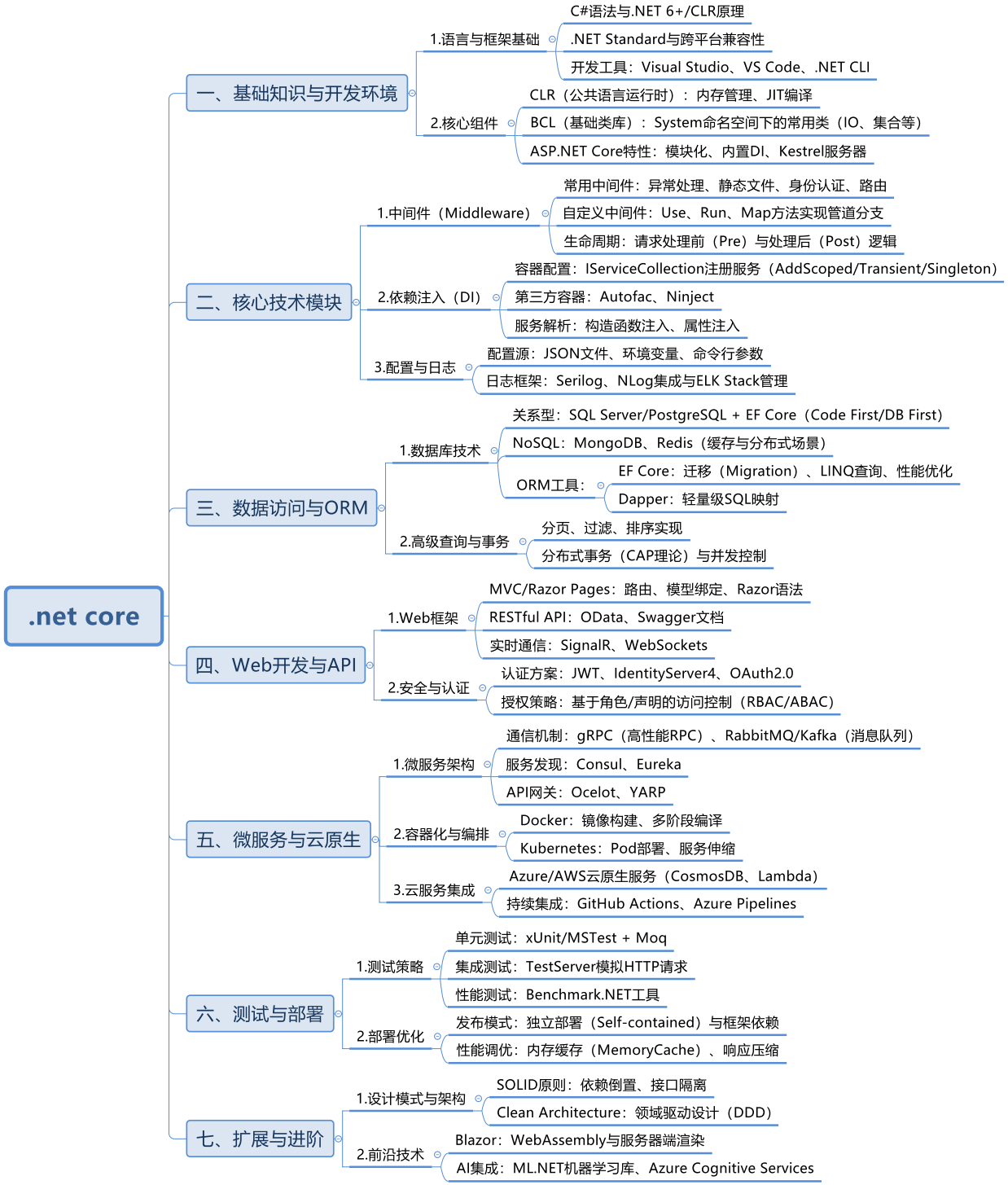
.net core
一、基础知识与开发环境
1.语言与框架基础
C#语法与.NET 6+/CLR原理
.NET Standard与跨平台兼容性
开发工具：Visual Studio、VS Code、.NET CLI
2.核心组件
CLR（公共语言运行时）：内存管理、JIT编译
BCL（基础类库）：System命名空间下的常用类（IO、集合等）
ASP.NET Core特性：模块化、内置DI、Kestrel服务器
二、核心技术模块
1.中间件（Middleware）
常用中间件：异常处理、静态文件、身份认证、路由
自定义中间件：Use、Run、Map方法实现管道分支
生命周期：请求处理前（Pre）与处理后（Post）逻辑
2.依赖注入（DI）
容器配置：IServiceCollection注册服务（AddScoped/Transient/Singleton）
第三方容器：Autofac、Ninject
服务解析：构造函数注入、属性注入
3.配置与日志
配置源：JSON文件、环境变量、命令行参数
日志框架：Serilog、NLog集成与ELK Stack管理
三、数据访问与ORM
1.数据库技术
关系型：SQL Server/PostgreSQL + EF Core（Code First/DB First）
NoSQL：MongoDB、Redis（缓存与分布式场景）
ORM工具：
EF Core：迁移（Migration）、LINQ查询、性能优化
Dapper：轻量级SQL映射
2.高级查询与事务
分页、过滤、排序实现
分布式事务（CAP理论）与并发控制
四、Web开发与API
1.Web框架
MVC/Razor Pages：路由、模型绑定、Razor语法
RESTful API：OData、Swagger文档
实时通信：SignalR、WebSockets
2.安全与认证
认证方案：JWT、IdentityServer4、OAuth2.0
授权策略：基于角色/声明的访问控制（RBAC/ABAC）
五、微服务与云原生
1.微服务架构
通信机制：gRPC（高性能RPC）、RabbitMQ/Kafka（消息队列）
服务发现：Consul、Eureka
API网关：Ocelot、YARP
2.容器化与编排
Docker：镜像构建、多阶段编译
Kubernetes：Pod部署、服务伸缩
3.云服务集成
Azure/AWS云原生服务（CosmosDB、Lambda）
持续集成：GitHub Actions、Azure Pipelines
六、测试与部署
1.测试策略
单元测试：xUnit/MSTest + Moq
集成测试：TestServer模拟HTTP请求
性能测试：Benchmark.NET工具
2.部署优化
发布模式：独立部署（Self-contained）与框架依赖
性能调优：内存缓存（MemoryCache）、响应压缩
七、扩展与进阶
1.设计模式与架构
SOLID原则：依赖倒置、接口隔离
Clean Architecture：领域驱动设计（DDD）
2.前沿技术
Blazor：WebAssembly与服务器端渲染
AI集成：ML.NET机器学习库、Azure Cognitive Services