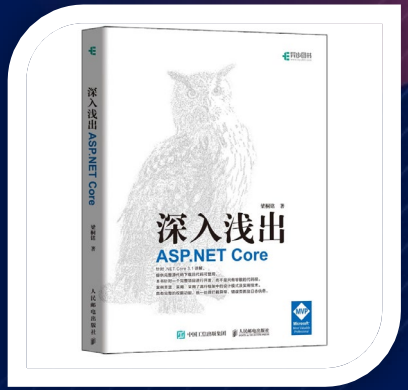
.NET

云原生开发挑战赛

暨.NET20周年特别活动-51Aspx

Upgrade to Dotnet6.0

梁桐铭



- 微软最有价值专家（Microsoft MVP）
- 华为云MVP
- 畅销书《深入浅出ASP.NET Core》作者

  华为云云享专家

  52ABP 框架作者

  2018 Microsoft Tech Summit 讲师

  2020中国DevOps大会讲师
- Microsoft AI Open Hack 教练
- 2015 年开始在国内推广 ASP.NETBoilerplate Project 开源框架。
- 欢迎关注公众号“梁桐铭博客”，分享技术又不止技术。

云原生 .NET 6 实践
Blazor DevOps 容器化

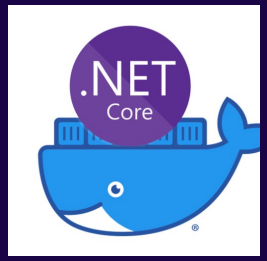
梁桐铭 – 2022年4月7日

内容概述

- 使用 ASP.NET Core Blazor 作为项目案例展示
- 案例会托管在的 Github 平台中，然后使用通过代码提交的时候触发流水线。
- 流水线会自动化的去执行 CI 编译构建功能，然后制作案例为 Docker 镜像推送到容器镜像服务中。
- 自动化将Docker镜像部署服务到 Linux 虚拟机的容器化的完整操作流程。


GitHub


P.NET Core Bla


.NET
Core


ubuntu

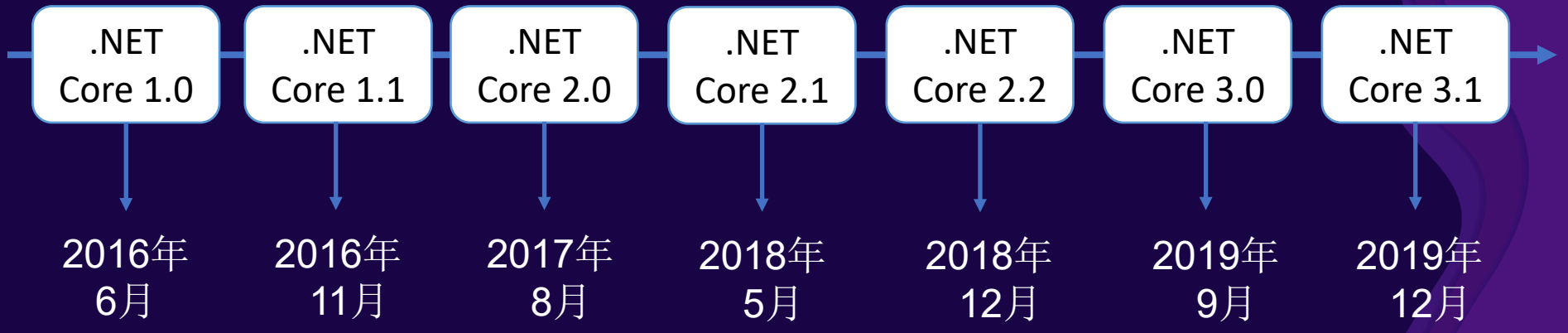
技术链条工具


GitHub Container Registry

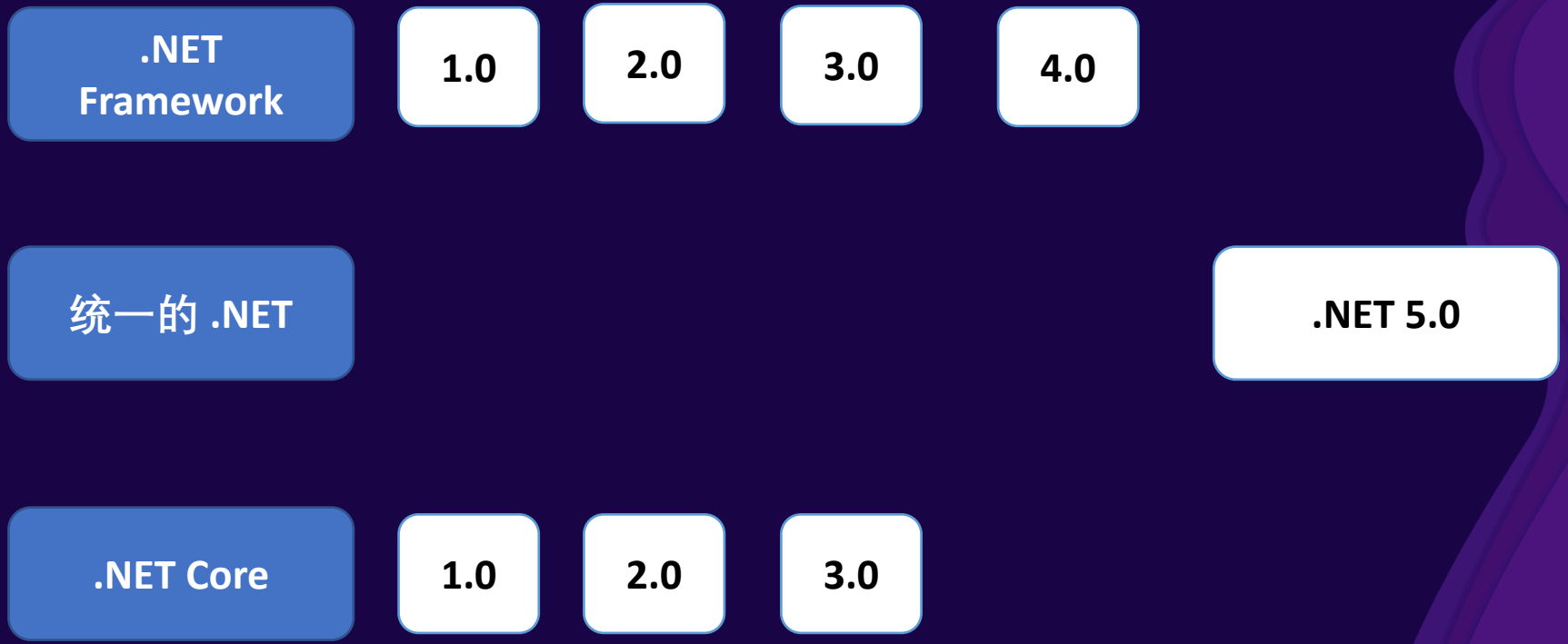

GitHub Actions

.NET 平台的演进

.NET Core 发行版本回顾



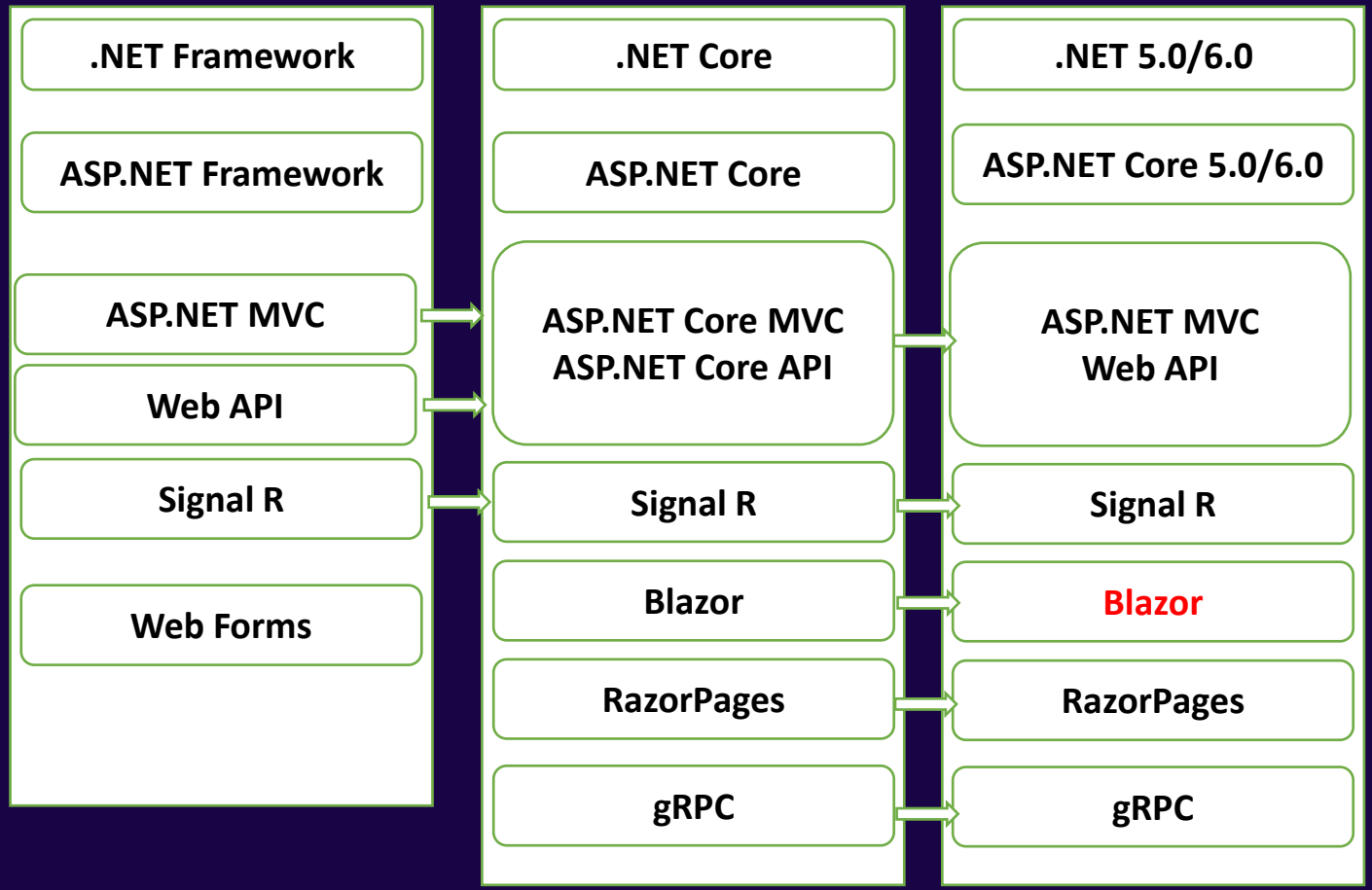

.NET
Core 3.1
.NET 5
.NET 6


.NET Framework
1.0
2.0
3.0
4.0
统一的 .NET
.NET 5.0
.NET Core
1.0
2.0
3.0

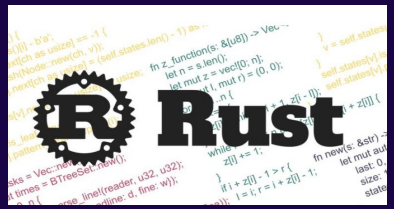
.NET 6.0 发布

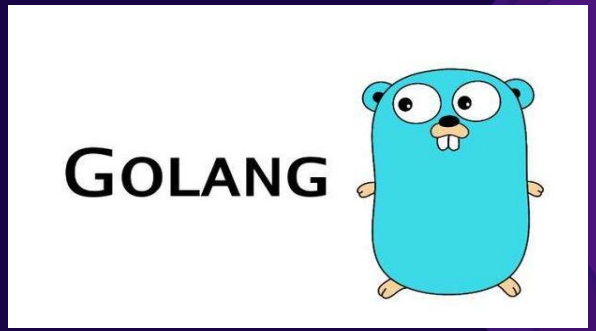
- 性能提升，EF Core的整体性能提升92%
- 支持C# 10.0的语法
- VS2022与.NET CLI中加入热重载
- dotnet monitor 加入云检测增强对系统的监控
- Blazor 组件针对JavaScript的加强、WebAssembly增加了对AOT的编译
- IL反编译代码增加的更加强大
- .NET 6 是一个长期支持 (LTS) 版本，将支持3年，支持所有的系统。
- 升级更简单，对于大多数项目而言，从.NET Core3.1、.NET 5.0 修改下项目文件中的版本号即可完成升级。

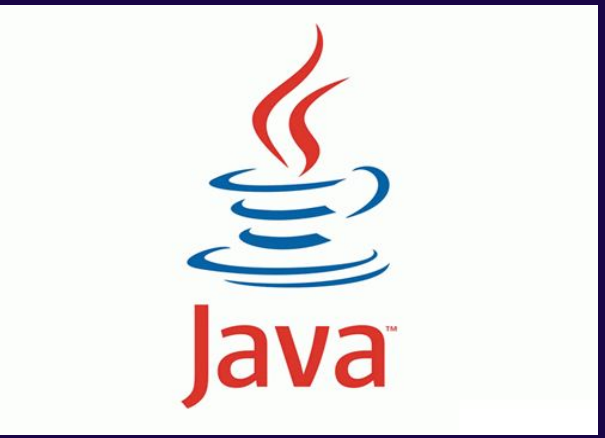
.NET 差异变化

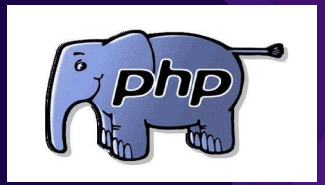

.NET Framework
ASP.NET Framework
ASP.NET MVC
Web API
Signal R
Web Forms
.NET Core
ASP.NET Core
ASP.NET Core MVC
ASP.NET Core API
Signal R
Blazor
RazorPages
gRPC
.NET 5.0/6.0
ASP.NET Core 5.0/6.0
ASP.NET MVC
Web API
Signal R
Blazor
RazorPages
gRPC

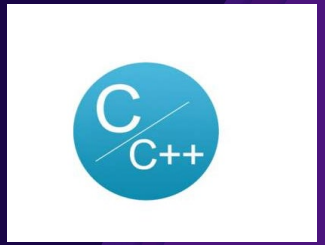
如何开发一个Web应用程序



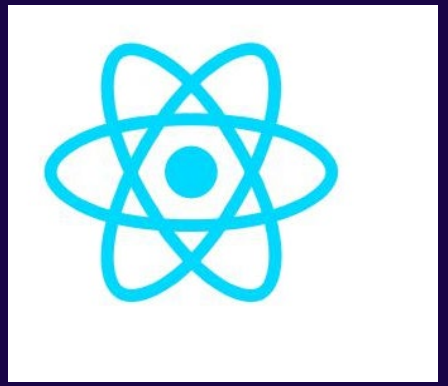


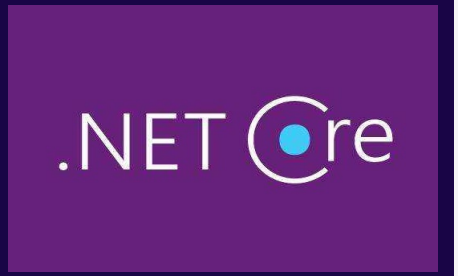






技术框架的选择





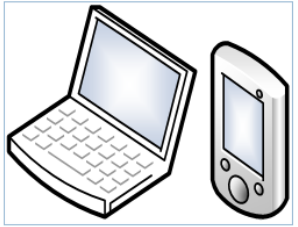
如何开发一个Web应用程序

服务端

C#
Java
PHP
Go

前端

Angular
React
Vue



是否可以用一种语言或框架就可以满足服务器端和客户端开发?

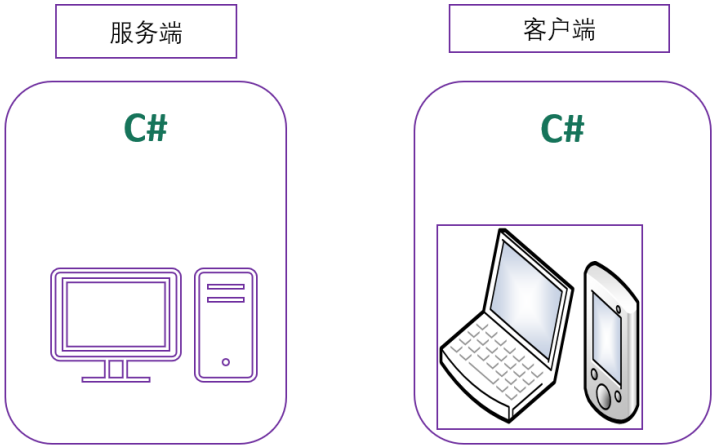
可以

一种语言满足开发需求

让C#代码可以在服务器和客户端浏览器中执行。

这意味着现有的.Net开发人员可以重复使用他们的C#技能，而不是学习新的JavaScript框架及其巨大的学习曲线。

- **你可能会说你撒谎，浏览器只能理解并执行JavaScript。**
- **我们如何在客户端浏览器中执行C#代码呢？**
- 答案就是WebAssembly



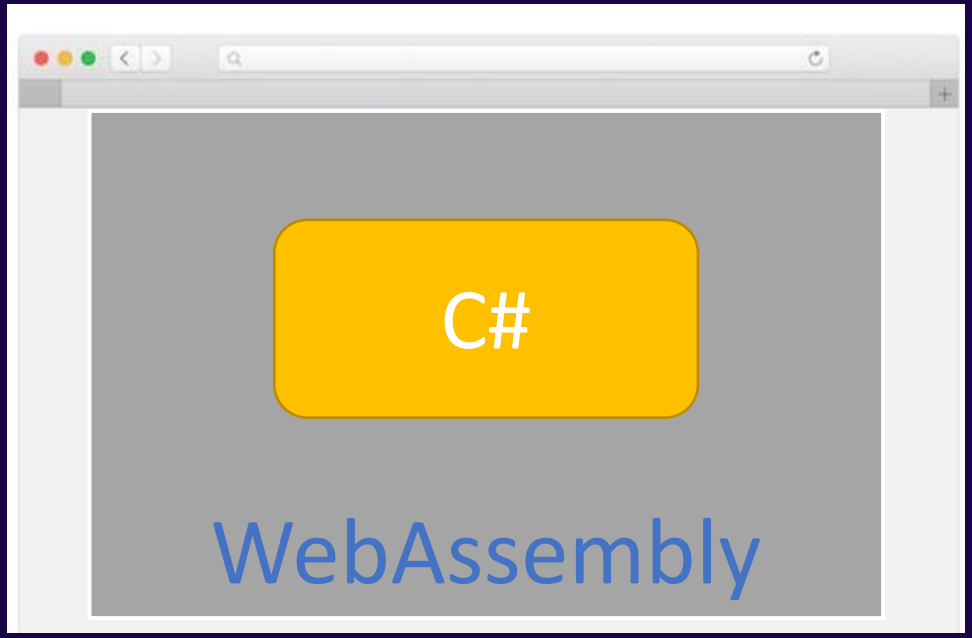
WebAssembly优点

- 文件加载快 - WebAssembly 文件体积更小，所以下载速度更快。
- 解析 - 解码 WebAssembly 比解析 JavaScript 要快
- 编译和优化 - 编译和优化所需的时间较少，因为在将文件推送到服务器之前已经进行了更多优化，JavaScript 需要为动态类型多次编译代码
- 性能高 - 执行可以更快，WebAssembly 指令更接近机器码

- 网页版本的坦克大战：
- https://www.wasm.com.cn/demo/

使用C#进行服务器端和客户端开发

Blazor 和 Webassembly



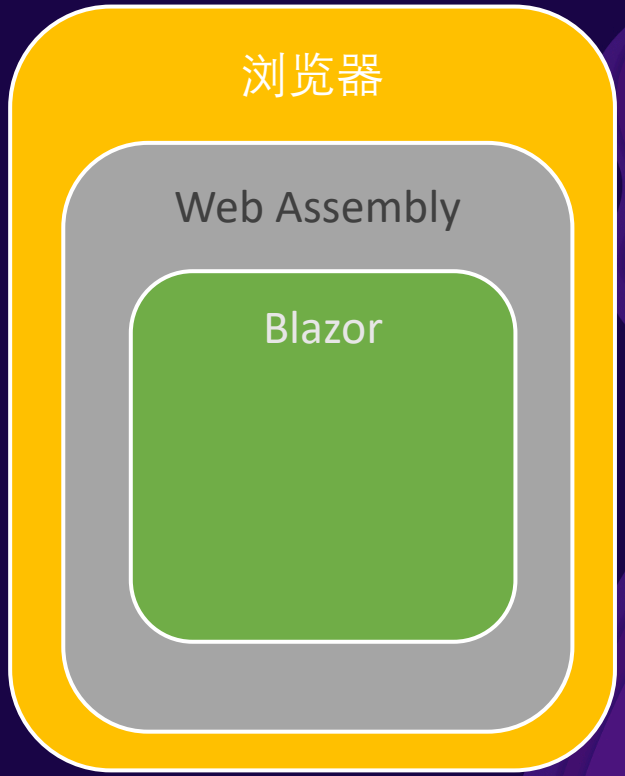
Blazor是通过C#编写的，它可以直接在浏览器中使用WebAssembly运行C#代码。

Webassembly 运行时

它和Angular、React、Vue等JavaScript框架一样，在相同的安全沙盒中运行。

不仅仅是C#，事实上，我们可以使用WebAssembly在浏览器中运行任何类型的代码。

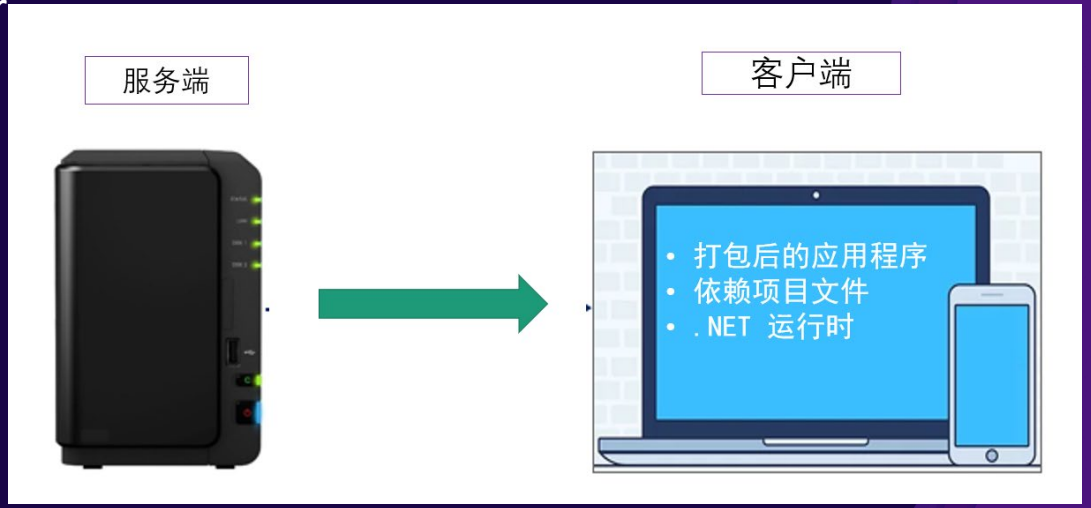
- C、C++、Rust



Blazor的托管模型- WebAssembly托管模式

客户端托管模式即SPA模式与Angular、Vue、React一致，在这种模式下，应用程序直接在浏览器中运行在WebAssembly上。

一个Blazor WebAssembly应用程序可以完全在客户端上运行，而不需要连接到服务器，或者我们可以选择使用Web API调用或SignalR来配置它与服务器进行交互。



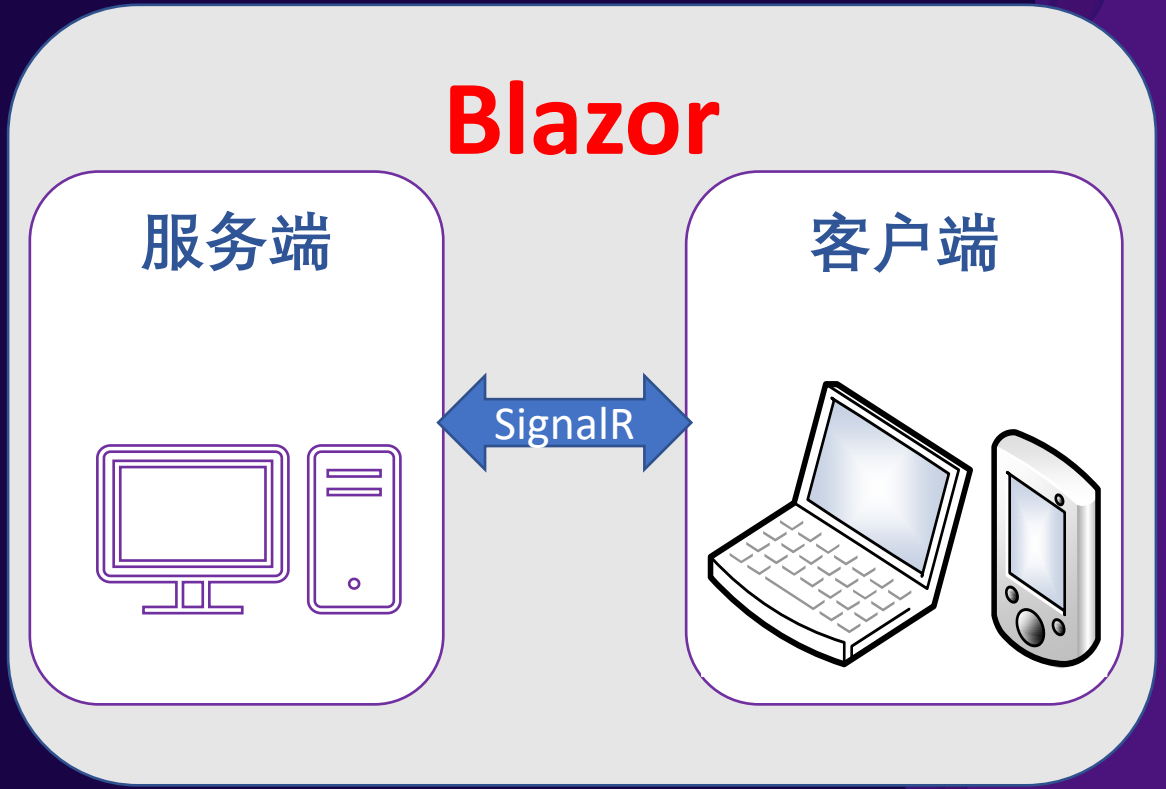
Blazor的托管模型- Blazor Server 模式

Blazor服务器端在将HTML内容发送到客户端的浏览器之前，会预先渲染HTML内容。这使得它对搜索引擎友好，而且没有明显的启动时间。

由于下载量明显小于Blazor WebAssembly应用程序，因此加载速度更快

客户端只需要一个浏览器就可以使用该应用。即使是不支持WebAssembly的浏览器也可以使用。

Blazor服务器端应用程序可以在旧的浏览器上运行，因为不需要Web汇编，只需要HTML和JavaScript。



周边的组件生态

Blazorise Demo

Ant Design Blazor

MatBlazor - Material Design

Bootstrap of Blazor

Bootstrap 风格的 Blazor UI 组件库

基于 Bootstrap 样式库精心打造，并且额外增加了 70 多种常用的组件，为您快速开发项目带来非一般的感觉

致力于打造全网最简单实用的|

最好玩的组件库

Awesome Blazor awesome

> A collection of awesome Blazor resources.

Blazor is a .NET web framework using C#/Razor and HTML that runs in the browser with WebAssembly.

Contributions are always welcome! Please take a look at the contribution guidelines pages first. Thanks to all contributors, you're awesome and wouldn't be possible without you!

If you need to search on this list you can try this great website: Awesome Blazor Browser.
Thanks @jsakamoto for this! Source code stars 34 last commit last wednesday

简谈云原生

云原生的特点是：构建应用简便快捷,基于微服务架构提高灵活性和可维护性，借助敏捷方法、DevOps支持持续迭代和运维自动化，

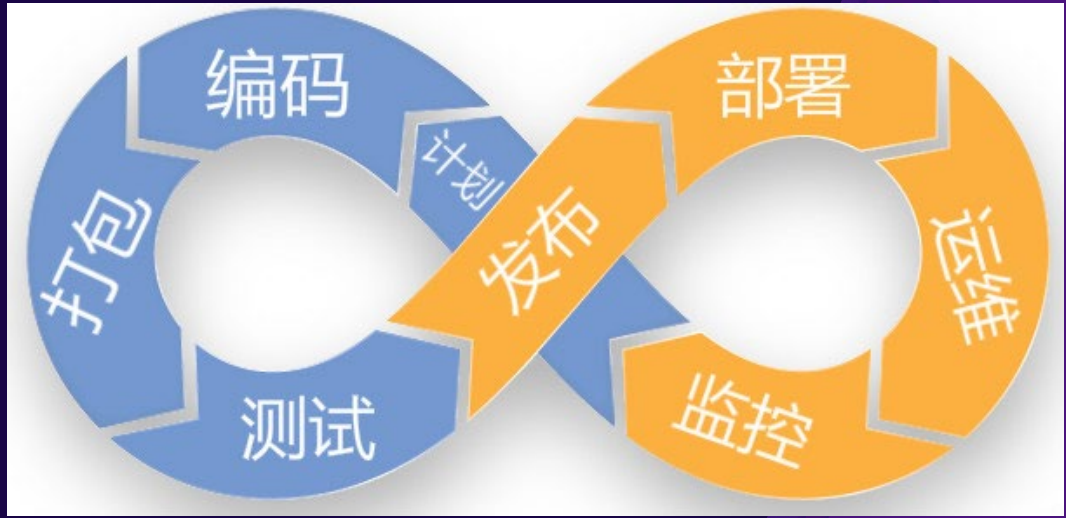
利用云平台设施实现弹性伸缩、动态调度、优化资源利用率。

云原生的四个要素:微服务、容器化、DevOps、持续交付

DevOps开发理念

- DevOps 最早开发（Development） 和运维（Operations）这两个领域的合并，随着范围的扩大和理念的完善现在包含了QA、产品等领域。

- 它的核心作用是打破部门墙，DevOps 的主要特点是：跟谁共同工作、如何共同工作。致力于把不同部门不同分工的人召集到一起，共同努力解决问题。将软件的整个开发生命周期中的持续开发，持续测试，持续集成，持续部署和持续监控，进行协同。



为什么您的团队需要 CI/CD 工作流

什么是 CI?

- **持续集成**是将代码集成到代码仓库中并尽可能早地自动构建/测试每个更改的做法 - 通常一天几次。

什么是 CD?

- **持续交付是**指该软件可以随时发布到生产环境中，通常是通过自动推送更改到过渡测试系统。
- **持续部署则**更进一步，自动将更改推送到生产环境。

Docker解决的问题

- 我们开发的软件项目，程序员自己从头到尾搭建了一套环境开始写代码,写完代码后程序员要把代码交给测试同学测试，

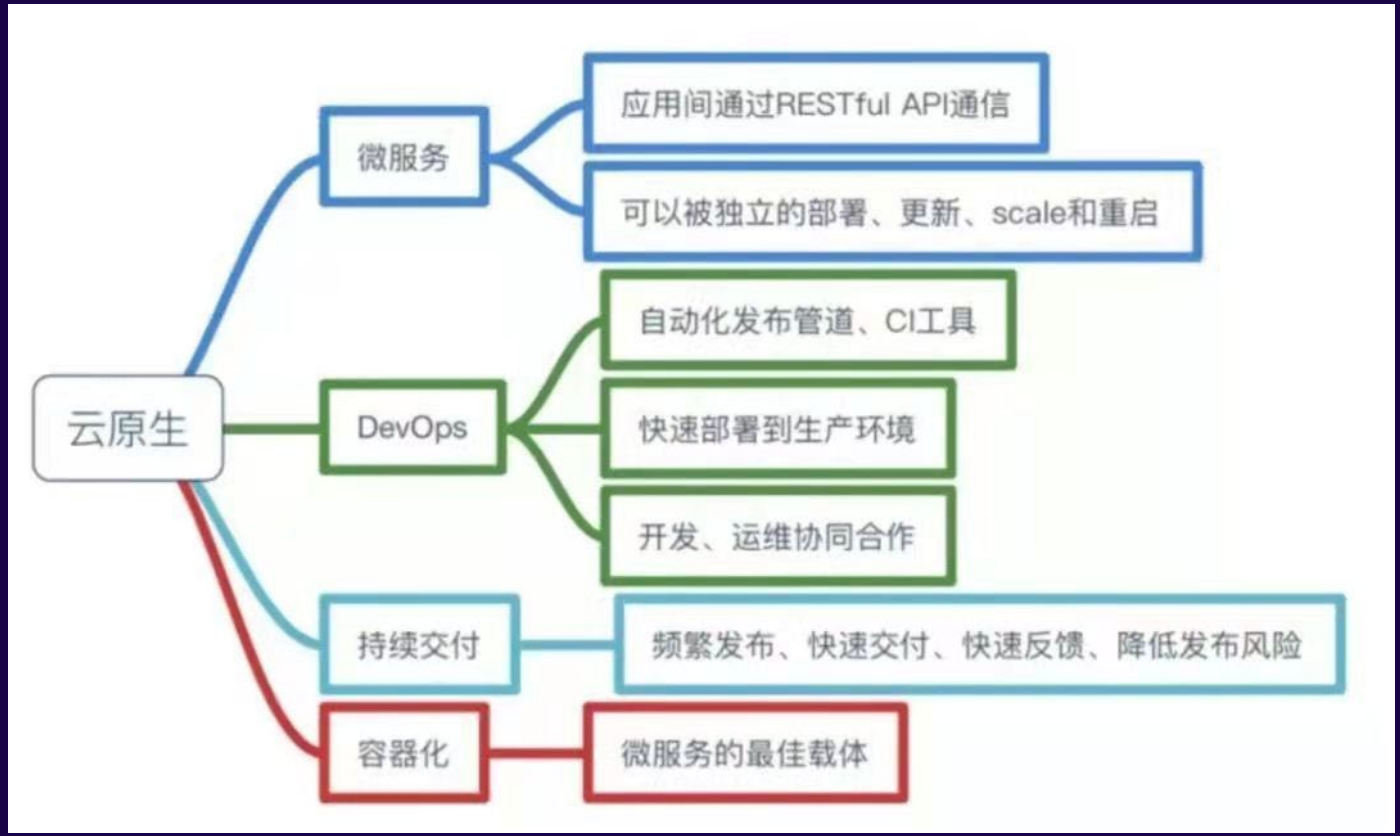
- 这时测试同学开始从头到尾搭建这套环境，测试过程中出现问题，程序员表示很无奈，因为“明明在人家的环境上可以运行的”。

- 测试同学测完后终于可以上线了，这时实施同学又要重新从头到尾搭建这套环境，费了九牛二虎之力搭建好环境开始上线，
- 糟糕，上线系统就崩溃了，这时心理素质好的程序员又可以施展演技了，“明明在人家的环境上可以运行的”。

Docker容器化带来的优点

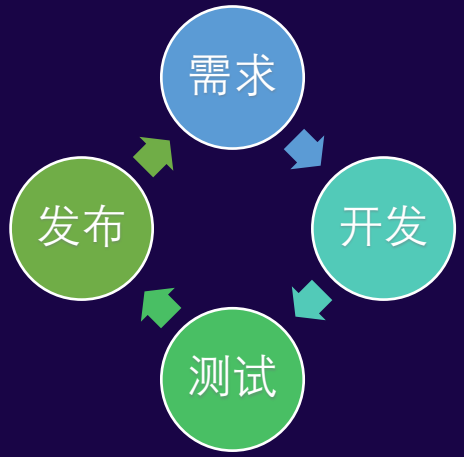
- 更高效的利用系统资源
- 更快的启动时间
- 一致的运行环境
- 更轻松的迁移
- 持续交付和部署
- 更轻松的维护和扩展

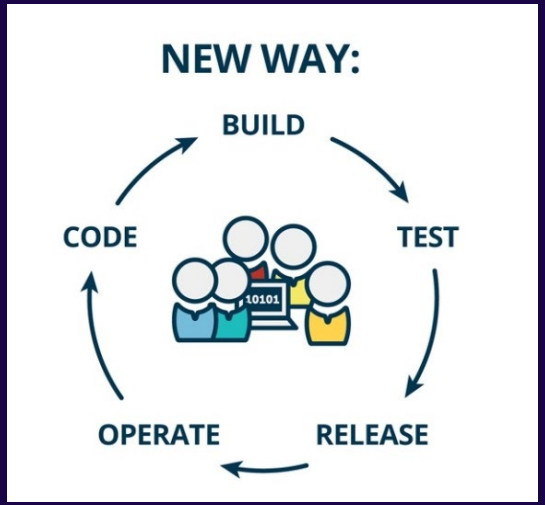
微服务四要素


云原生
微服务
应用间通过RESTful API通信
可以被独立的部署、更新、scale和重启
DevOps
自动化发布管道、CI工具
快速部署到生产环境
开发、运维协同合作
持续交付
频繁发布、快速交付、快速反馈、降低发布风险
容器化
微服务的最佳载体

一个项目/产品的迭代流程

1. 开发团队根据需求开发功能。
2. 开发完毕后进入测试。
3. 测试完成后交付验收
4. 验收通过部署到生产环境





- 持续集成、
- 持续发布功能，
- 帮助企业高质量、高效率交付业务。

Show Me The Code

Show Me The Code

- https://github.com/yoyomooc
- https://github.com/yoyomooc/dotnet_20Year_Blazor_DevOps

《云原生 .NET 6 实践 Blazor DevOps 容器化》

欢迎大家到 YoYoMooc.com 来学习.NET有关的知识

当前应用程序系统运行环境	
容器化运行	是
容器ID	230ee9cd68ff
CPU核心数量	16
cgroup内存使用情况	42848256
当前占用物理内存空间(MB)	92
.NET 版本	6.0.3
Runtime目录信息	/usr/share/dotnet/shared/Microsoft.NETCore.App/6.0.3/
服务器名称	230ee9cd68ff

学习小结

- 一个完整的Blazor 前后端分离的案例
- 一套完善的Docker与Docker-compose运维套件
- 可用的Github Action 工具包含镜像仓库与自动化的CI 工具

51Aspx
源码服务专家

感谢收看

大赛详情链接

https://club.51aspx.com/netscec/

扫码报名

微信客服