What's new in the .NET Platform

Immo Landwerth
Rich Lander

Product Managers for .NET

Agenda

Runtime

Libraries

SDK

We're showing a small subset of new features.

There are PLENTY.

What's new in .NET 9?

GC

Garbage Collector

Server GC memory reduction

Primarily adapts to application memory

Server GC now uses much less memory in environments with significant resources, for example >4 cores with >4GB RAM

.NET 9 will grow memory use as needed by the application or if application memory allocation rate increases.

.NET 8 is similar but treats machine/vm/container resources as a primary input for initial/ongoing memory use

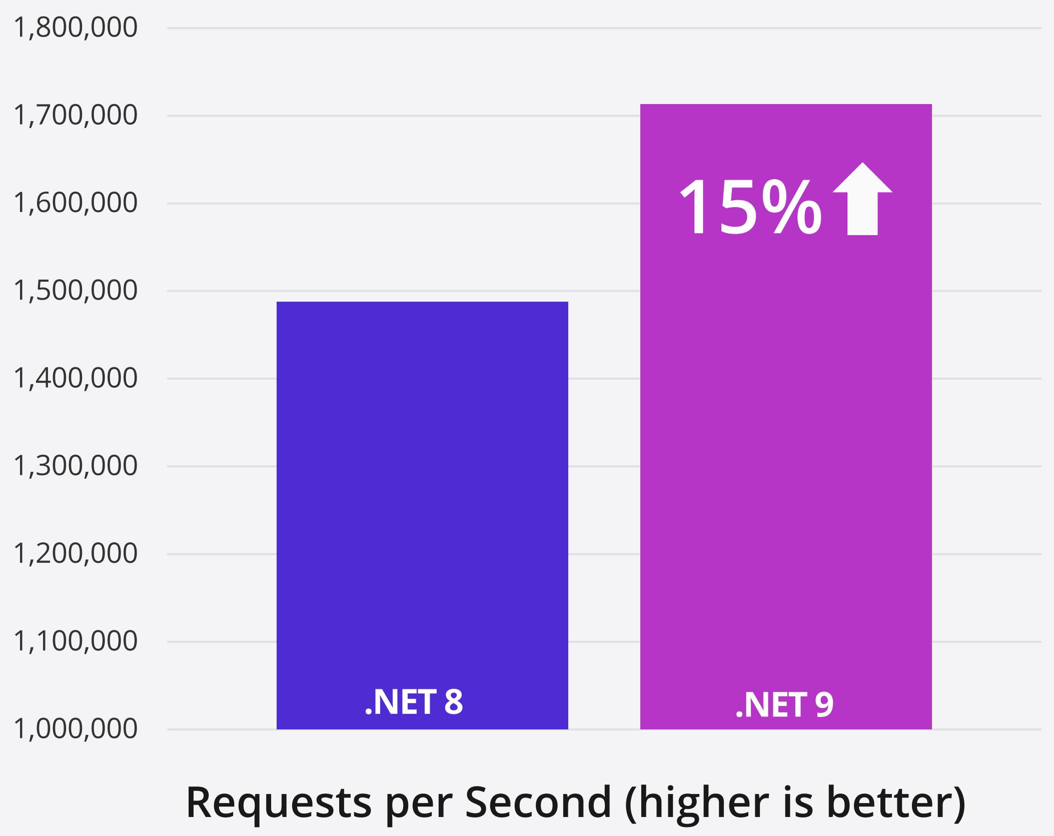
In practical terms, .NET 8 has a bias to starting off big and .NET 9 is the opposite.

.NET 9 Minimal API Performance

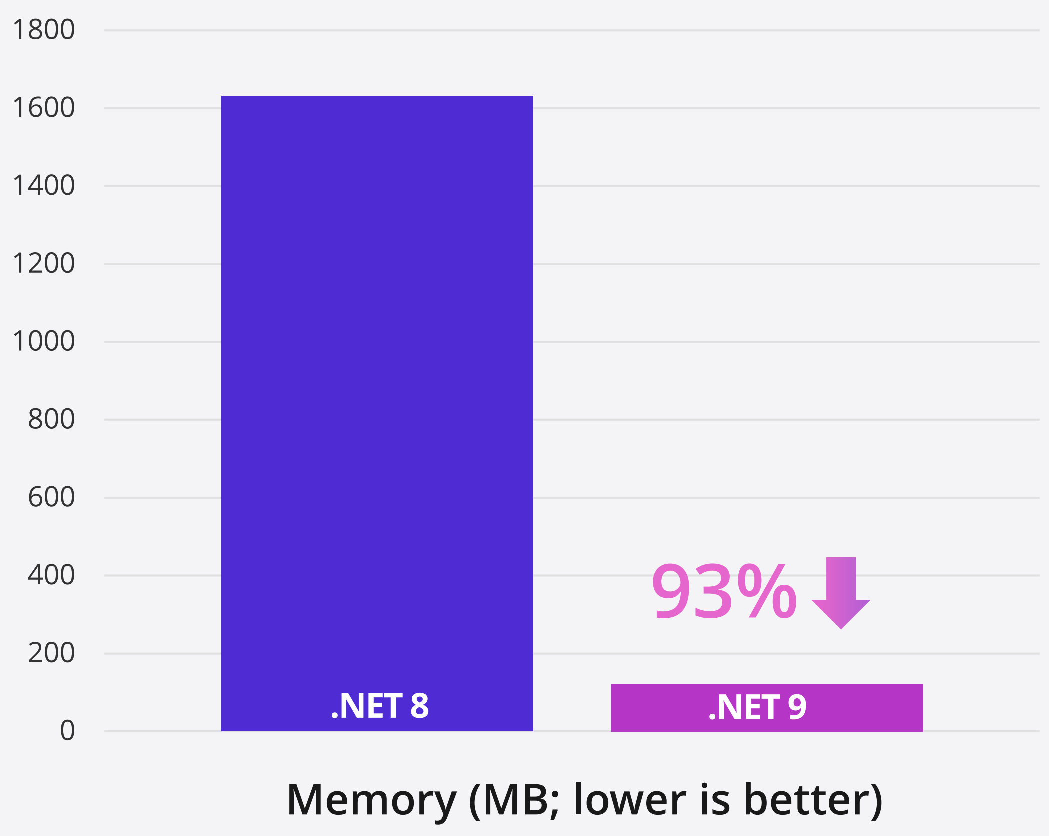
JSON Benchmarks

TechEmpower

Intel Gold 56 cores (logical) Linux
Source: aka.ms/aspnet/benchmarks



Requests per Second (higher is better)



Memory (MB; lower is better)

Server GC memory reduction

Trades a bit of throughput for lower memory

The new implementation is more adaptive than before, which comes at a modest throughput cost.

Depending on the application, the throughput cost may or may not be observable.

Server GC can be configured to use the legacy implementation, which can be useful for testing and remains supported in production.

RyuJIT

Code generation

See Andy Ayer’s talk on Thursday for a deep dive

Profile Guided Optimization (PGO)

It's the .NET re-compiler.

DPGO now has a fast path for common and uncommon casts.

```
static bool IsValid(IFoo foo)
{
    if (foo is MyFoo my)
    {
        // special logic for Myfoo
    }

    // regular logic
}
```

If foo typically is or is not `MyFoo`, the cast will be fast.

Just one of the PGO improvements this release.

A quick lesson on loops

I’ll only keep you for a while

This loop:

```
for (int i = 0; i < nums.Length; i++)
{
    sum += nums[i];
}
```

Gets lowered by the C# compiler to:

```
while (num2 < array2.Length)
{
    num += array2[num2];
    num2++;
}
```

Per sharplab.io.

The optimizations also apply to `foreach`, which is very similar.

Strength reduction (x64)

A nice addition to the JIT

The JIT needs to index into each array element.

```
for (int i = 0; i < nums.Length; i++)
{
    sum += nums[i];
}
```

.NET 8: *address-to-nums + i * 4*

.NET 9: *pointer-to-current-element += 4* *(arrays)*

.NET 9: address-to-nums + i; i + 4 *(spans)*

Removes a multiplication, per iteration.

We adopted post-indexed addressing on Arm64, which provides a similar outcome for arrays, as a CPU capability.

Strength reduction can also be applied in the body of a loop, like for `sum += i * 3;`

Induction variable (IV) widening (x64)

Please address the machine, politely

The JIT treats `i` as 8 bytes (64-bit register) on each iteration.

```
for (int i = 0; i < nums.Length; i++)
```

.NET 8: Widens the IV on each iteration, to 8 bytes.

.NET 9: Widens the IV once and then only reads the first 32-bits for body-of-the-loop accesses.

Widening is more expensive than narrowing.

This need is less relevant on Arm64.

Host

.NET app launcher

myapp.exe

./myapp

Control-flow enforcement technology

Enabled by default on Windows

CET provides hardware protect against Return-Oriented-Programming attacks, on Windows.

Enabled for the apphost, like myapp.exe

Not enabled for the dotnet launcher, like `dotnet run`

Has a modest performance cost.

Can be disabled. We recommend it.

Configure .NET install search behavior

Great for software suites

Multiple framework-dependent apps can share a private runtime

Previously, the only viable options were to deploy self-contained apps privately, each with a copy of the runtime, or rely on framework-dependent apps using a globally installed runtime.

You can also limit search behavior to a subset of options.

Core Libraries

New in .NET 9

Focus of .NET 9: Fundamentals

We added ~4,000 new types and members across the core libraries

- Performance
- Usability
- Security

Many more improvements!

- ReadOnlySet<T>
- Span-based lookups
- JSON Nullable & Required Enforcement
- params with span/IEnumerable<T>
- Guid V7
- JSON Schema Generation
- New Lock Type
- Hybrid Cache
- Metrics and Gauges
- More LINQ Methods
- More Hardware Intrinsics
- CPU Usage Info
- Compression Levels Control
- SearchValues<T>.IndexOfAny
- Reflection Emit Persistence
- Base64Url
- More span overloads
- Task When Each
- OrderedDictionary<TKey, TValue>

DEMO: Library Features in .NET 9

Immo Landwerth

BinaryFormatter is inherently insecure.

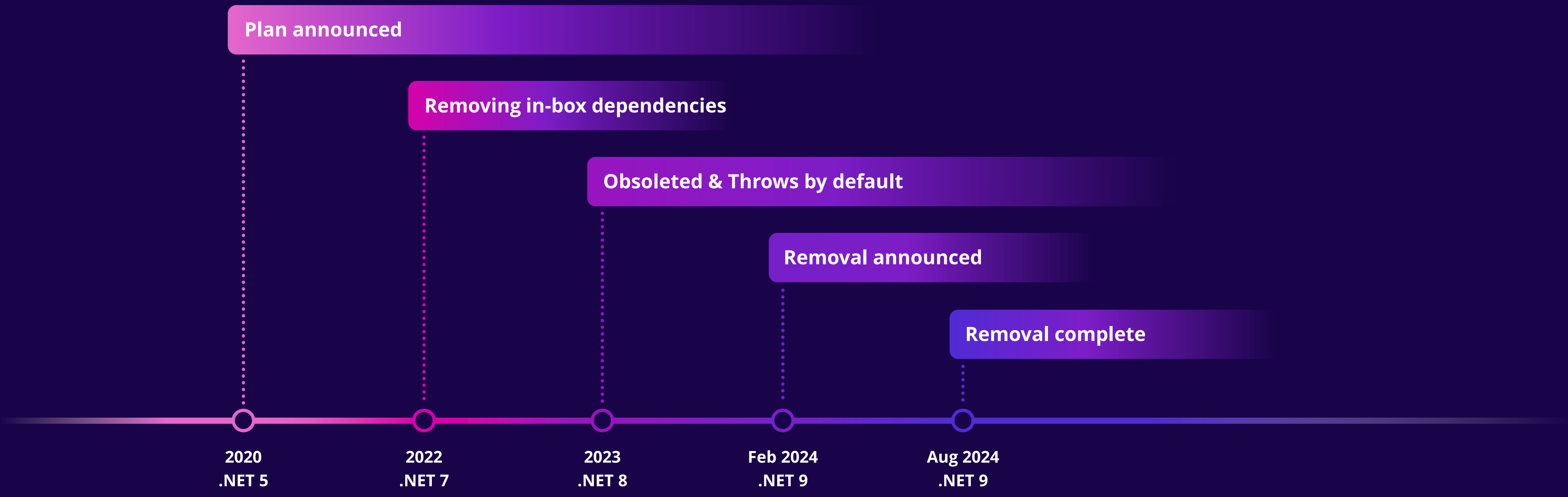
Stop using it.

State in .NET 9

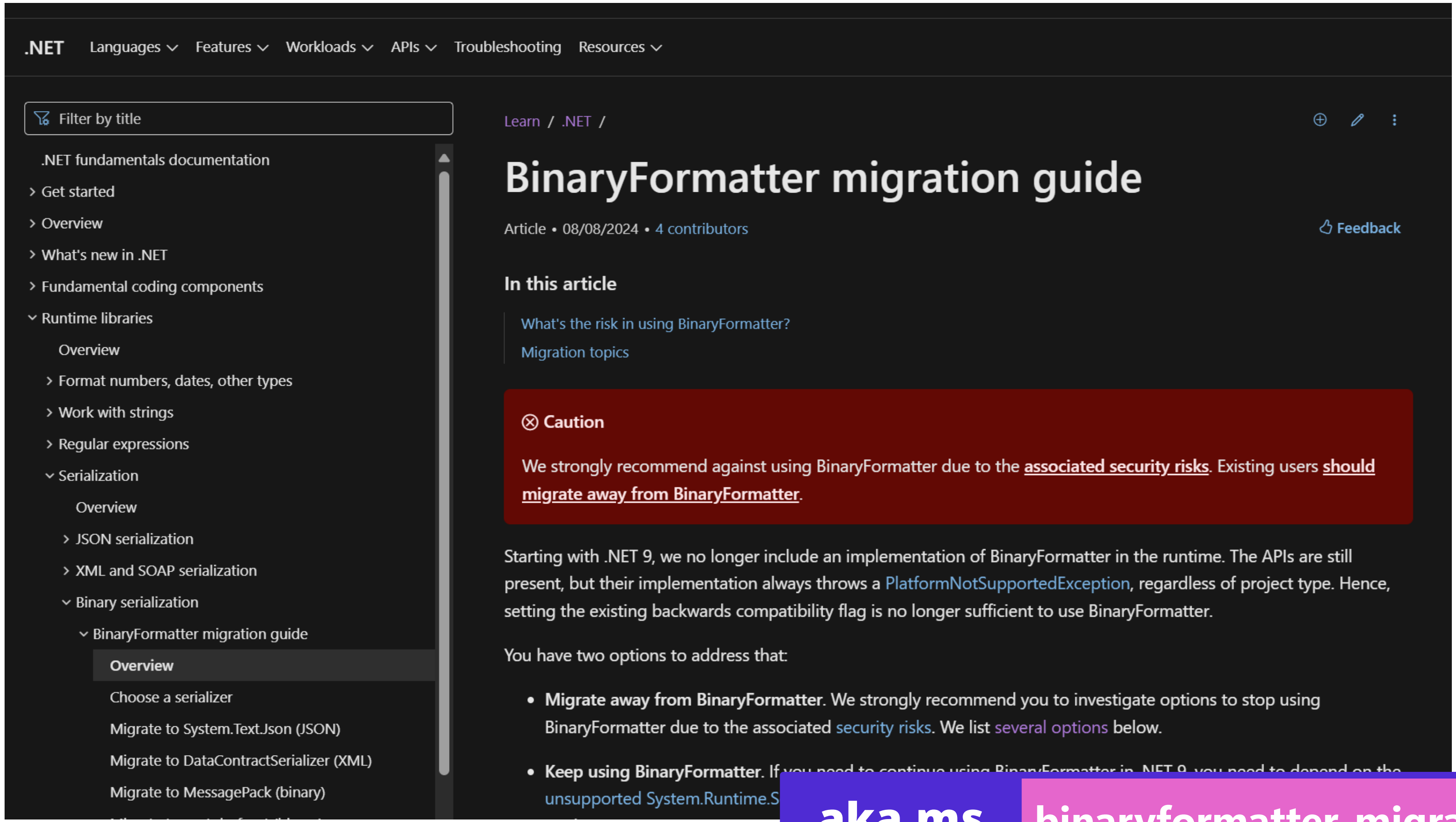
- **Marked as obsolete** (since .NET 8)

- The **runtime only includes a throwing implementation** (new in .NET 9)
- You can install an **unsupported compat package** to restore the functional (**but insecure**) implementation

Binary Formatter Deprecation Timeline



BinaryFormatter Migration Guide



BinaryFormatter migration guide

Article • 08/08/2024 • 4 contributors

In this article

What's the risk in using BinaryFormatter?
Migration topics

Caution

We strongly recommend against using BinaryFormatter due to the associated security risks. Existing users should migrate away from BinaryFormatter.

Starting with .NET 9, we no longer include an implementation of BinaryFormatter in the runtime. The APIs are still present, but their implementation always throws a PlatformNotSupportedException, regardless of project type. Hence, setting the existing backwards compatibility flag is no longer sufficient to use BinaryFormatter.

You have two options to address that:

- **Migrate away from BinaryFormatter.** We strongly recommend you to investigate options to stop using BinaryFormatter due to the associated security risks. We list several options below.
- **Keep using BinaryFormatter.** If you need to continue using BinaryFormatter in .NET 9, you need to depend on the unsupported System.Runtime.S

aka.ms binaryformatter-migration-guide

SDK

New in .NET 9

Package vulnerability auditing

Just got “transitive”

Package auditing now considers all dependencies, both direct and indirect (AKA “transitive”).

You will likely start seeing more vulnerabilities with `dotnet restore` and commands that call it.

```
/app/Tests/Tests.csproj : warning NU1903: Package
'System.Net.Http' 4.3.0 has a known high severity
vulnerability, https://github.com/advisories/GHSA-
7jgj-8wvc-jh57
```

Some System.* packages are false positives. We’re working on improving that. For now, please resolve all vulnerability warnings.

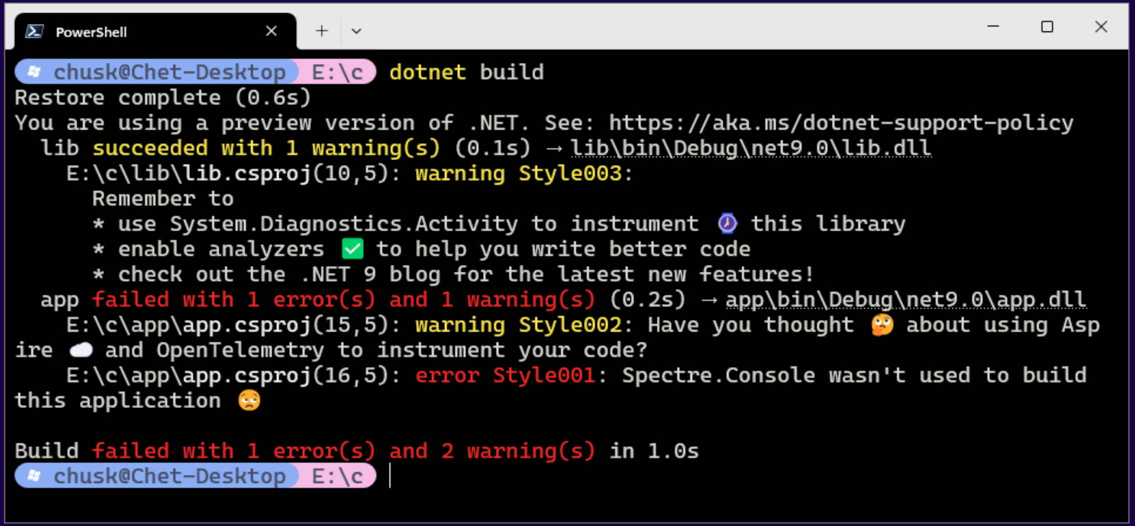
Terminal Logger

Pretty colors; Pretty summaries

Focus your attention on the important things in your build – your projects, their primary outputs, and their diagnostics.

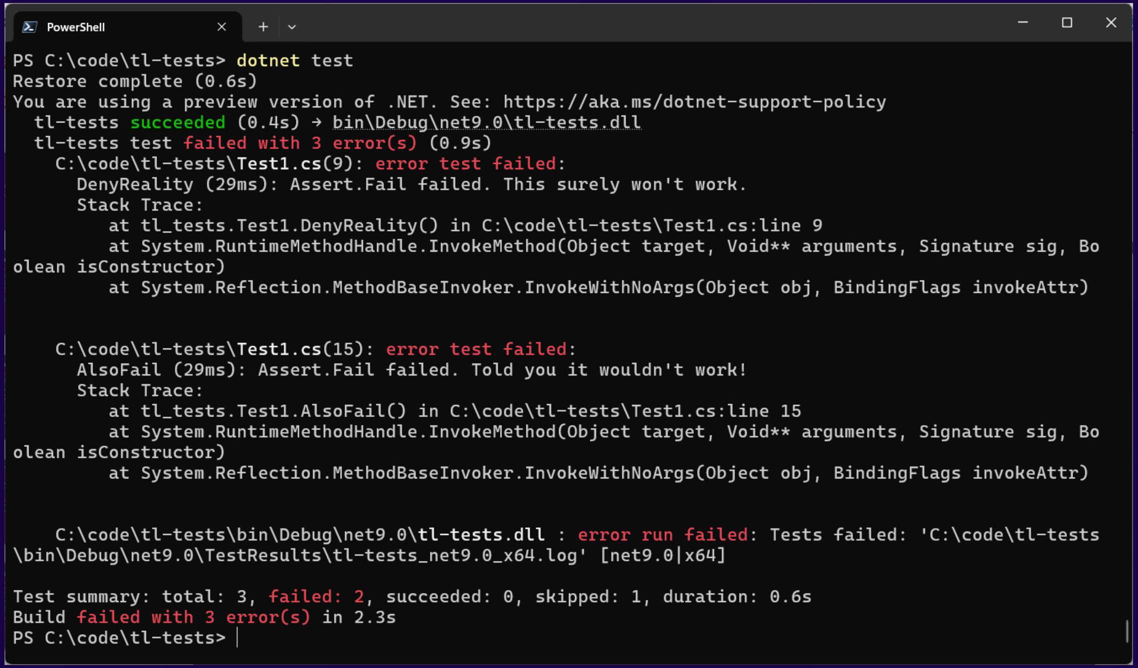
At-a-glance info about what's being built *right now* and how long it's taking.

Supports hyperlinks, color, and multi-line diagnostics.



```
 chusk@Chet-Desktop  E:\c  dotnet build
Restore complete (0.6s)
You are using a preview version of .NET. See: https://aka.ms/dotnet-support-policy
  lib succeeded with 1 warning(s) (0.1s) → lib\bin\Debug\net9.0\lib.dll
    E:\c\lib\lib.csproj(10,5): warning Style003:
      Remember to
      * use System.Diagnostics.Activity to instrument 🕓 this library
      * enable analyzers ✅ to help you write better code
      * check out the .NET 9 blog for the latest new features!
  app failed with 1 error(s) and 1 warning(s) (0.2s) → app\bin\Debug\net9.0\app.dll
    E:\c\app\app.csproj(15,5): warning Style002: Have you thought 🤔 about using Asp
ire ☁️ and OpenTelemetry to instrument your code?
    E:\c\app\app.csproj(16,5): error Style001: Spectre.Console wasn't used to build
this application 😥

Build failed with 1 error(s) and 2 warning(s) in 1.0s
 chusk@Chet-Desktop  E:\c 
```



```
PS C:\code\tl-tests> dotnet test
Restore complete (0.6s)
You are using a preview version of .NET. See: https://aka.ms/dotnet-support-policy
  tl-tests succeeded (0.4s) → bin\Debug\net9.0\tl-tests.dll
  tl-tests test failed with 3 error(s) (0.9s)
    C:\code\tl-tests\Test1.cs(9): error test failed:
      DenyReality (29ms): Assert.Fail failed. This surely won't work.
      Stack Trace:
         at tl_tests.Test1.DenyReality() in C:\code\tl-tests\Test1.cs:line 9
         at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Bo
olean isConstructor)
         at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)


    C:\code\tl-tests\Test1.cs(15): error test failed:
      AlsoFail (29ms): Assert.Fail failed. Told you it wouldn't work!
      Stack Trace:
         at tl_tests.Test1.AlsoFail() in C:\code\tl-tests\Test1.cs:line 15
         at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Bo
olean isConstructor)
         at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)


    C:\code\tl-tests\bin\Debug\net9.0\tl-tests.dll : error run failed: Tests failed: 'C:\code\tl-tests
\bin\Debug\net9.0\TestResults\tl-tests_net9.0_x64.log' [net9.0|x64]

Test summary: total: 3, failed: 2, succeeded: 0, skipped: 1, duration: 0.6s
Build failed with 3 error(s) in 2.3s
PS C:\code\tl-tests>
```

.NET Tool Roll-forward

Configure on install

Some tools are configured to run on only one .NET version.

This is, in fact, the default.

You can override this setting on install, with `--allow-roll-forward`

For example:

```
$ dotnet tool install -g --allow-roll-forward nuget.packagesourcemapper
```

If you forget to do this at install-time, you can also do it when you run the app (if it is installed as a local tool):

```
$ dotnet tool run –allow-roll-forward packagesourcemapper
```

Publish to insecure registries

Enabling local workflows

`dotnet publish` can push images to a container registry

Some users push to a local http-endpoint registry, often running in a container

.NET 8: Publish only supports https registries

.NET 9: Publish supports https and http registries.

Example:
```
$ docker run -d -p 5000:5000 --restart always --name registry registry:2
$ set DOTNET_CONTAINER_INSECURE_REGISTRIES=localhost:5000
$ dotnet new web
$ dotnet publish -t:PublishContainer -p:ContainerRepository=testapp -p:ContainerRegistry=localhost:5000
$ docker run -it -d -p 8080:8080 testapp
$ curl http://localhost:8080
Hello World!
```

If you've configured insecure registries in Docker or Podman, the SDK can make use of that configuration.

Get .NET 9

Download .NET 9
aka.ms/get-dotnet-9

Thank you