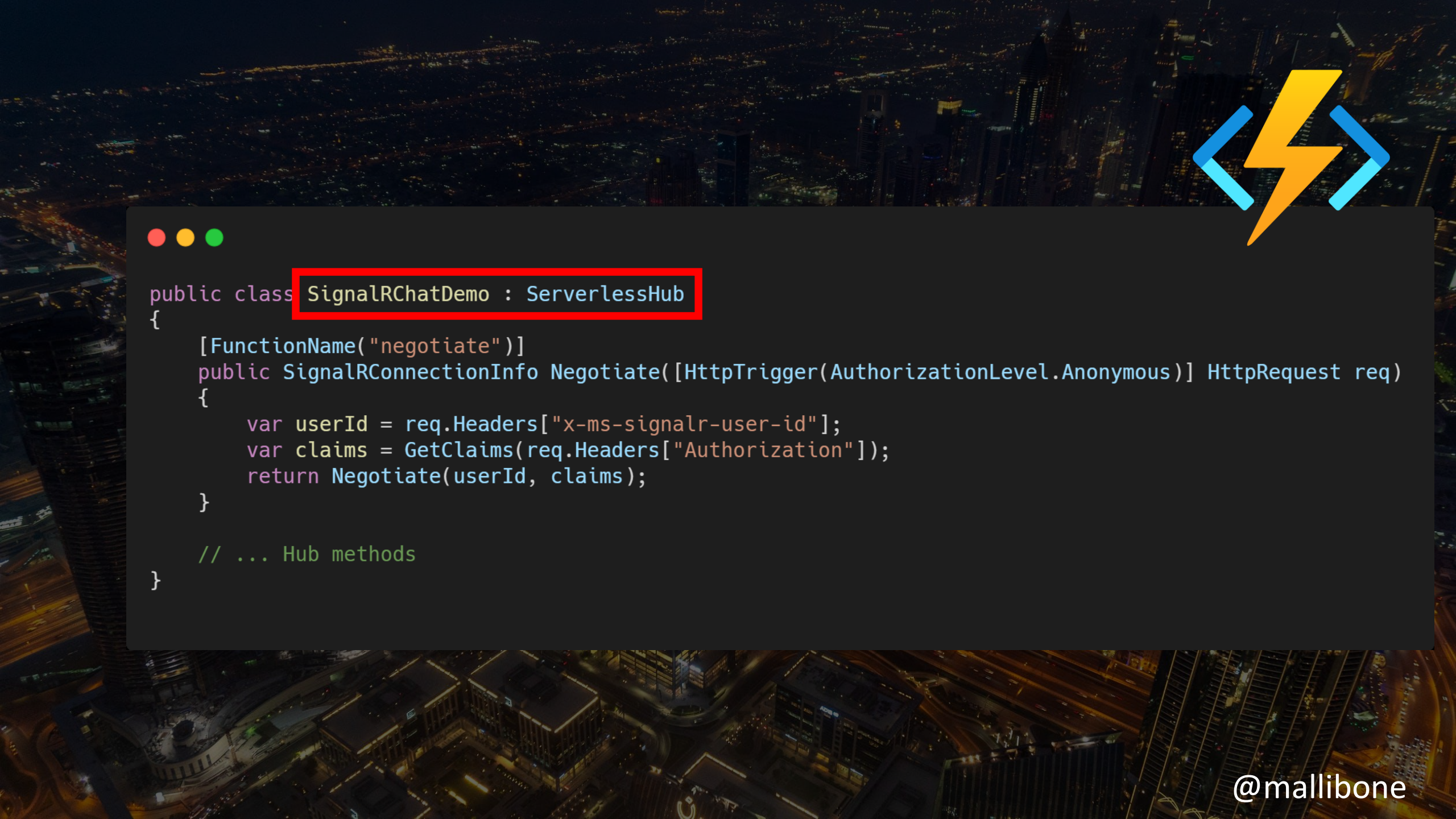
Takeaways

- Works as in other parts of ASP.NET
- Many options for choosing the Auth Provider
- Use Tokens for Authentication
- Enable features with Claims

@mallibone

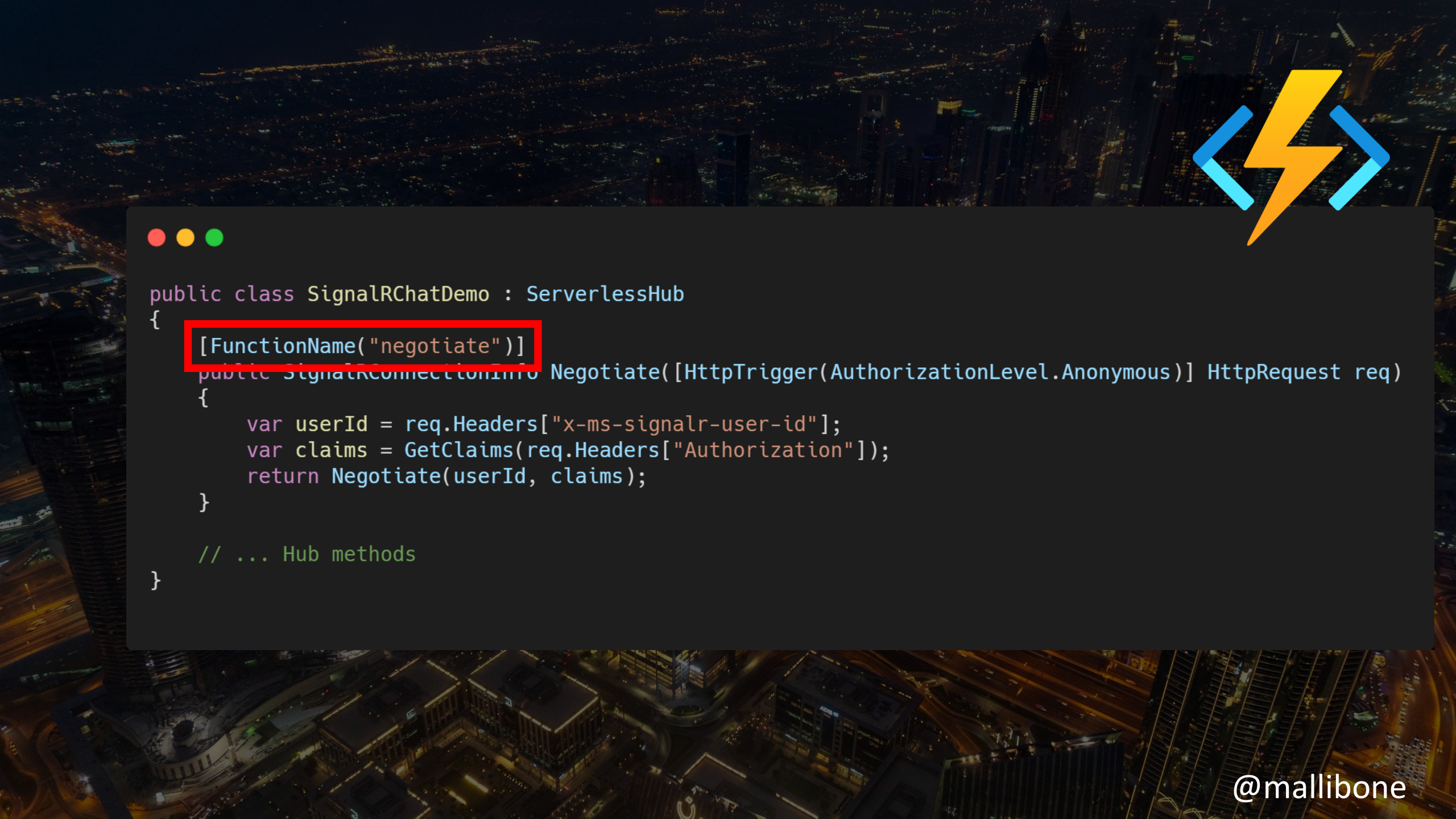
@mallibone


1. Connect to SignalR
@mallibone

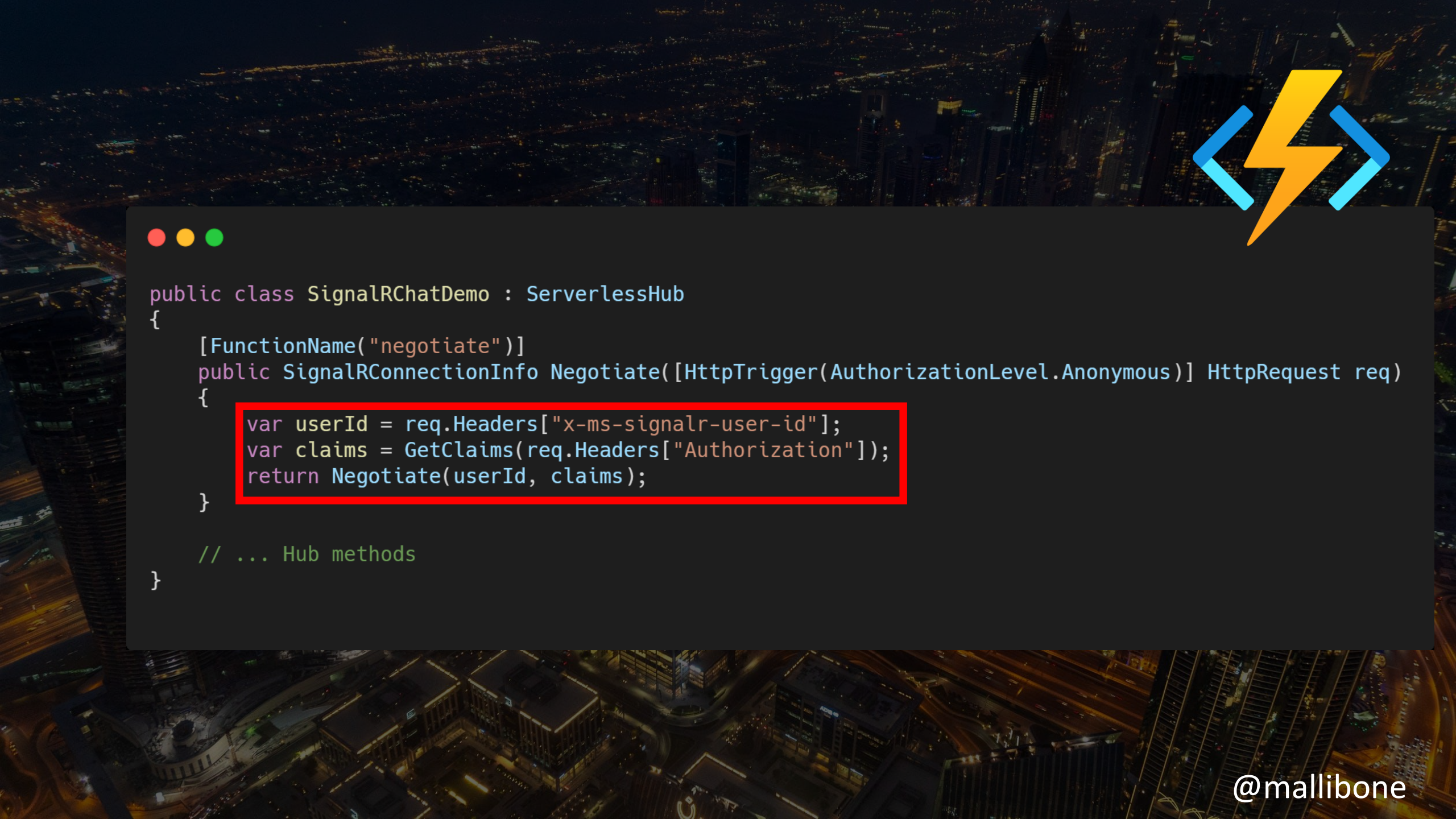


```
public class SignalRChatDemo : ServerlessHub
{
    [FunctionName("negotiate")]
    public SignalRConnectionInfo Negotiate([HttpTrigger(AuthorizationLevel.Anonymous)] HttpRequest req)
    {
        var userId = req.Headers["x-ms-signalr-user-id"];
        var claims = GetClaims(req.Headers["Authorization"]);
        return Negotiate(userId, claims);
    }

    // ... Hub methods
}
```

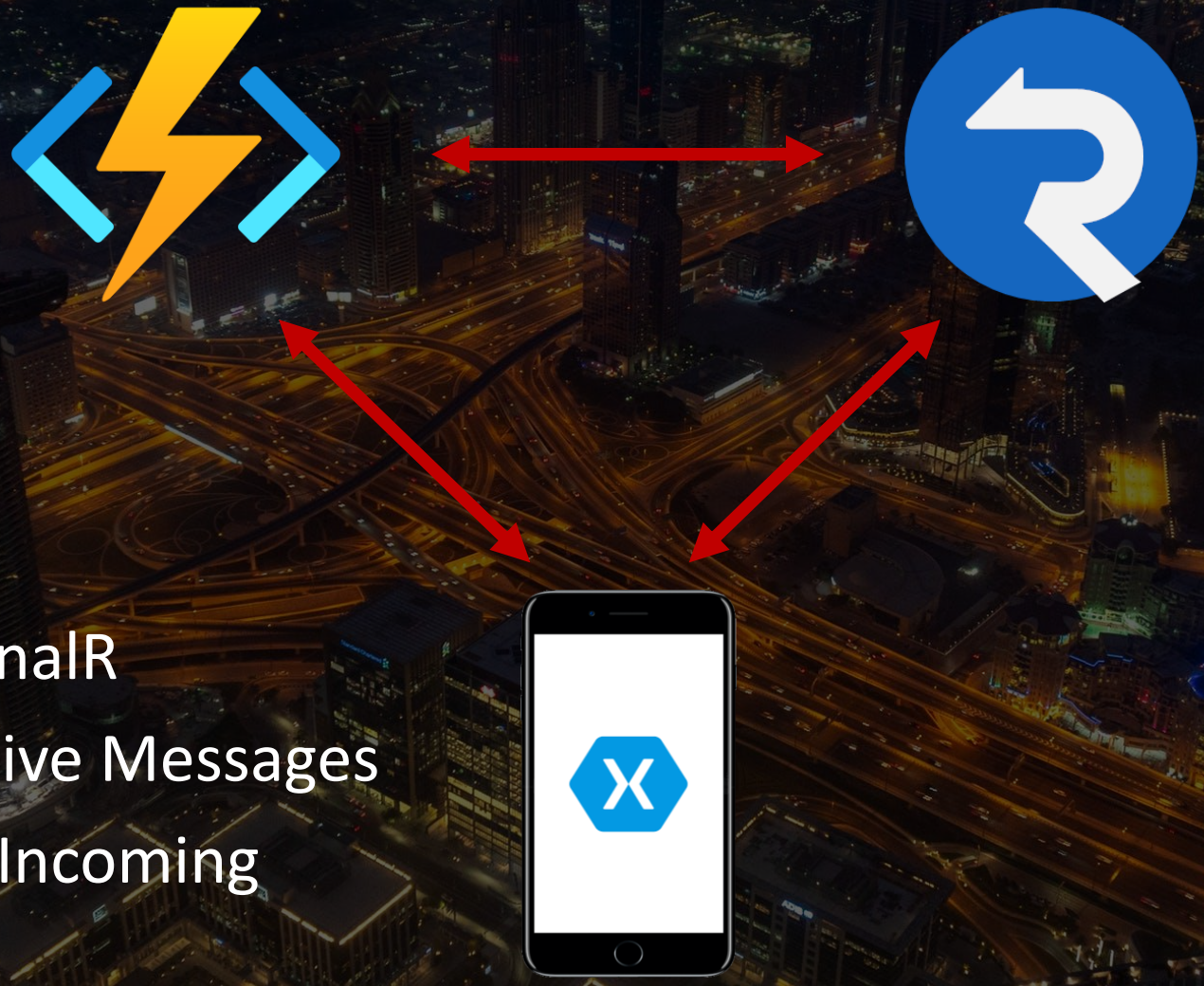
@mallibone



```
public class SignalRChatDemo : ServerlessHub
{
    [FunctionName("negotiate")]
    public SignalRConnectionInfo Negotiate([HttpTrigger(AuthorizationLevel.Anonymous)] HttpRequest req)
    {
        var userId = req.Headers["x-ms-signalr-user-id"];
        var claims = GetClaims(req.Headers["Authorization"]);
        return Negotiate(userId, claims);
    }

    // ... Hub methods
}
```

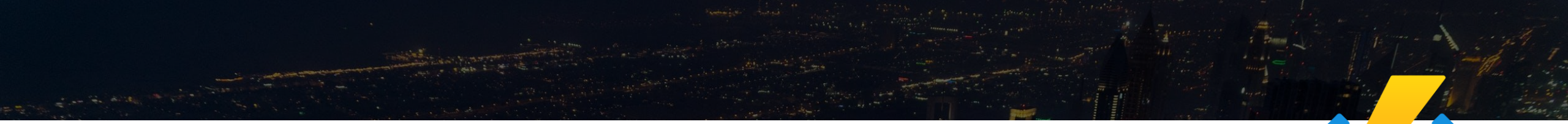


```
public class SignalRChatDemo : ServerlessHub
{
    [FunctionName("negotiate")]
    public SignalRConnectionInfo Negotiate([HttpTrigger(AuthorizationLevel.Anonymous)] HttpRequest req)
    {
        var userId = req.Headers["x-ms-signalr-user-id"];
        var claims = GetClaims(req.Headers["Authorization"]);
        return Negotiate(userId, claims);
    }

    // ... Hub methods
}
```



1. Connect to SignalR
2. Send and Receive Messages
3. Process/Relay Incoming messages

@mallibone



StayConnectedWithSignalRAzureFunction | App keys

Function App

Search (Ctrl+/)

Refresh

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Security
- Events (preview)

Functions

- Functions
- App keys
- App files
- Proxies

Deployment

- Deployment slots
- Deployment Center

Settings

- Configuration
- Authentication / Authorization
- Application Insights

Host keys (all functions)

Use Host keys with your clients to access all your HTTP functions in the app. _master key grants admin access to Functions Runtime APIs.

New host key | Show values

Filter host keys

Name	Value	
_master	Hidden value. Click to show value	Renew key value
default	Hidden value. Click to show value	Renew key value

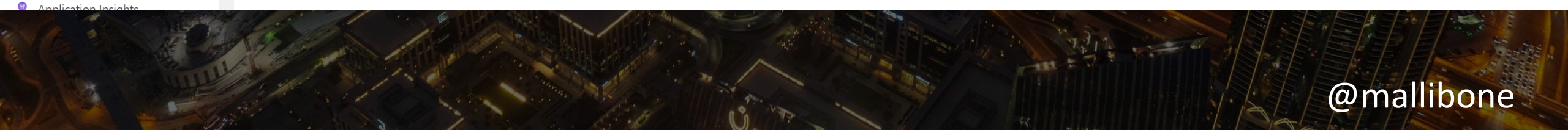
System keys

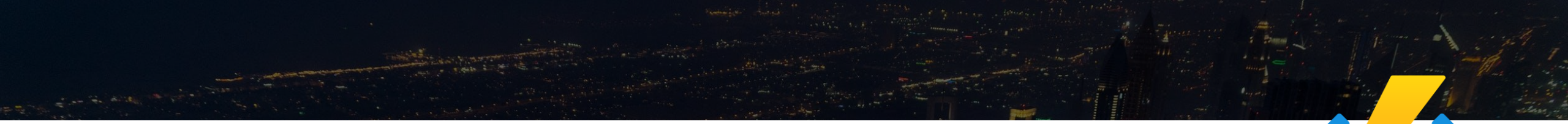
System keys are automatically managed by the Function runtime. System Keys provide granular access to functions runtime features.

Show values

Filter system keys

Name	Value	
signalr_extension	Hidden value. Click to show value	Renew key value





StayConnectedWithSignalRAzureFunction | App keys

Function App

Search (Ctrl+/)

Refresh

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Security
- Events (preview)

Functions

- Functions
- App keys
- App files
- Proxies

Deployment

- Deployment slots
- Deployment Center

Settings

- Configuration
- Authentication / Authorization
- Application Insights

Host keys (all functions)

Use Host keys with your clients to access all your HTTP functions in the app. _master key grants admin access to Functions Runtime APIs.

New host key | Show values

Filter host keys

Name	Value	
_master	Hidden value. Click to show value	Renew key value
default	Hidden value. Click to show value	Renew key value

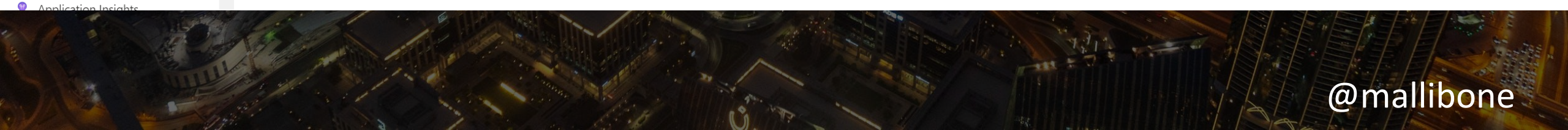
System keys

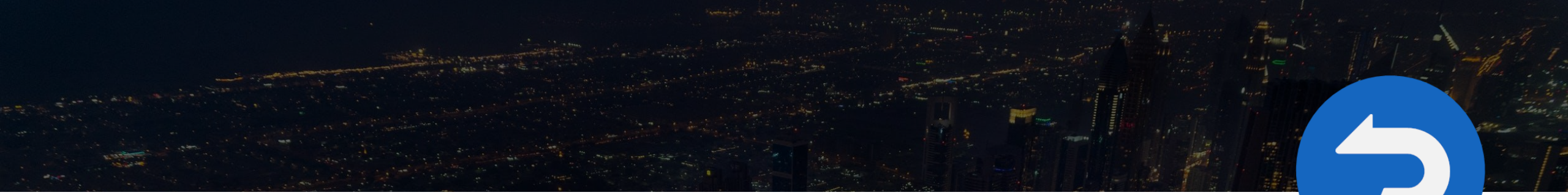
System keys are automatically managed by the Function runtime. System Keys provide granular access to functions runtime features.

Show values

Filter system keys

Name	Value	
signalr_extension	Hidden value. Click to show value	Renew key value





Dashboard > scwsrSignalRAzureFunction

scwsrSignalRAzureFunction | Settings

SignalR

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Events

Settings

- Keys
- Quickstart
- Scale
- Settings
- CORS
- Identity
- Private endpoint connections
- Network access control
- Properties
- Locks

Monitoring

Save | Discard

Choose **Default** mode when all the hubs have hub servers, choose **Serverless** mode if not.
It usually takes up to 30 seconds to take effect.

Service Mode

Default | Serverless | Classic

Upstream Settings (Preview)

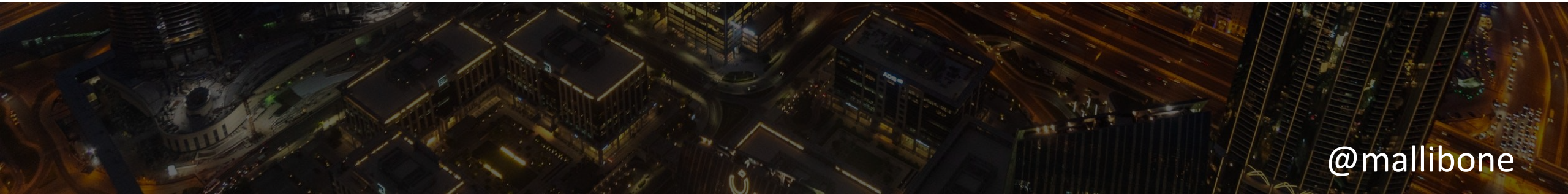
Upstreams allow external services to be notified when certain events for certain hubs happen. When the specified events happen, we'll send a POST request to the **first** matching URL pattern with the parameters evaluated. There are three supported parameters: `{hub}`, `{event}`, and `{category}`.
The matching rule supports three formats. Take the *Event Rules* as an example:

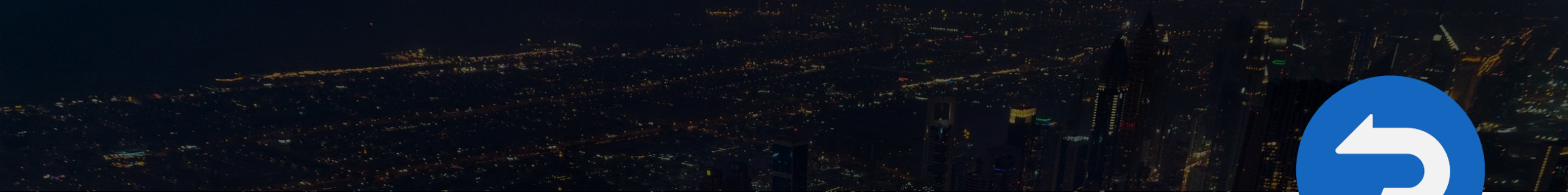
- Use *asterisk*(*) to match any event.
- Use *comma*(,) to join multiple events, for example, `connect,disconnect` matches events *connect* and *disconnect*.
- Use the full event name to match the event, for example, `connect` matches *connect* event.

Move up | Move down | Move to top | Move to bottom | Insert | Delete

Upstream URL Pattern	Hub Rules	Event Rules	Category Rules
https://signalr-gnabber-function.azurewebsites.net/runtime/webhooks/signalr?code=[redacted]	*	*	*

Add an upstream URL pattern





Dashboard > scwsrSignalRAzureFunction

scwsrSignalRAzureFunction | Settings

SignalR

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Events

Settings

- Keys
- Quickstart
- Scale
- Settings
- CORS
- Identity
- Private endpoint connections
- Network access control
- Properties
- Locks

Monitoring

Save | Discard

Choose **Default** mode when all the hubs have hub servers, choose **Serverless** mode if not.
It usually takes up to 30 seconds to take effect.

Service Mode

Default | Serverless | Classic

Upstream Settings (Preview)

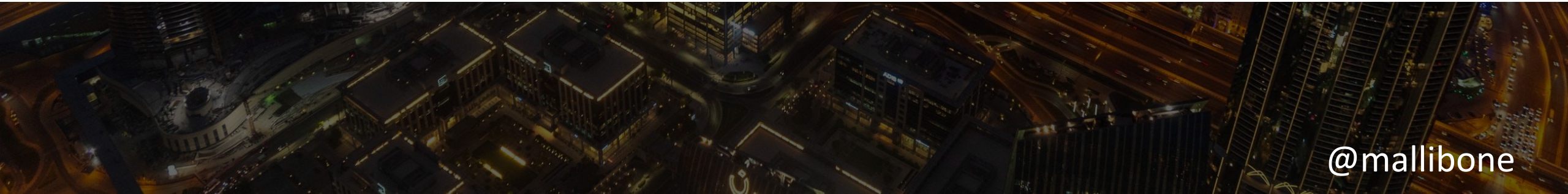
Upstreams allow external services to be notified when certain events for certain hubs happen. When the specified events happen, we'll send a POST request to the **first** matching URL pattern with the parameters evaluated. There are three supported parameters: `{hub}`, `{event}`, and `{category}`.
The matching rule supports three formats. Take the *Event Rules* as an example:

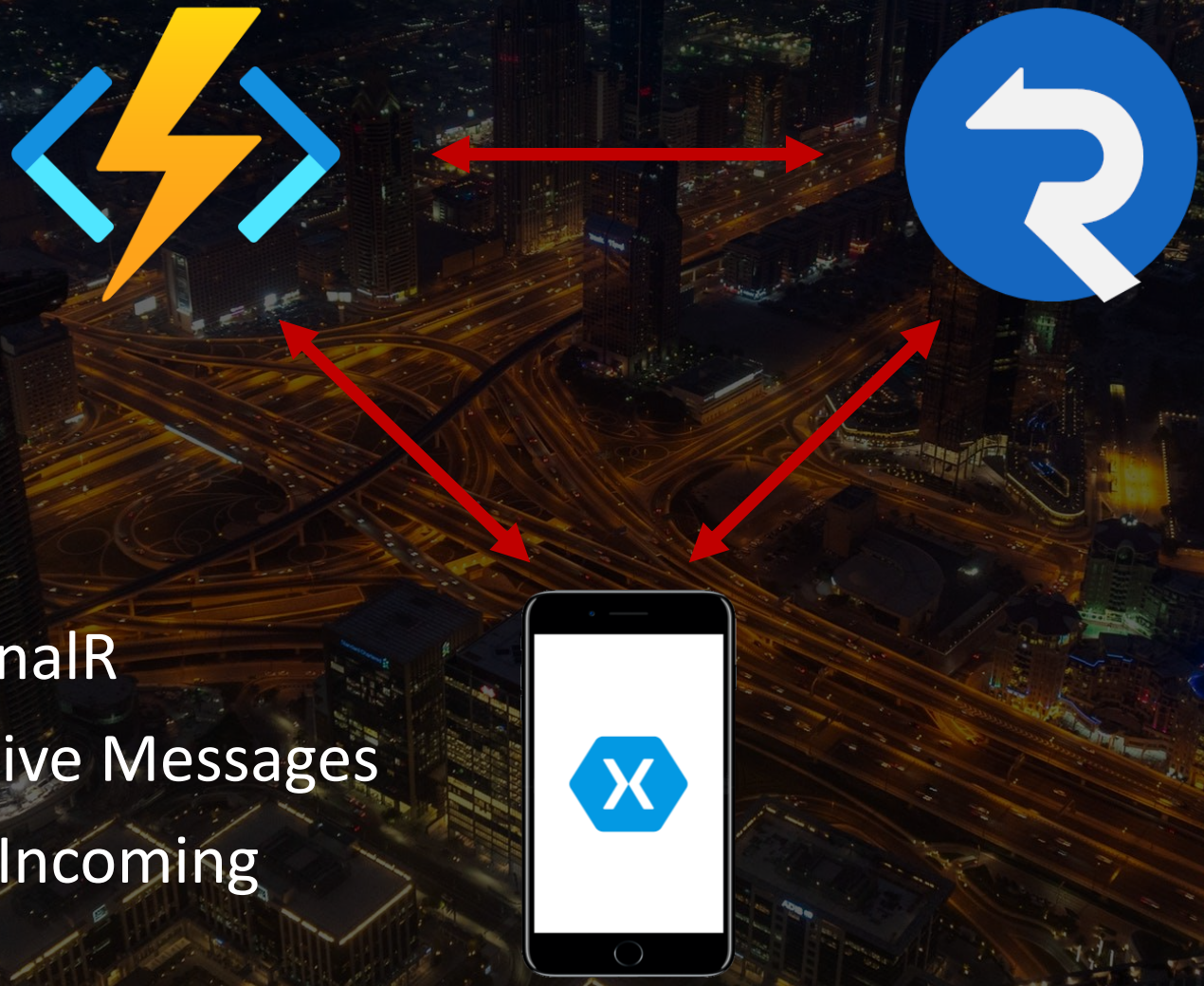
- Use *asterisk*(*) to match any event.
- Use *comma*(,) to join multiple events, for example, `connect,disconnect` matches events *connect* and *disconnect*.
- Use the full event name to match the event, for example, `connect` matches *connect* event.

Move up | Move down | Move to top | Move to bottom | Insert | Delete

Upstream URL Pattern	Hub Rules	Event Rules	Category Rules
https://signalr-gnabber-function.azurewebsites.net/runtime/webhooks/signalr?code=[redacted]	*	*	*

Add an upstream URL pattern

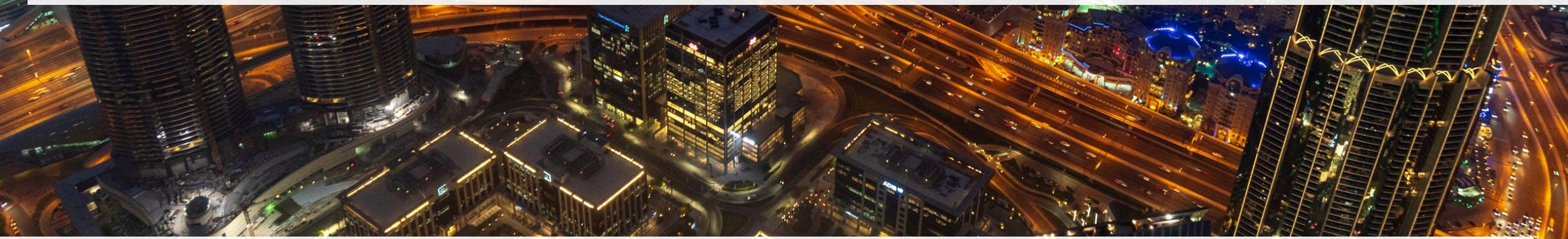




1. Connect to SignalR
2. Send and Receive Messages
3. Process/Relay Incoming messages

@mallibone

Security



CORS

Dashboard > Resource groups > StayConnectedWithSignalR > StayConnectedWithSignalRAzureFunction

StayConnectedWithSignalRAzureFunction | CORS

Function App

Search (Ctrl+/)

Push
Properties
Locks

App Service plan

App Service plan
Quotas
Change App Service plan

Development Tools

Console
Advanced Tools
App Service Editor (Preview)
Extensions

API

API Management
API definition
CORS

Monitoring

Alerts
Metrics
Health check
Logs

Save Discard

CORS

Cross-Origin Resource Sharing (CORS) allows JavaScript code running in a browser on an external host to interact with your backend. Specify the origins that should be allowed to make cross-origin calls (for example: http://example.com:12345). To allow all, use "*" and r
as part of domain or after TLD. Learn more

Request Credentials

Enable Access-Control-Allow-Credentials

Allowed Origins

https://functions.azure.com
https://functions-staging.azure.com
https://functions-next.azure.com
https://gnabber.rey-technology.com

CORS

Dashboard > Resource groups > StayConnectedWithSignalR > StayConnectedWithSignalRAzureFunction

StayConnectedWithSignalRAzureFunction | CORS

Function App

Search (Ctrl+/)

Save Discard

- Push
- Properties
- Locks

App Service plan

- App Service plan
- Quotas
- Change App Service plan

Development Tools

- Console
- Advanced Tools
- App Service Editor (Preview)
- Extensions

API

- API Management
- API definition
- CORS

Monitoring

- Alerts
- Metrics
- Health check
- Logs

CORS

Cross-Origin Resource Sharing (CORS) allows JavaScript code running in a browser on an external host to interact with your backend. Specify the origins that should be allowed to make cross-origin calls (for example: http://example.com:12345). To allow all, use "*" and
as part of domain or after TLD. Learn more

Request Credentials

Enable Access-Control-Allow-Credentials

Allowed Origins

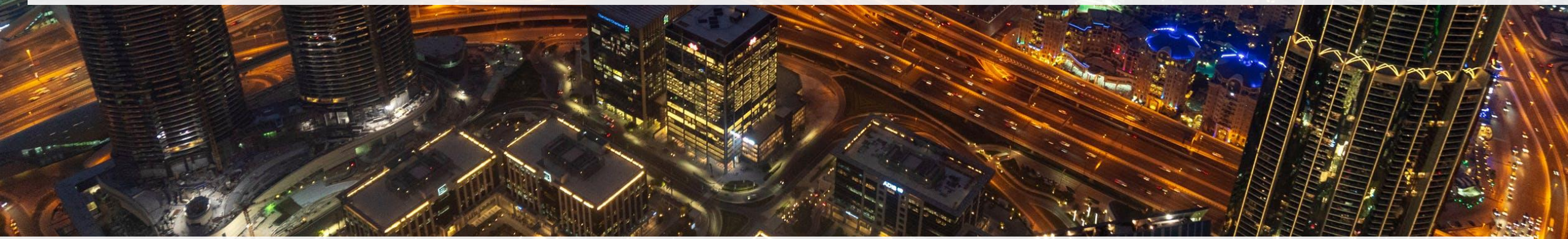
https://functions.azure.com

https://functions-staging.azure.com

https://functions-next.azure.com

https://gnabber.rey-technology.com

Authenticate the Azure Functions way



Authorization

```
public class SignalRChatDemo : ServerlessHub
{
    // ... Hub methods

    [FunctionAuthorize]
    [FunctionName(nameof(Broadcast))]
    public async Task MuteUser([SignalRTrigger]InvocationContext invocationContext, string userName,
                               TimeSpan muteDuration, ILogger logger)
    {
        // ... mute a user from the chat
    }

    // ... Hub methods
}
```

Authorization

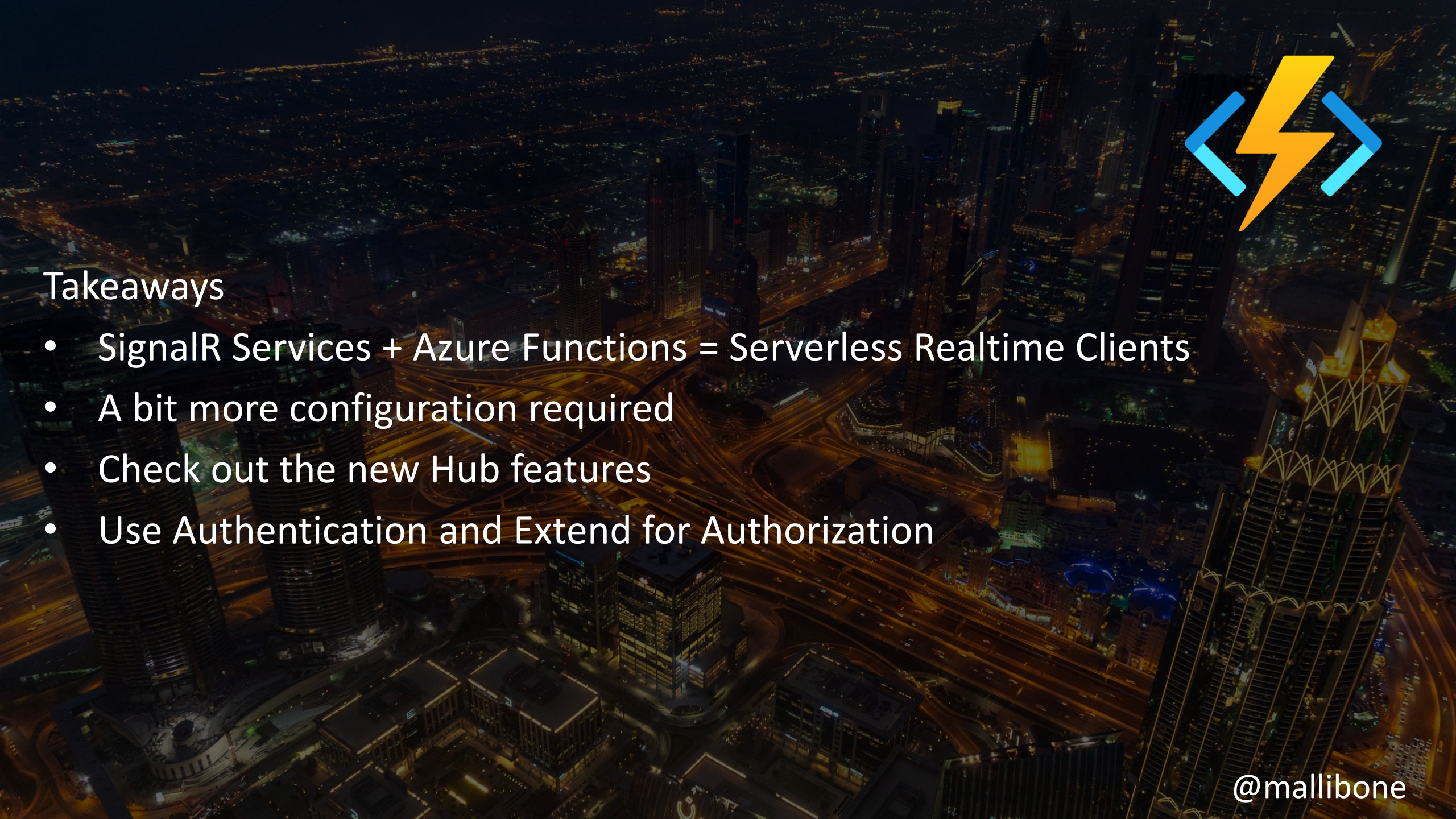
```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = true, Inherited = true)]
internal class FunctionAuthorizeAttribute : SignalRFilterAttribute
{
    public override Task FilterAsync(InvocationContext invocationContext, CancellationToken cancellationToken)
    {
        if (invocationContext.Claims.TryGetValue("admin", out var value)
            && bool.TryParse(value, out var isAdmin)
            && isAdmin)
        {
            return Task.CompletedTask;
        }

        throw new Exception($"{invocationContext.ConnectionId} doesn't have admin claim");
    }
}
```

Authorization

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = true, Inherited = true)]
internal class FunctionAuthorizeAttribute : SignalRFilterAttribute
{
    public override Task FilterAsync(InvocationContext invocationContext, CancellationToken cancellationToken)
    {
        if (invocationContext.Claims.TryGetValue("admin", out var value)
            && bool.TryParse(value, out var isAdmin)
            && isAdmin)
        {
            return Task.CompletedTask;
        }

        throw new Exception($"{invocationContext.ConnectionId} doesn't have admin claim");
    }
}
```

Authorization

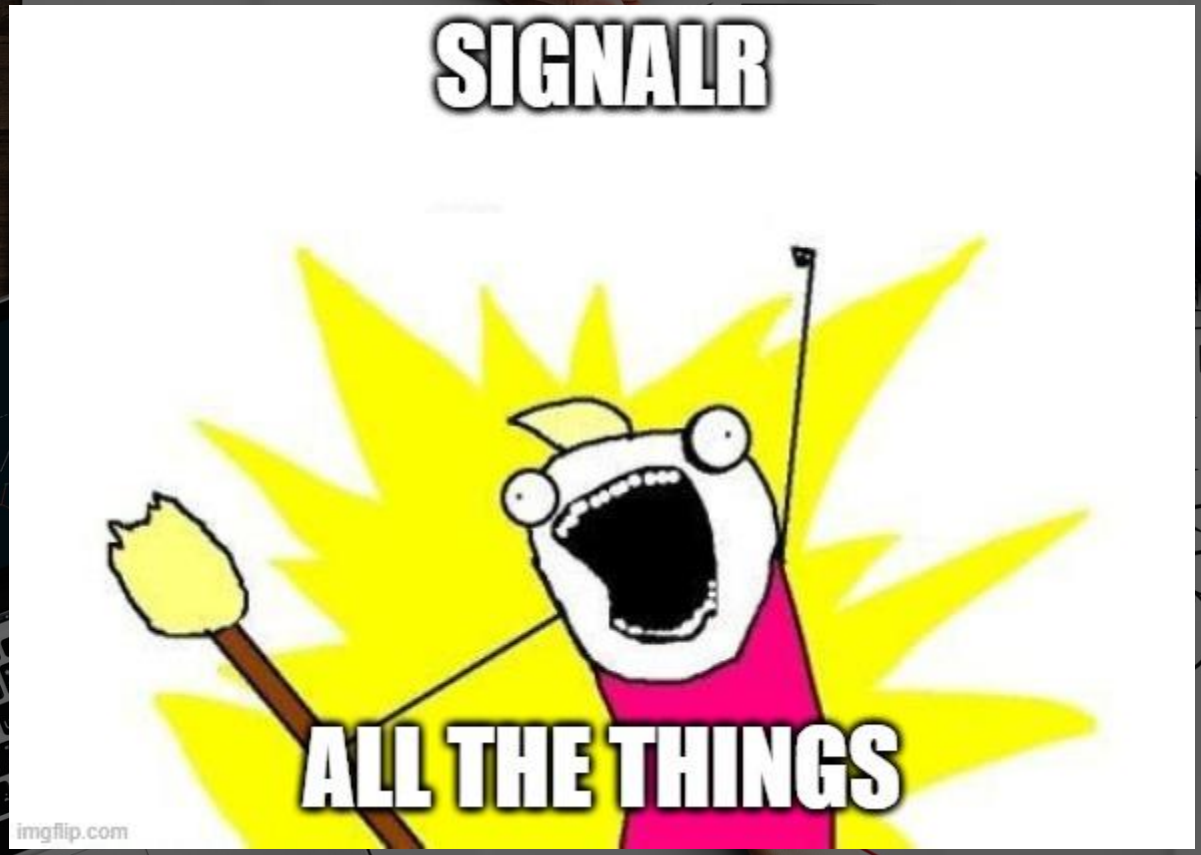
```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = true, Inherited = true)]
internal class FunctionAuthorizeAttribute : SignalRFilterAttribute
{
    public override Task FilterAsync(InvocationContext invocationContext, CancellationToken cancellationToken)
    {
        if (invocationContext.Claims.TryGetValue("admin", out var value)
            && bool.TryParse(value, out var isAdmin)
            && isAdmin)
        {
            return Task.CompletedTask;
        }

        throw new Exception($"{invocationContext.ConnectionId} doesn't have admin claim");
    }
}
```



Takeaways

- SignalR Services + Azure Functions = Serverless Realtime Clients
- A bit more configuration required
- Check out the new Hub features
- Use Authentication and Extend for Authorization

@mallibone



@mallibone

SignalR – Some yay and nay scenarios

- ✔ Content that updates in the background
- ✘ Static Content

- ✔ Web API extension
- ✔ Web and/or Platform Clients

- ✘ Streaming Large files
- ✘ Between Headless-Services

REY TECHNOLOGY

Thank you for your time!

Mark Allibone @mallibone

Rey Technology

https://nullpointers.io

https://mallibone.com

FEATURED COMMUNITY BLOG

Microsoft® MVP Most Valuable Professional