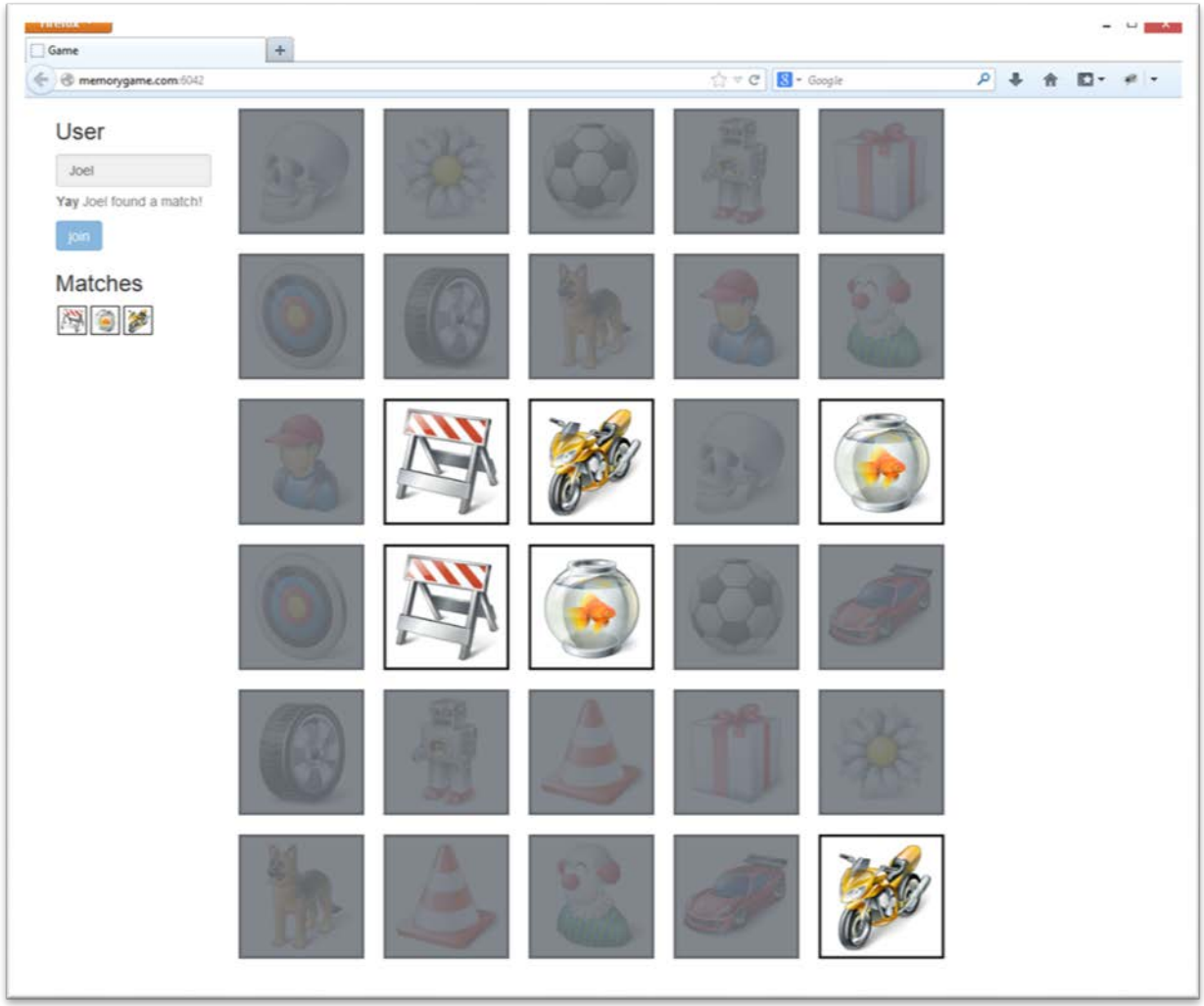
Testing and Scaleout

Joel Neubeck
http://joel.Neubeck.net
joel@neubeck.net

pluralsight
hardcore developer training

Testing Strategies



Testing Server - GameState

```
private static readonly Lazy<GameState>
  _instance = new Lazy<GameState>(
            () => new
  GameState(GlobalHost.ConnectionManager.GetHubC
  ontext<GameHub>()));
```

Testing Server - GameState

```
private static readonly Lazy<GameState>
public GameState(IHubContext context, bool testable)
{
    Clients = context.Clients;
    Groups = context.Groups;
}
private GameState(IHubContext context)
{
    Clients = context.Clients;
    Groups = context.Groups;
 }
```

Moching GameState with Moq

```
public class TestableHubContext : IHubContext
    {
        public TestableHubContext(IHubConnectionContext clients,
               IGroupManager groups)
        {
            Clients = clients;
            Groups = groups;
        }
        public IHubConnectionContext Clients { get; private set; }
        public IGroupManager Groups { get; private set; }
    }
```

Moching GameState with Moq

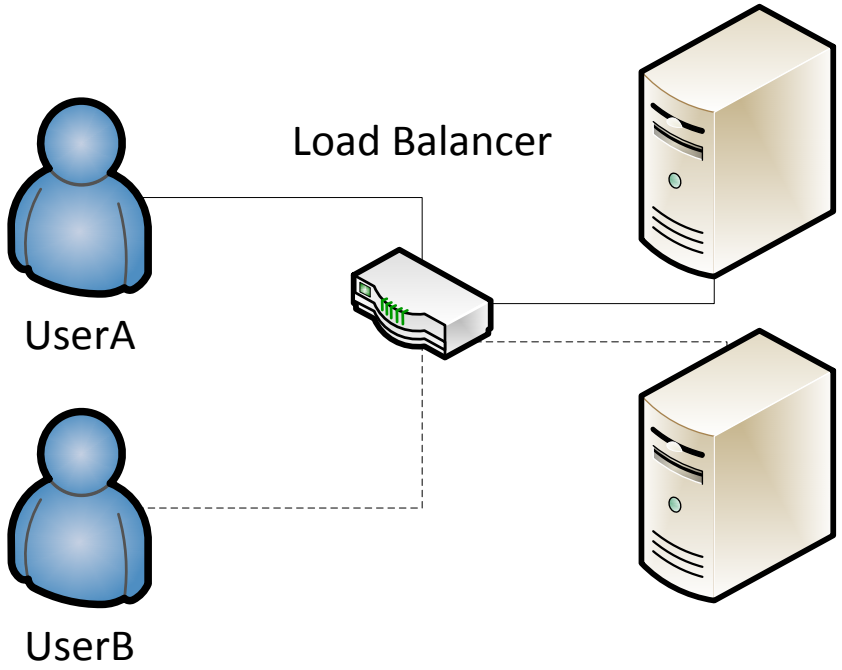
```
private GameState _state;
public void SetUpTests()
{
     var mockContext = new Mock<IHubContext>();
     IHubConnectionContext clients =
         new Mock<IHubConnectionContext>().Object;
     mockContext.Setup(mock => mock.Clients).Returns(clients);

     IGroupManager groups = new Mock<IGroupManager>().Object;
     mockContext.Setup(mock => mock.Groups).Returns(groups);

     var context = new TestableHubContext(clients, groups);
     _state = new GameState(context, true);
   }
```

Moching GameState with Moq

```
[TestMethod]
public void TestGameStateCreateGame()
{
    SetUpTests();
    const string userName = "joel";
    const string connectionId = "1234";

    var player1 = _state.CreatePlayer("Joel");
    var player2 = _state.CreatePlayer("Bob");

    var game = _state.CreateGame(player1, player2);

    Assert.AreEqual(player1.Group, player2.Group);
}
```

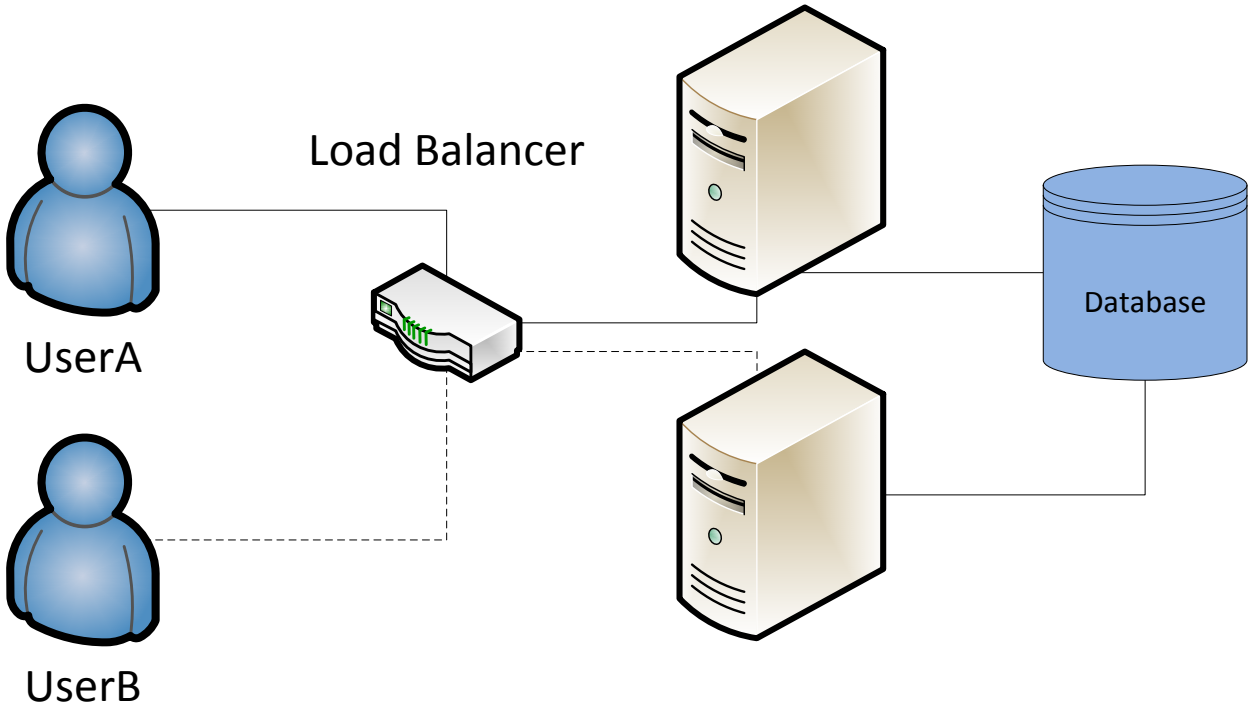
SignalR Scaleout Strategies

- **Scale Up**
  - Larger server with more RAM and CPU power
- **Scale Out**
  - More servers to handle load



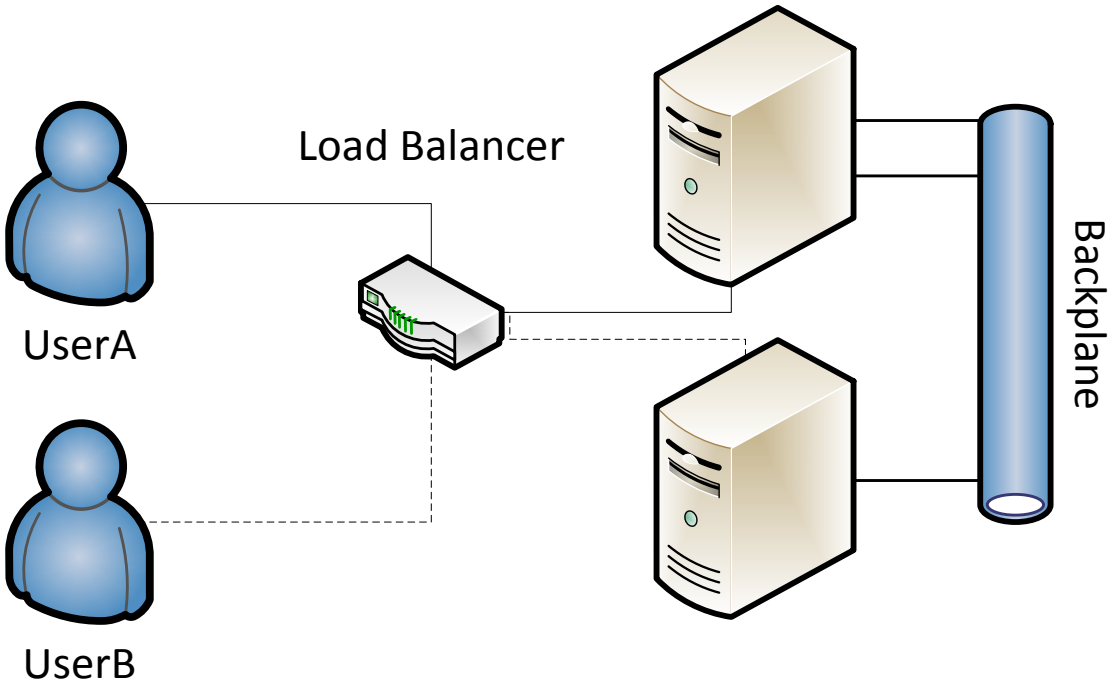
SignalR Scaleout – State Management

- **Move state to Database**
  - Players Table
  - Game Table



SignalR Scaleout - Backplane

- **Backplane**
  - A way in which each application instances sends messages so they are distributed to other application instances.



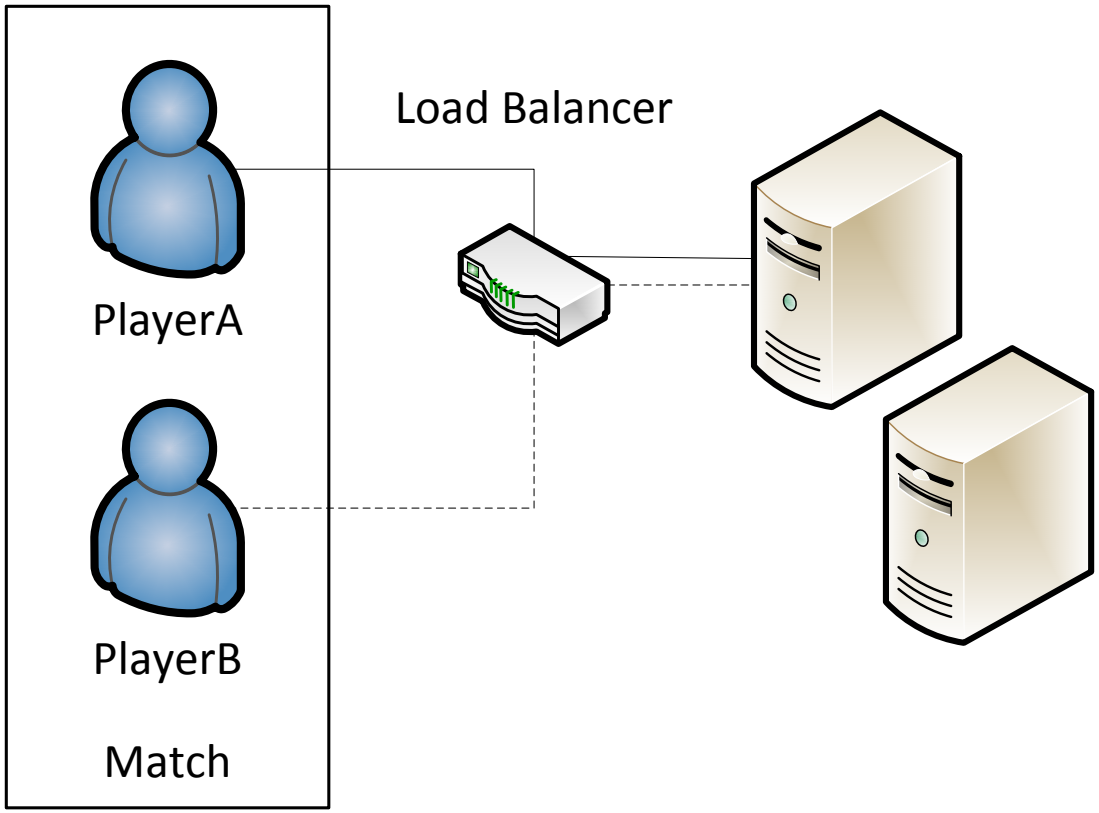
SignalR Scaleout - Backplanes

- **SignalR currently provides three backplanes:**
  - Windows Azure Service Bus
    - Messaging infrastructure
  - Redis
    - An in memory key-value store that uses a publish/subscribe pattern for sending messages
  - SQL Server
    - Service Broker messaging backed by a SQL table

Backplane Limitations

“Using a backplane, the maximum message throughput is lower than it is when clients talk directly to a single server node. That's because the backplane forwards every message to every node, so the backplane can become a bottleneck. “

Alternative to a Backplane



Course Summary

- **The following are some of the things we learned in this course:**
  - How to integrate SignalR into a ASP.Net MVC project
  - Maintain game state
  - Create and implement a SignalR Hub
  - Connect a SignalR hub to our client with Javascript
  - Test and Scale our application