Building Configurable Applications Using .NET Generic Host

Understanding Configuration in .NET
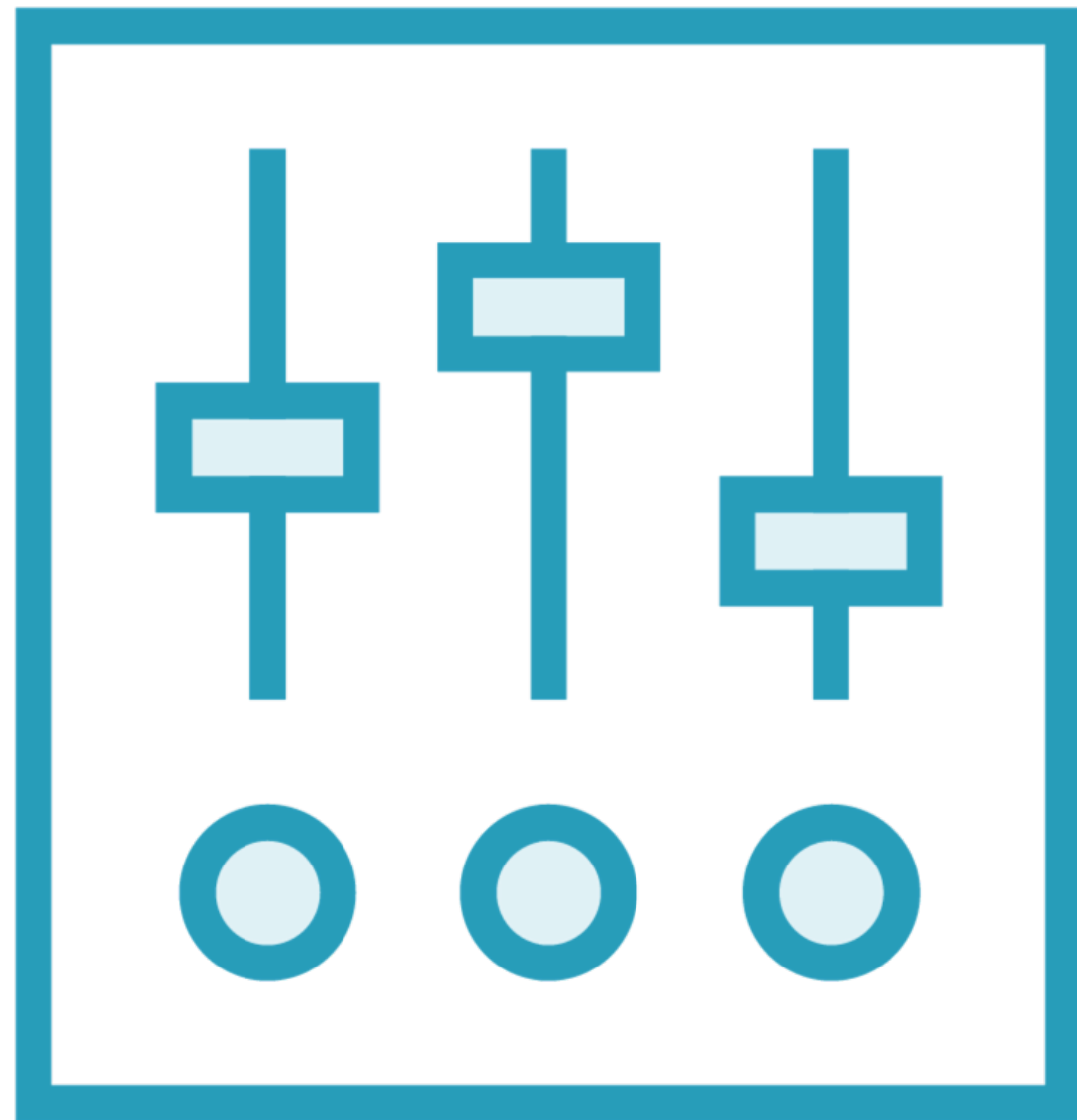
Matthew Tester
Technology Consultant

@matttester   matthewtester.com

Version Check

.NET 5.0 onwards
.NET Core 2.1 onwards

A Configurable Application

Portable
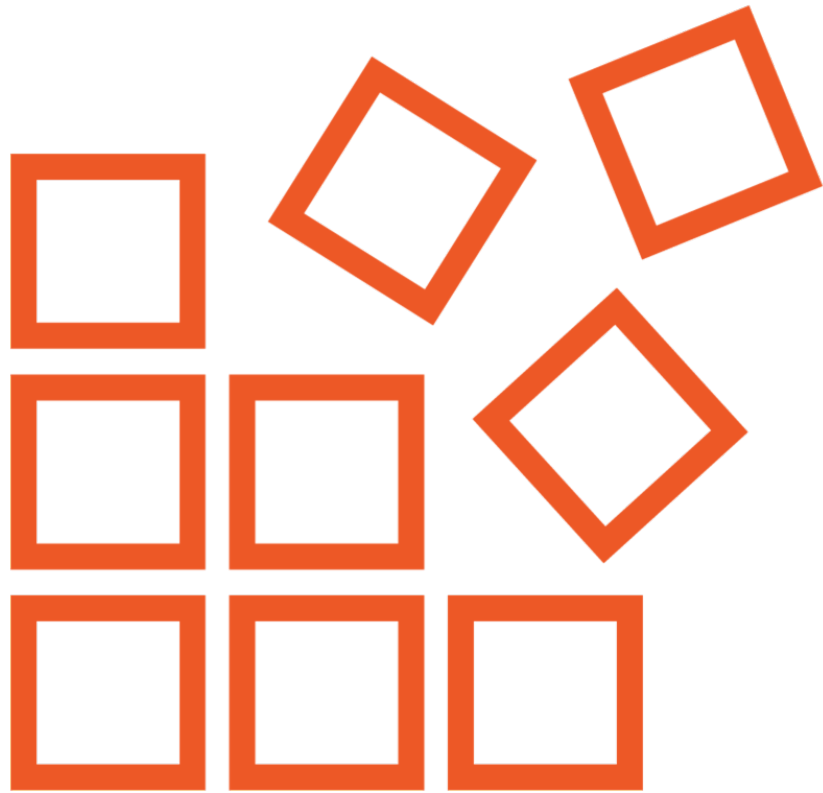- Runs in different environments without rebuilding

Flexible
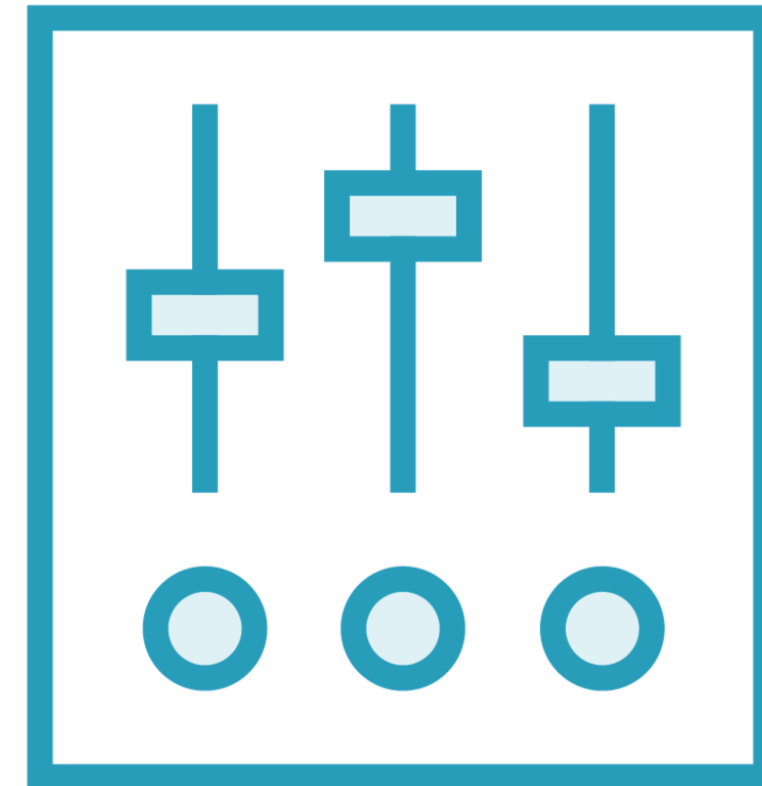- Can respond to changing business requirements

Maintainable
- Doesn't require full development cycle for a simple change
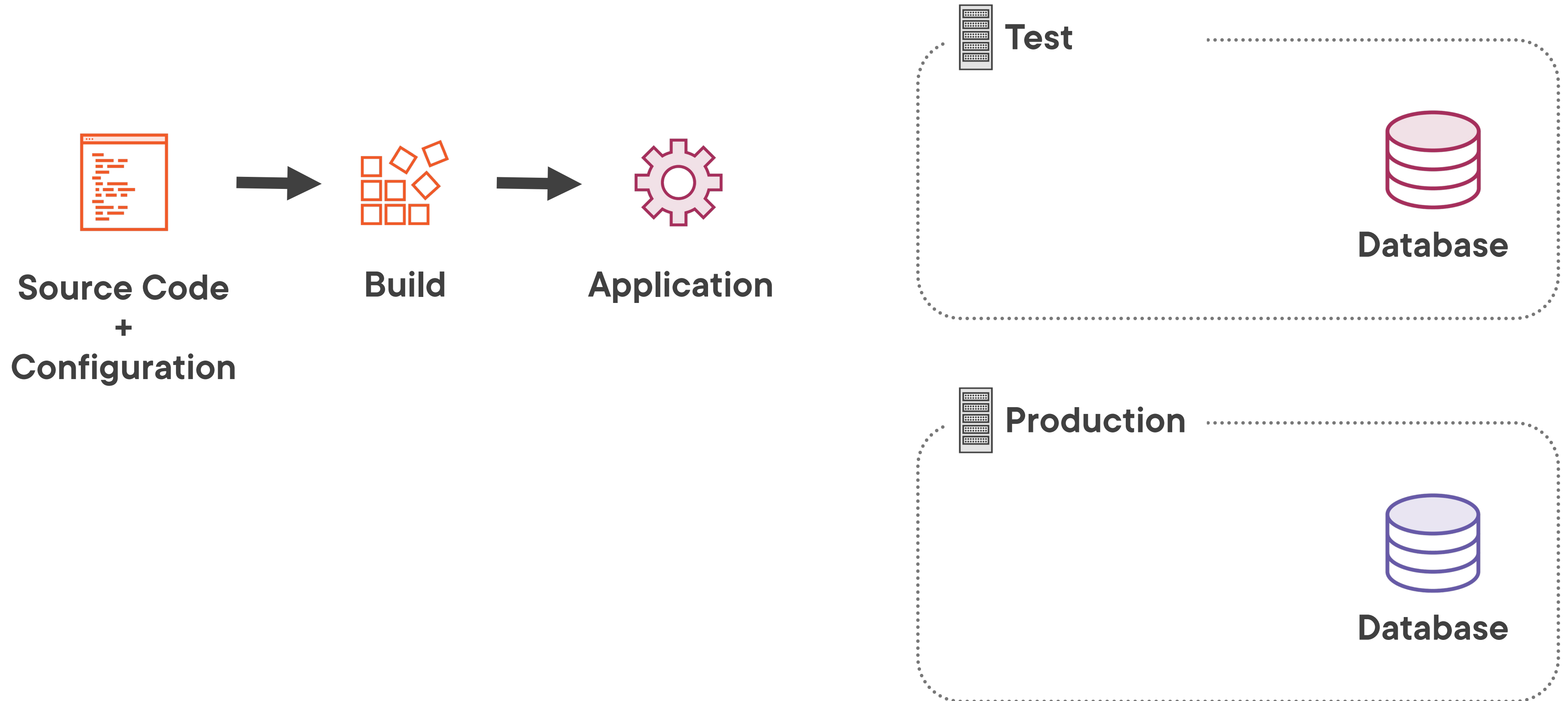
Types of Configuration

Compile-time
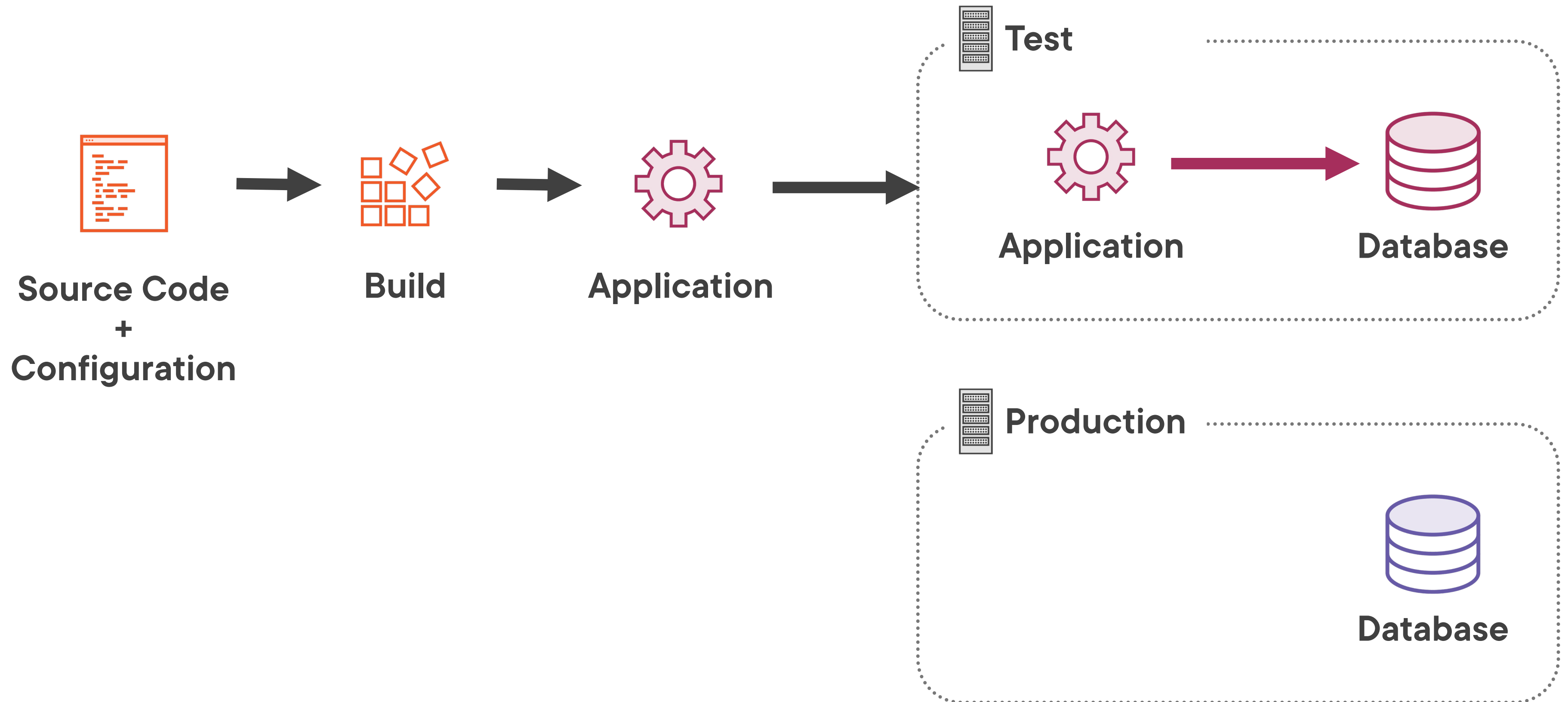Configuration values built into the application

Runtime
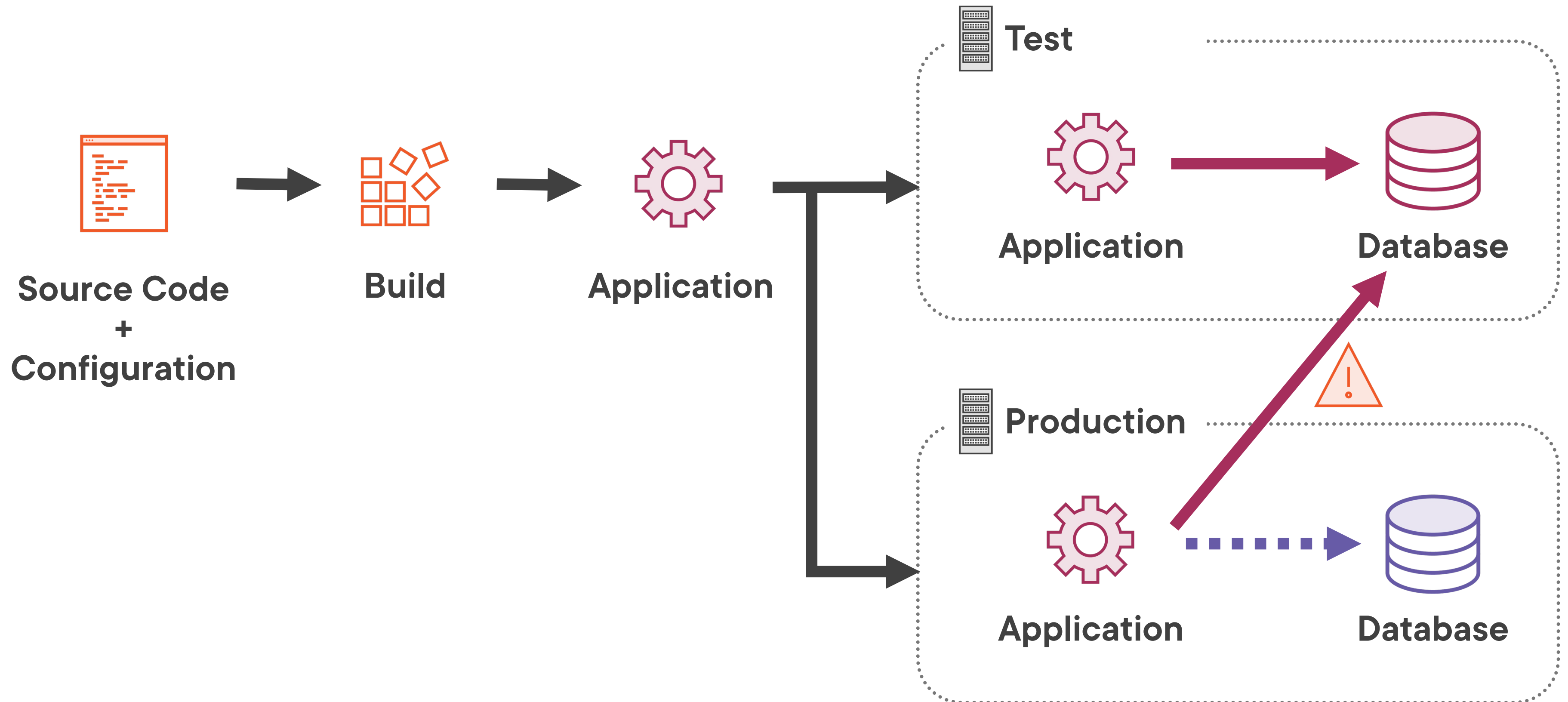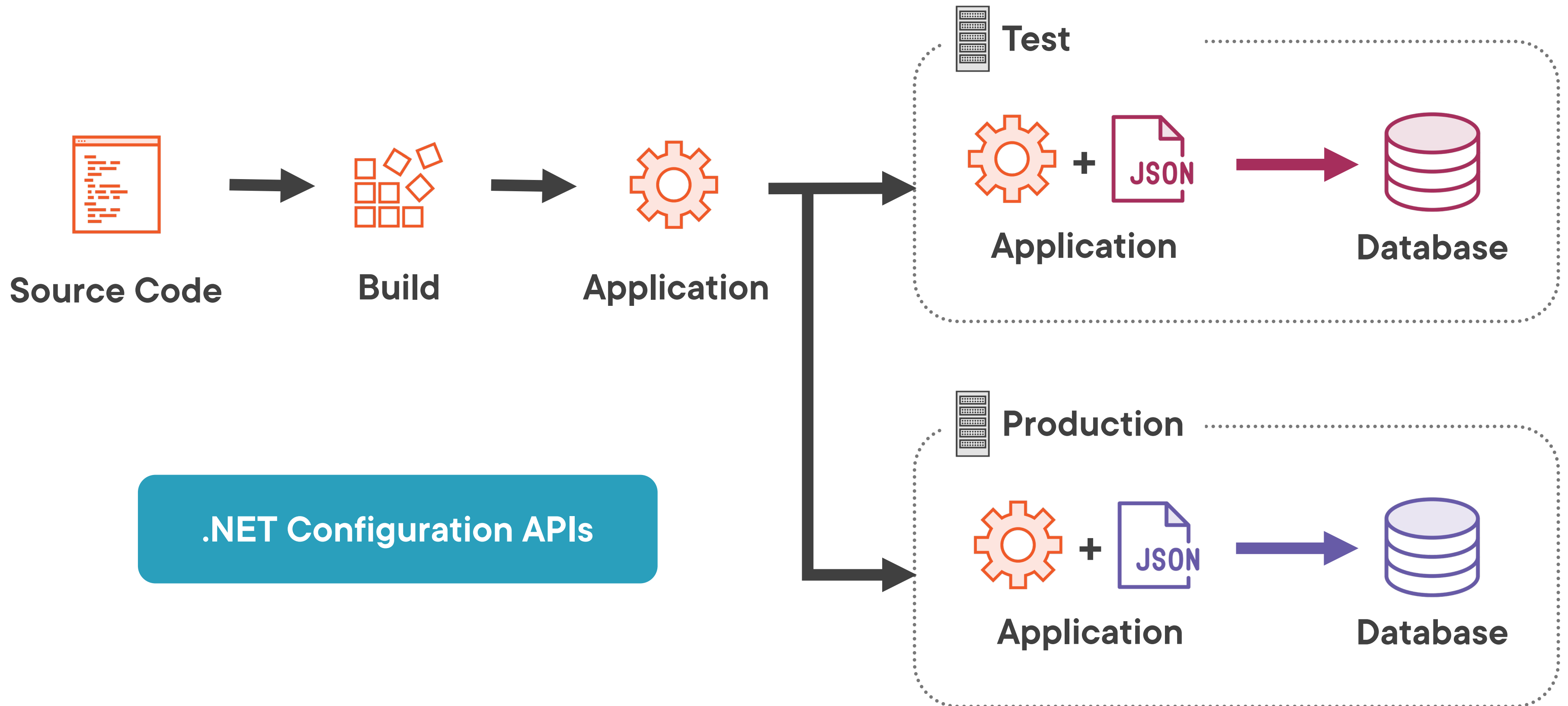Application loads configuration values at runtime

Compile-time Configuration

Source Code + Configuration → **Build** → **Application**

Test

Database

Production

Database

Compile-time Configuration

Source Code + Configuration → **Build** → **Application** →

Test
Application → **Database**

Production
Database

Compile-time Configuration

Runtime Configuration

Sells outdoor equipment

Online store

Competitive market

Streamline the cataloguing process

New product images need to be:
- Resized
- Optimized
- Watermarked

Carved Rock Fitness: Image Process

Raw Photos → **Web Designer** → **Manually Adjust** → **Product Images**

Takes a lot of time

Carved Rock Fitness: Image Process

Carved Rock Fitness: Image Process

Raw Photos

JPG TIFF

Image Service

Product Images

Runs without interaction

.NET Generic Host
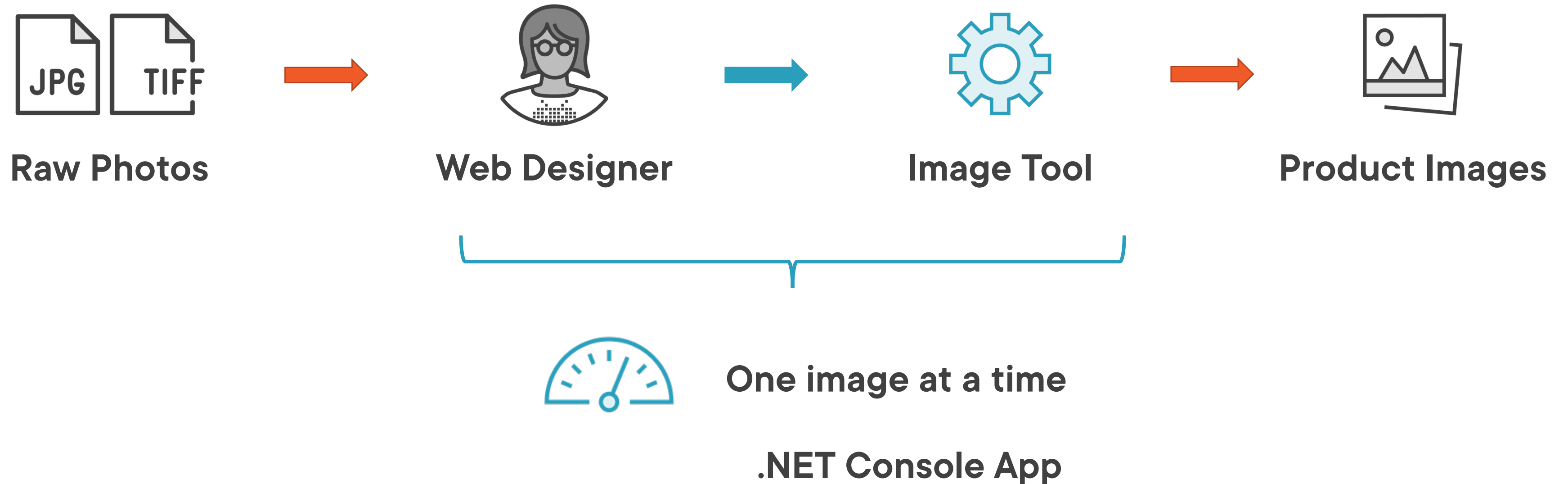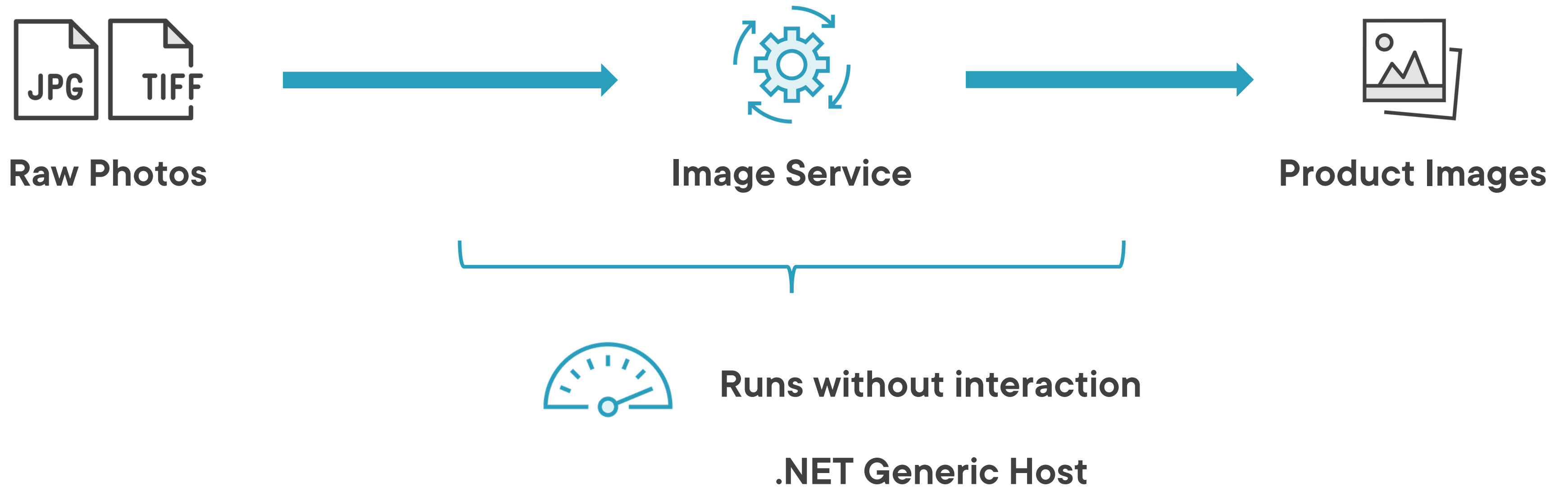
Possible Configuration
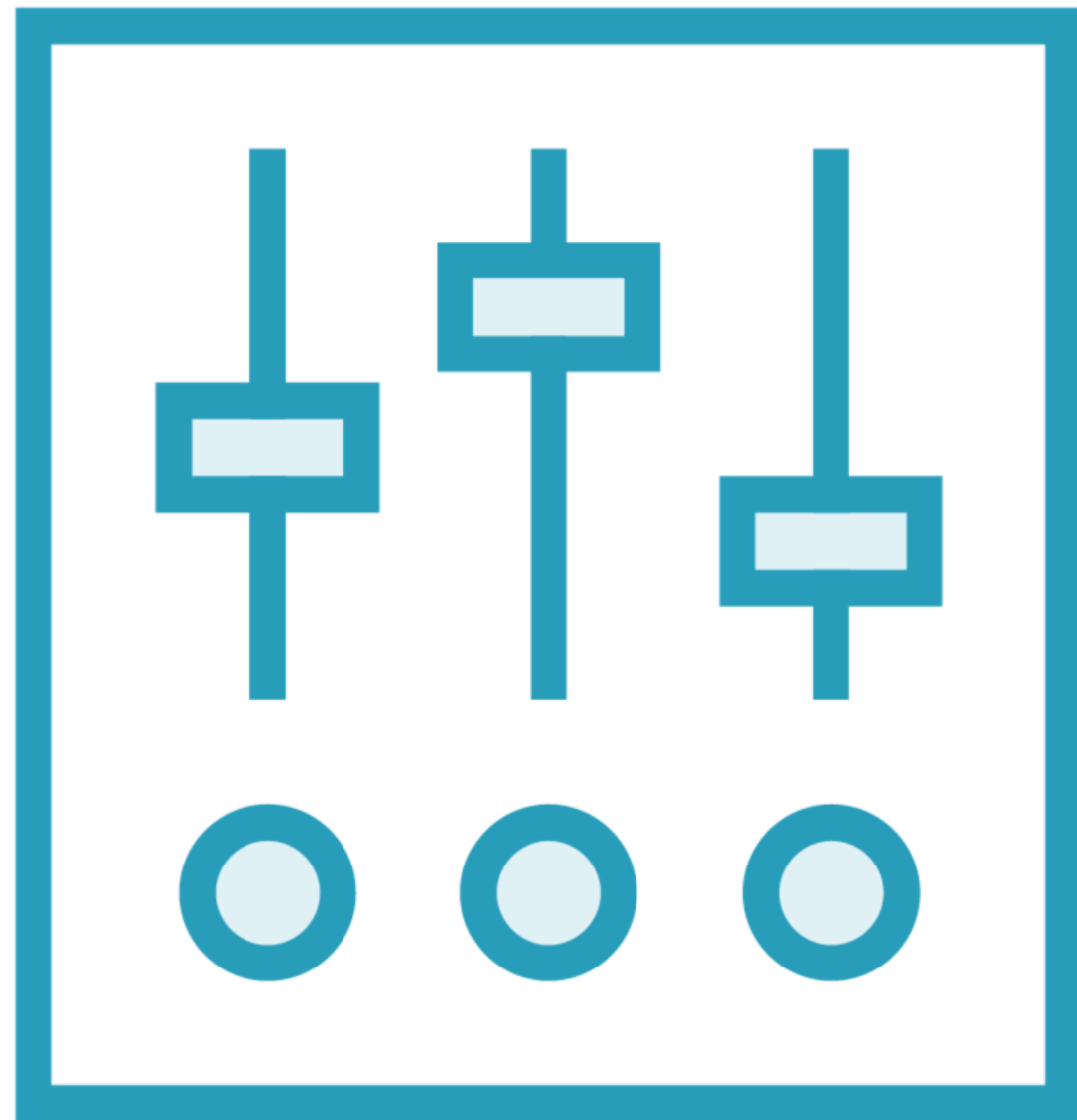
Images dimensions
- Thumbnail, medium and large

Compression level

Watermark text

Output image filename prefixes
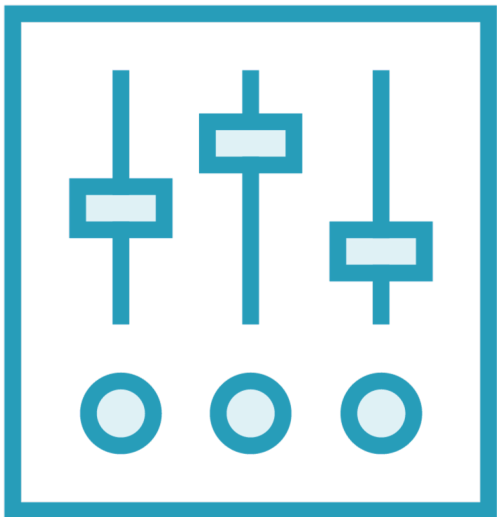- E.g. thumbnail-backpack.jpg

Configuration in .NET

Configuration

Key-value pairs

Key	Value
thumbnailWidth	52
compressionLevel	0.2
...	...

Configuration in .NET
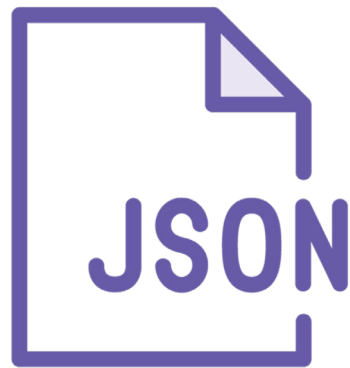
Source
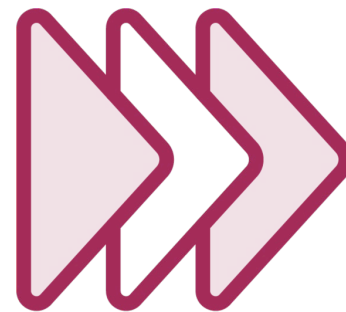
Configuration Provider
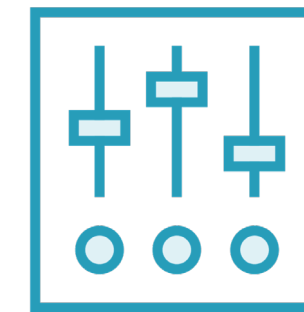
Configuration

```json
{
  "thumbnailWidth": 48,
  "compressionLevel": 0.7
}
```

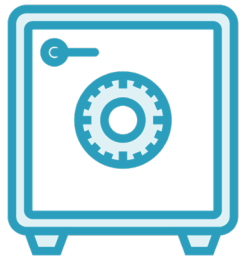| Key	Value
thumbnailWidth	48
compressionLevel	0.7

.NET Configuration Providers

Files e.g. JSON, XML, INI

Command-line parameters

Environment variables
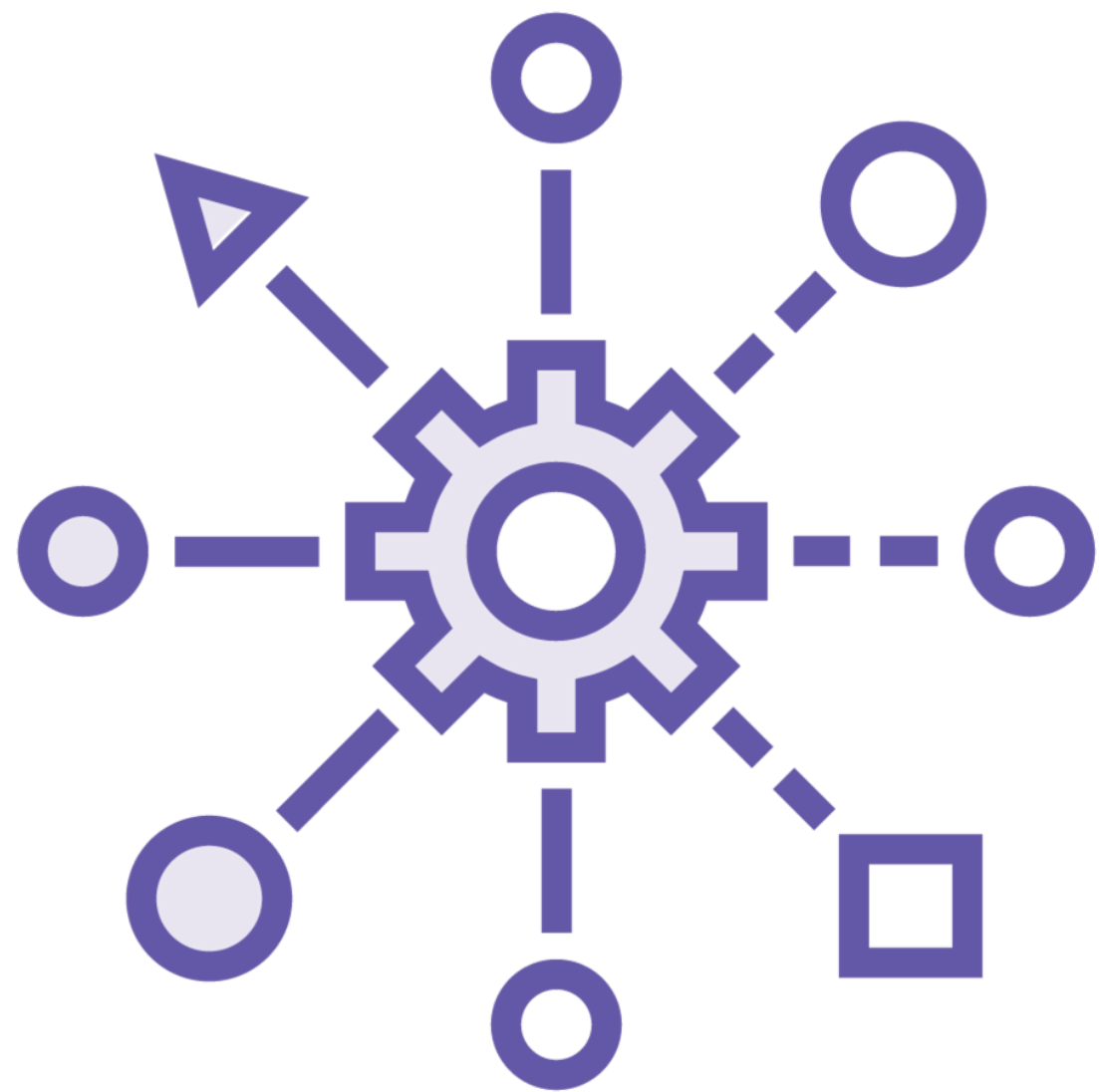
Azure Key Vault

… And more, with option to create a custom provider.

Modular Design

Follows .NET modular design using NuGet

- Only reference what you need

Configuration API

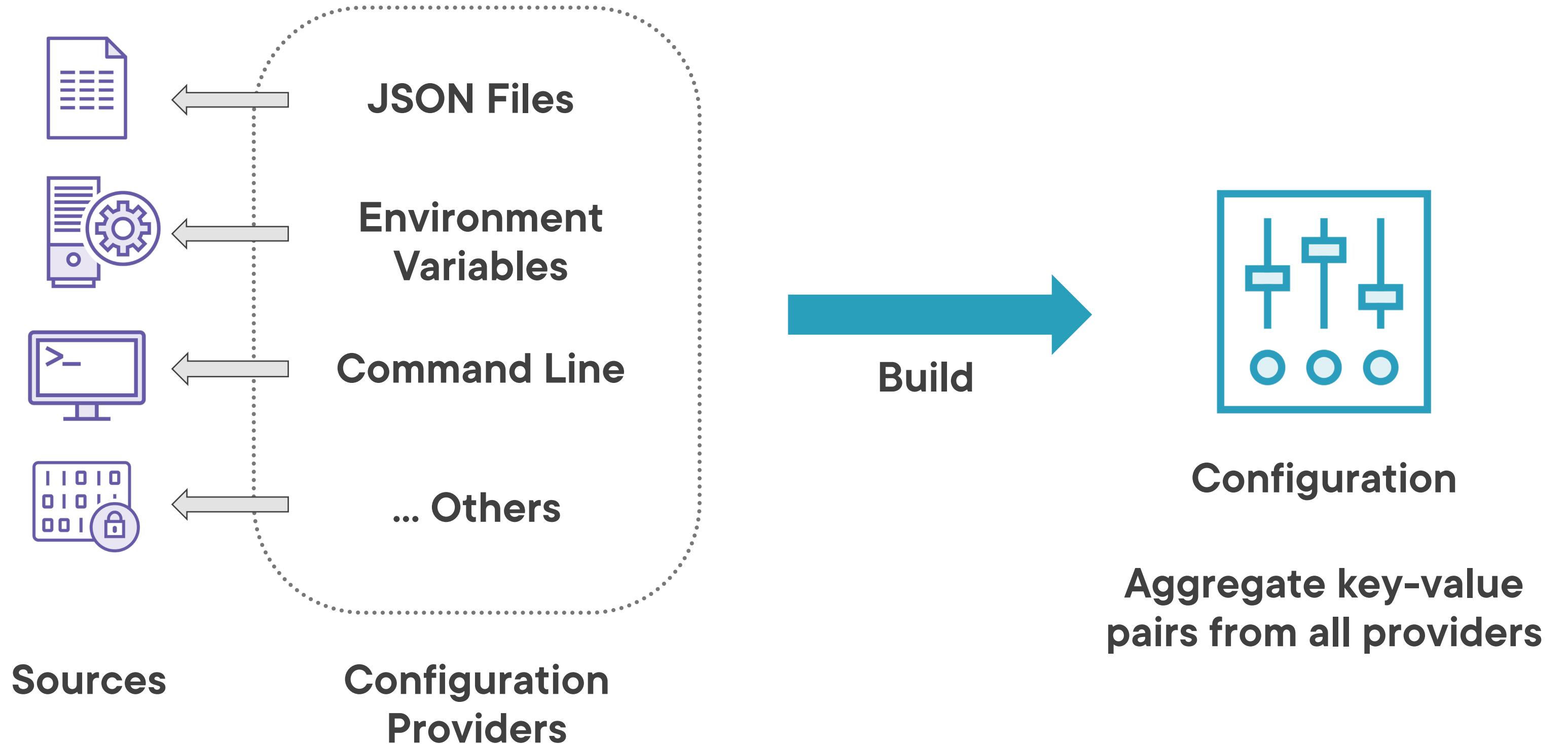- Microsoft.Extensions.Configuration

Configuration providers are separate NuGet packages, e.g.

- Microsoft.Extensions.Configuration.Json

Not limited to using just one provider

Building Configuration

Sources

Configuration Providers

- JSON Files
- Environment Variables
- Command Line
- ... Others

Build

Configuration

Aggregate key-value pairs from all providers

Building Configuration

Sources

Configuration Providers

JSON Files

Environment Variables

Command Line

... Others

```
thumbnailWidth = 512

thumbnailWidth = 48
```

Configuration
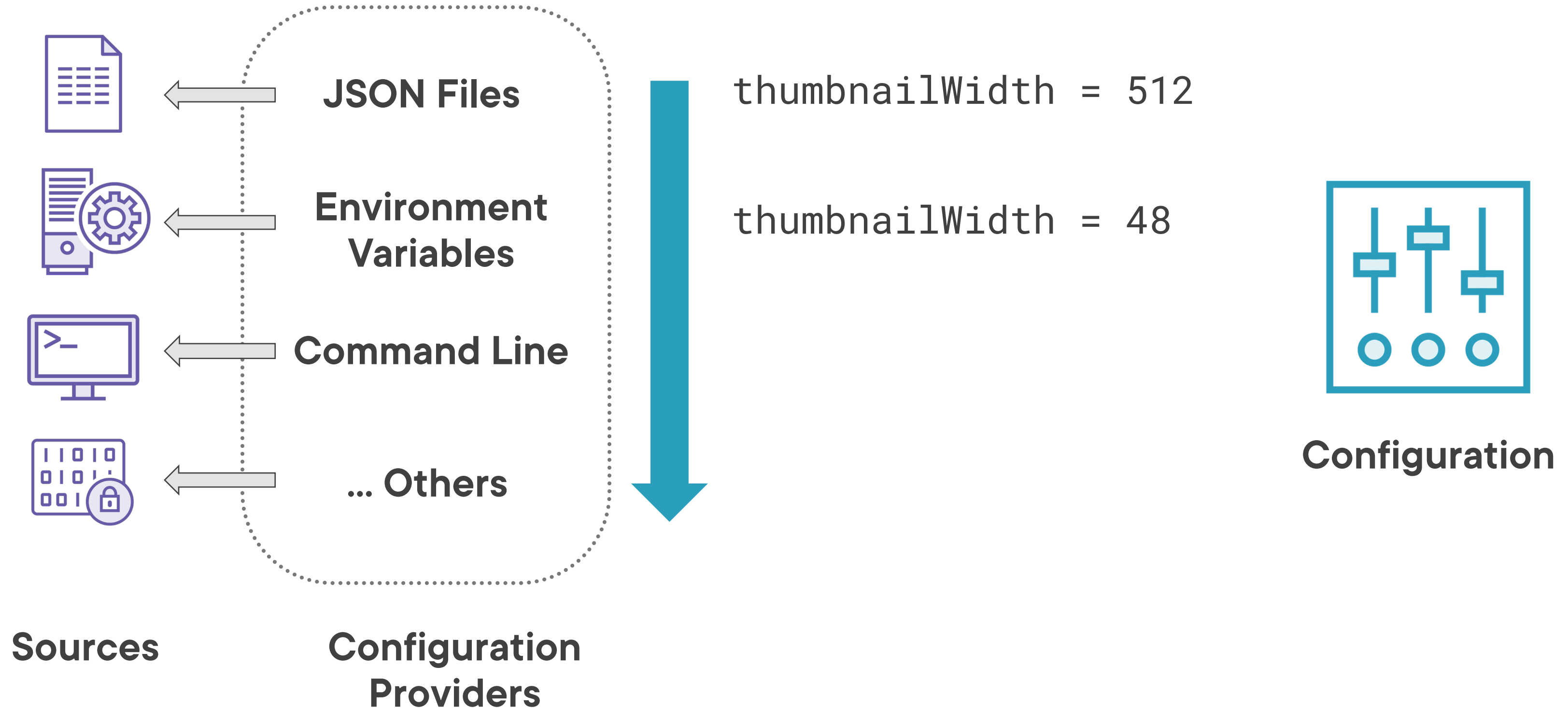
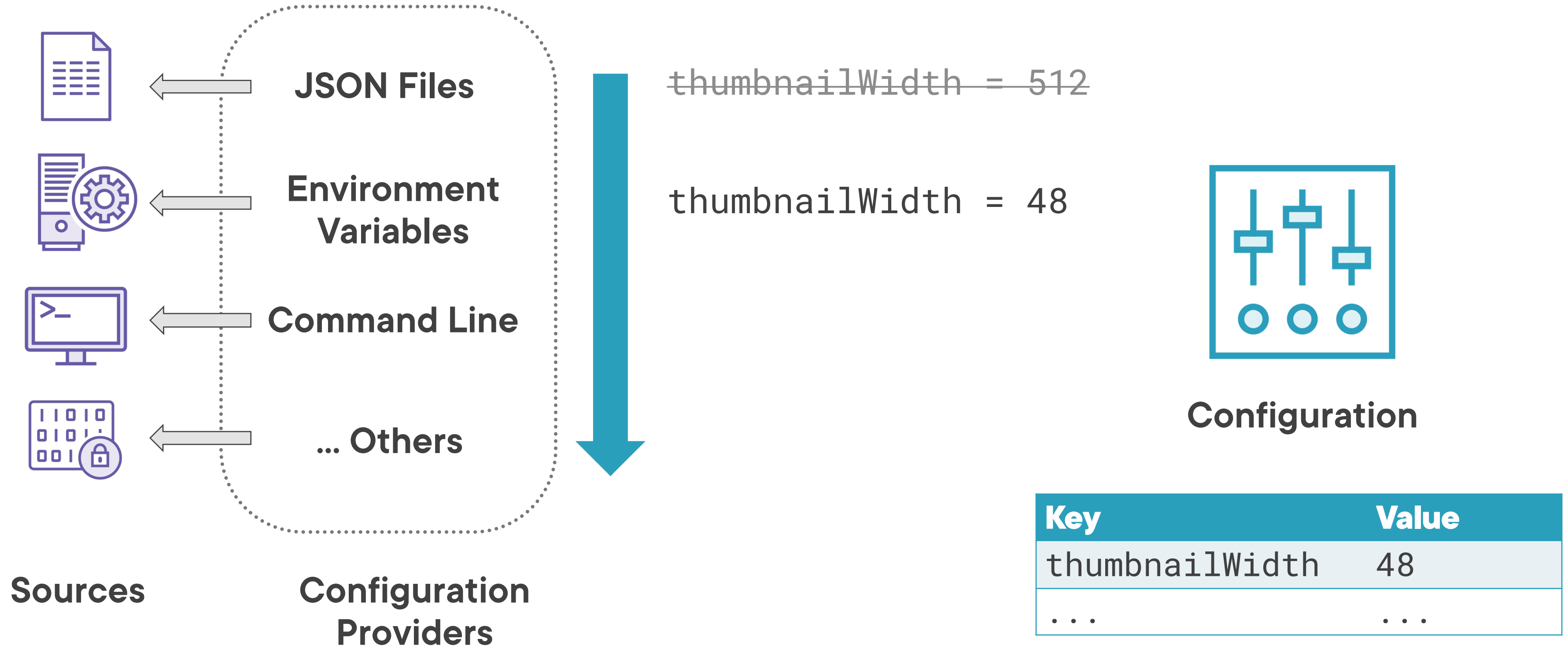# Building Configuration

Summary

Configurable applications are flexible, portable and maintainable

Runtime vs. compile-time configuration

.NET Configuration
- **Providers load from a configuration source**
- **Builds dictionary of key-value pairs**
- **Multiple providers work in order, overwriting duplicate keys.**

Modular design using NuGet packages

Up Next:
Exploring the Configuration API