Question 1. What is .NET Aspire and what problem does it aim to solve?

✅ *.NET Aspire is a cloud-native development stack by Microsoft that simplifies building, orchestrating, and observing microservice-based .NET applications. It streamlines service configuration, communication, and deployment.*

Question 2. Explain the role of the AppHost project in .NET Aspire.

✅ *The AppHost serves as the main orchestration project. It wires up all services and dependencies, manages lifecycle, injects configs, and launches the Aspire Dashboard for visibility.*

Question 3. What are ServiceDefaults in .NET Aspire?

✅ *AddServiceDefaults() applies a set of baseline configurations like health checks, OpenTelemetry tracing, and retry policies to your services—ensuring consistency and resiliency by default.*

Question 4. How does .NET Aspire simplify service-to-service communication?

✅ *Aspire allows logical service referencing using WithReference(), which injects endpoint details dynamically. This eliminates hardcoded URLs and supports environment-agnostic communication.*

Question 5. What is the Aspire Dashboard used for?

✅ *It provides a visual, real-time interface to monitor all your Aspire services—including logs, traces, metrics, health status, and configuration details.*

🧠 Conceptual & Best Practices

Question 6. How does .NET Aspire handle configuration management across services?

✅ *It uses centralized environment variables and appsettings.json files, automatically propagated across services using DI, keeping configuration clean and consistent.*

Question 7. Describe the observability features built into .NET Aspire.

✅ *Aspire offers built-in logging, distributed tracing (via OpenTelemetry), service metrics, health monitoring, and visual diagnostics—all integrated through the Aspire Dashboard.*

Question 8. What is the purpose of WithReference in Aspire?

✅ *It links services together by name, enabling automatic resolution of endpoints, shared configuration, and inter-service communication without manual wiring.*

Question 9. How does Aspire support health checks and diagnostics?

✅ *By default, Aspire exposes /health endpoints when using AddServiceDefaults(). These are visualized in the dashboard and help validate system readiness and uptime.*

Question 10. What are the benefits of Aspire over Docker Compose or Kubernetes for local development?

✅ *Aspire eliminates YAMLs and low-level config by offering code-first orchestration. It's faster for local devs, integrates with .NET tooling, and provides real-time service insight.*

🧩 Scenario-Based Questions

Question 11. How would you make an existing microservice resilient using Aspire?

✅ *Add AddServiceDefaults() to apply default retries and health checks, and configure Polly-based resilience policies using AddResilienceHandler().*

Question 12. How do you integrate Polly in a .NET Aspire app?

✅ *Use builder.AddResilienceHandler("name") with .Configure(...) to define policies like retries, timeouts, or circuit breakers. Apply them to outgoing service calls.*

Question 13. You're asked to connect SQL Server and Redis in Aspire. What do you do?

✅ *Use AddSqlServer() and AddRedis() inside AppHost. Aspire starts containers, discovers endpoints, and injects connection strings automatically via DI.*

Question 14. If a microservice crashes, how can Aspire observability help?

✅ *Use the dashboard to view service logs, traces, and health status. You can trace calls across services and identify the failure source quickly.*

🧪 Code-Based Questions

Question 15. What does this code do?

```
builder.AddProject<Projects.MyApiService>("my-api")
       .WithReference(api =>
api.WithEnvironment("ASPNETCORE_ENVIRONMENT", "Development"));
```

✅ *It adds the MyApiService project to AppHost, sets the environment variable, and registers it for inter-service discovery.*

Question 16. When would you use .AddHttpClientResilienceHandler()?

✅ *When calling external APIs or internal services where network failures are possible. It helps ensure graceful degradation and retry logic.*

🚀 Deployment & Testing

Question 17. How do you deploy an Aspire app using Azure Developer CLI?

✅ *Run "azd init", set up your project, and deploy with azd up. This provisions Azure resources and pushes your Aspire app to the cloud in one step.*

Question 18. How does Aspire simplify integration testing?

✅ *It enables you to spin up the full app graph (AppHost + services) inside test projects, making real end-to-end tests easy to run and debug locally.*

Question 19. What makes Aspire CI/CD ready?

✅ *Its structured project model, health checks, tracing, and built-in config support make it easy to plug into CI/CD pipelines without complex container orchestration.*

Question 20. How do you run and validate everything from one place in Aspire?

✅ *Just run the AppHost project—this starts all services, connects them via DI and references, applies policies, and launches the dashboard for full visibility.*