Slide 1: Introduction to .NET Aspire

* Title: "Introduction to .NET Aspire"
* Content: ".NET Aspire is designed to improve the experience of building .NET cloud-native apps. It provides a consistent, opinionated set of tools and patterns that help you build and run distributed apps. .NET Aspire is designed to help you with orchestration, components, and tooling." **[1](https://learn.microsoft.com/en-us/dotnet/aspire/get-started/aspire-overview" \t "_blank)**

Slide 2: Why .NET Aspire?

* Title: "Why .NET Aspire?"
* Content: ".NET Aspire improves the experience of building .NET cloud-native apps by providing a consistent, opinionated set of tools and patterns. It helps with orchestration, components, and tooling, making it easier to build and run distributed apps." **[1](https://learn.microsoft.com/en-us/dotnet/aspire/get-started/aspire-overview" \t "_blank)**

Slide 3: .NET Aspire Components

* Title: ".NET Aspire Components"
* Content: ".NET Aspire components are NuGet packages designed to simplify connections to popular services and platforms. They handle many cloud-native concerns for you through standardized configuration patterns, such as adding health checks and telemetry." **[1](https://learn.microsoft.com/en-us/dotnet/aspire/get-started/aspire-overview" \t "_blank)**

Slide 4: .NET Aspire Project Templates

* Title: ".NET Aspire Project Templates"
* Content: ".NET Aspire provides two starter templates to help you get started: the .NET Aspire Application template, which includes the AspireSample.AppHost and AspireSample.ServiceDefaults projects, and the .NET Aspire Starter Application template, which includes additional boilerplate UI and API projects." **[1](https://learn.microsoft.com/en-us/dotnet/aspire/get-started/aspire-overview" \t "_blank)**

Slide 5: Setting Up .NET Aspire

* Title: "Setting Up .NET Aspire"
* Content: "To work with .NET Aspire, you'll need to install the .NET Aspire workload, which includes project templates and tooling experiences for Visual Studio and the dotnet CLI. You can install it using the Visual Studio installer or the dotnet workload install aspire command." **[2](https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/setup-tooling" \t "_blank)**

Slide 6: Deploying a .NET Aspire Application

* Title: "Deploying a .NET Aspire Application"
* Content: "The final artifacts of a .NET Aspire application are .NET apps and configuration that can be deployed to your cloud environments. With the strong container-first mindset of .NET Aspire, the .NET SDK native container builds serve as a valuable tool to publish these apps to containers with ease." **[3](https://devblogs.microsoft.com/dotnet/introducing-dotnet-aspire-simplifying-cloud-native-development-with-dotnet-8/" \t "_blank)**

Slide 7: Conclusion

* Title: "Conclusion"
* Content: "In conclusion, .NET Aspire is a powerful tool for building .NET cloud-native apps. It provides a consistent, opinionated set of tools and patterns, making it easier to orchestrate, manage components, and deploy applications. With its standardized project templates and tooling, developers can focus on building their applications rather than managing infrastructure."

Good day everyone, today I would like to introduce you to .NET Aspire, a cloud-optimized framework specifically tailored for developing observable, production-grade distributed applications. It's uniquely structured as an opinionated stack, ensuring cloud readiness and streamlined deployment. This framework is encapsulated within a suite of NuGet packages, each addressing distinct aspects of cloud-native application needs.

Emphasizing a microservices architecture, .NET Aspire facilitates the creation of applications composed of multiple small, interconnected components, rather than relying on a single, large codebase. This approach inherently supports extensive service integration, including databases, messaging systems, and caching services, essential for modern cloud-native applications.

In our journey today, we will delve into the key features of .NET Aspire, discuss its benefits, explore how to create an application using it, and understand its components and project structure. By the end of this session, you should have a clear understanding of .NET Aspire and how it simplifies cloud-native development. Let's begin our exploration.

.NET Aspire is a set of tools and guidelines designed specifically for creating advanced applications that are ready for the cloud environment. This means the apps you create with .NET Aspire are prepared to be deployed on cloud services, can handle complex tasks across multiple computers or services, and are easy to monitor and maintain.

Instead of having one big program, cloud-native applications (like those built with .NET Aspire) are usually made up of smaller, independent parts called microservices. These small parts work together but are built and managed separately. This approach makes it easier to handle and update different parts of the application without disrupting the whole system.

The apps are made to work across several computers or servers, often in different locations, to ensure they can handle many users and tasks efficiently. They are specifically designed to make the most of what cloud computing offers, like the ability to easily increase capacity, improve reliability, and manage the application effectively.

In summary, .NET Aspire helps developers build complex, efficient, and reliable cloud-based applications by breaking them down into smaller, manageable pieces that work together seamlessly.

Orchestration in the context of .NET Aspire means organizing and managing the different parts of a cloud-native application. It makes setting up and connecting the various pieces of your app easier by offering tools and shortcuts that handle the complex, behind-the-scenes work for you. This includes managing how different services find and communicate with each other, setting up environment variables, and configuring containers.

With .NET Aspire, you can:

1. \*\*App Composition\*\*: Define what makes up your application, including .NET projects, containers (isolated environments where your app can run), executables, and cloud resources. Think of it as listing out the ingredients and the recipe for your application.

2. \*\*Service Discovery and Connection String Management\*\*: The application needs to connect different services, like databases or other apps. .NET Aspire simplifies this by automatically providing the necessary information (connection strings) and mechanisms (service discovery) to the parts of your app that need them. This makes it easier for developers to focus on building the app rather than getting bogged down in configuration details.

**Here will be nice an example of dataprotection keys stored in db (two projects will share same connection string)**

.NET Aspire components are like add-ons or plugins that you can use to easily connect your application to commonly used services and platforms, such as Redis (a data storage system) or PostgreSQL (a database system). These components take care of many routine tasks and settings for you. They ensure your application can automatically check its own health, gather useful data on how it's performing, and easily connect with other services.

These components are designed to be used together efficiently. They automatically share configuration details with other parts of your application. So, if one part of your application (Example.ServiceFoo) uses another part (Example.ServiceBar), .NET Aspire ensures that ServiceFoo gets all the necessary settings and information it needs from ServiceBar to work properly. This helps your entire application communicate and function smoothly without you having to manually set up each connection and configuration.

.NET Aspire provides a comprehensive suite of tools and processes that cover every aspect of creating and managing cloud applications. It does so by using what's called an app model, a framework that defines all the components of your application and how they interact. This includes everything from the code projects and programs that make up your application to containers (isolated environments where parts of your app run) and any external services or cloud infrastructure your app needs.

Within each .NET Aspire application, there's a special project known as the App host project. This is where you define the structure and behavior of your application using the app model. You do this through a specific set of instructions or methods provided by .NET Aspire, which you can access using something called the IDistributedApplicationBuilder. This builder is like a toolbox that allows you to specify how your application should be built, configured, and run, and you get it by running a command called DistributedApplication.CreateBuilder.

<https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/app-host-overview>

<https://learn.microsoft.com/en-us/dotnet/aspire/service-discovery/overview>