Microsoft

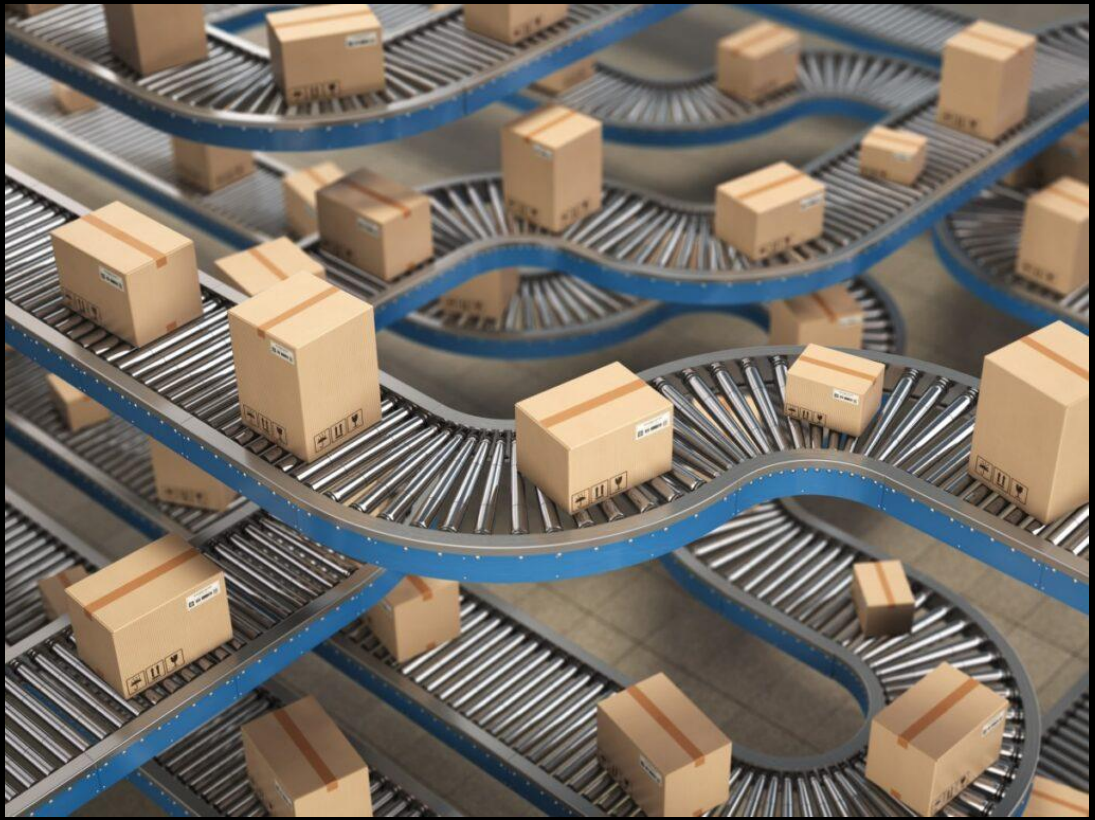
Integrate Your Changes

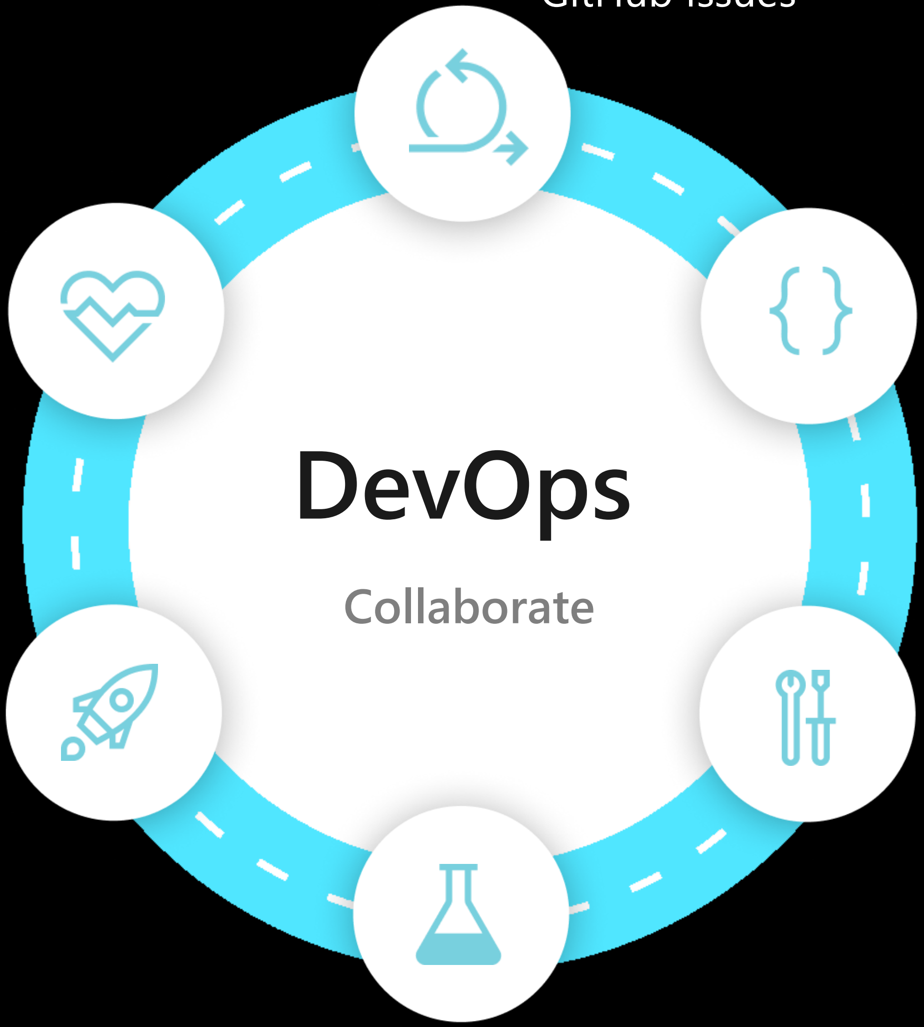
Triggering your pipeline

Eric Laan (ericlaan@microsoft.com)
Digital and Application Innovation specialist

Remco Brosky (remcobrosky@microsoft.com)
Cloud Solution Architect

Day 2, Developer Experience Days, Apr 2023





Plan
Azure Boards
GitHub Issues

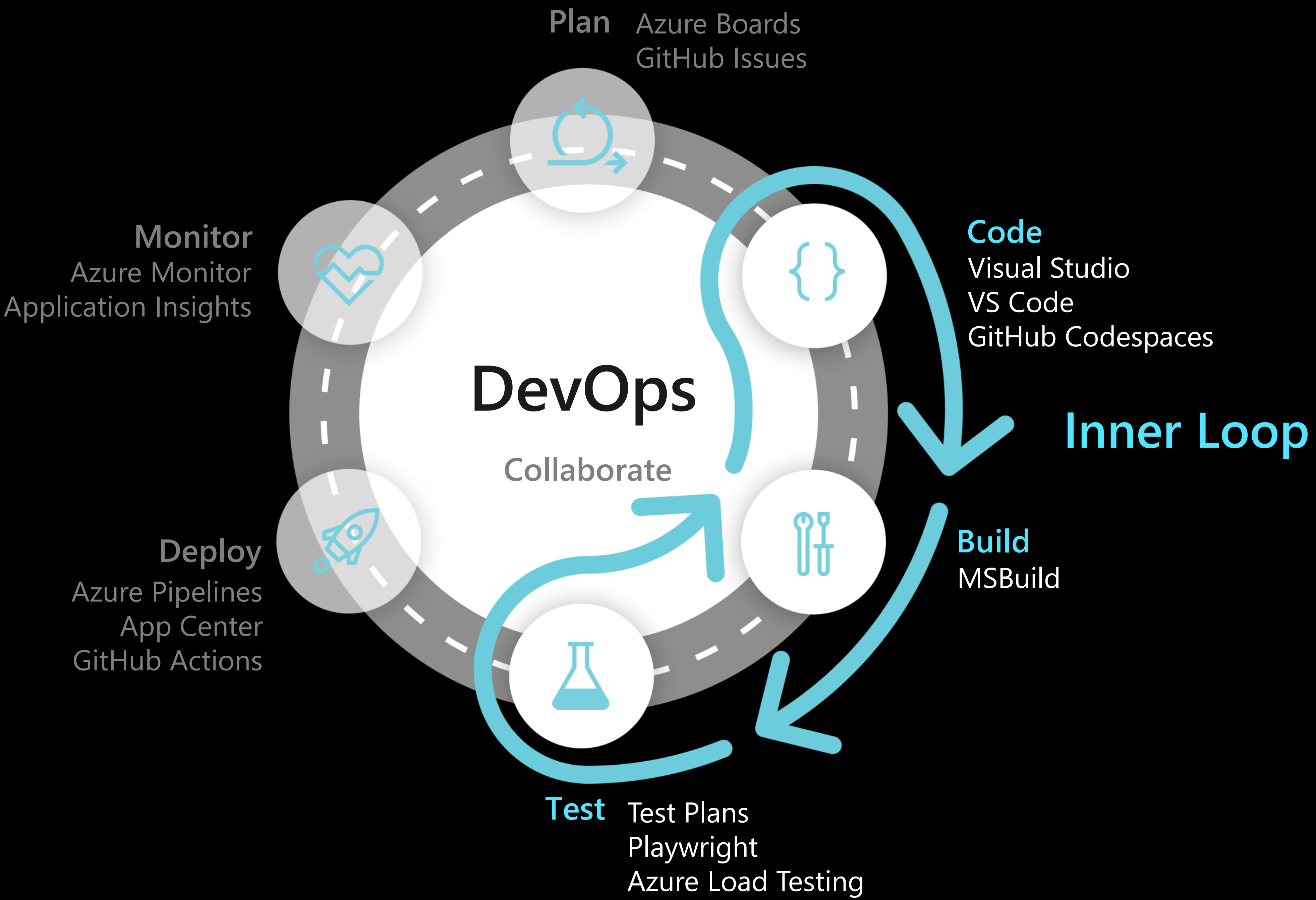
Code
Visual Studio
VS Code
GitHub Codespaces

Build
MSBuild

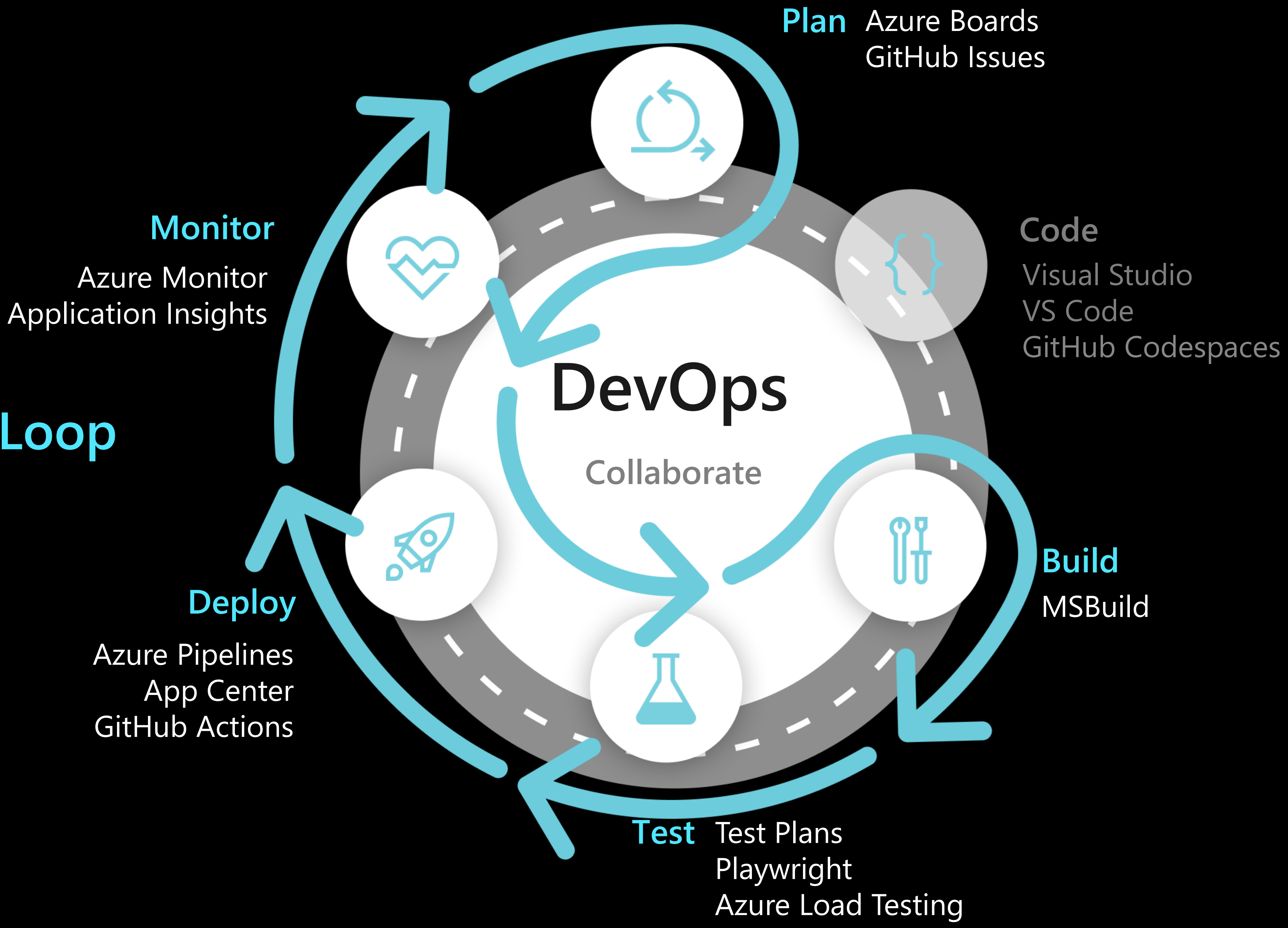
Test
Test Plans
Playwright
Azure Load Testing

Deploy
Azure Pipelines
App Center
GitHub Actions

Monitor
Azure Monitor
Application Insights

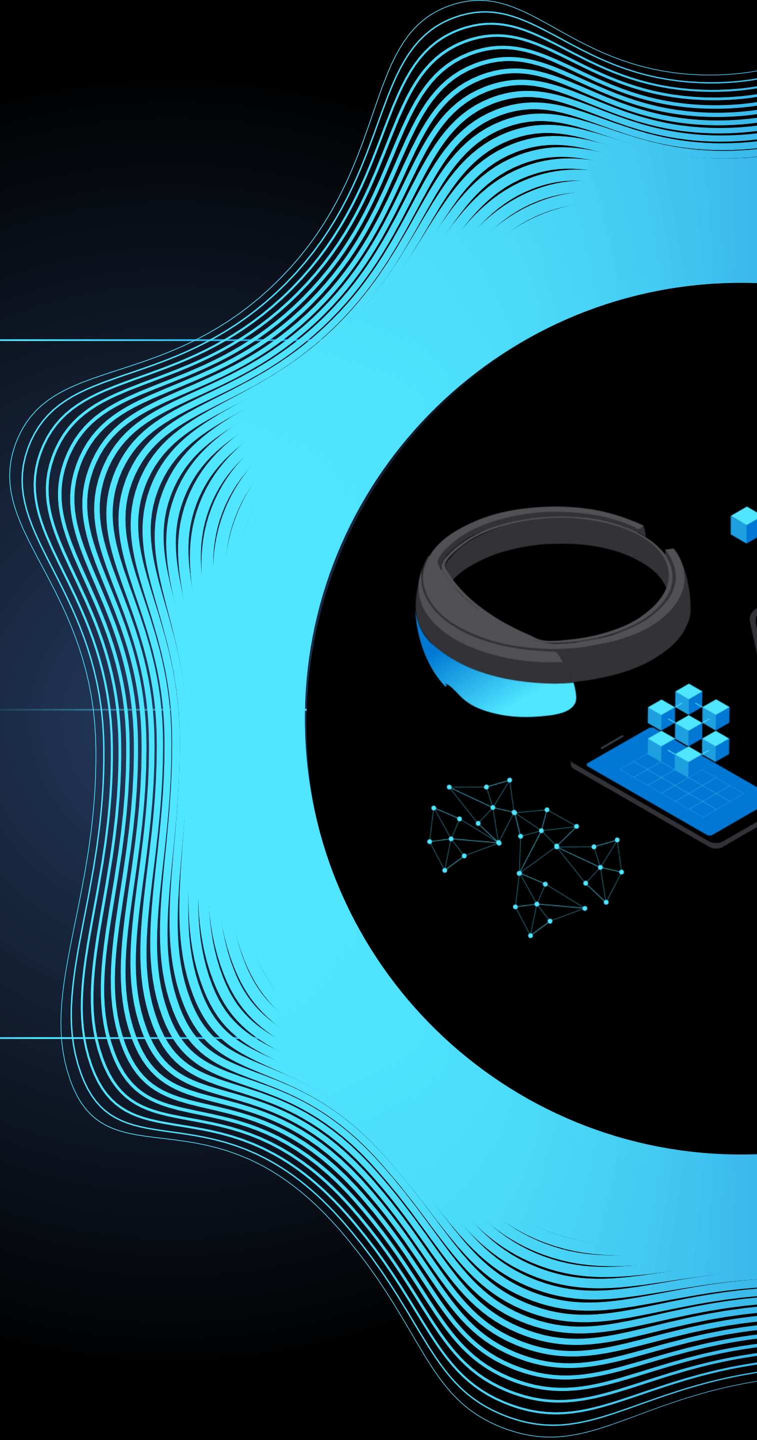

Plan
Azure Boards
GitHub Issues
Monitor
Azure Monitor
Application Insights
Deploy
Azure Pipelines
App Center
GitHub Actions
DevOps
Collaborate
Code
Visual Studio
VS Code
GitHub Codespaces
Inner Loop
Build
MSBuild
Test
Test Plans
Playwright
Azure Load Testing

Outer Loop


Plan
Azure Boards
GitHub Issues
Code
Visual Studio
VS Code
GitHub Codespaces
Build
MSBuild
Test
Test Plans
Playwright
Azure Load Testing
Deploy
Azure Pipelines
App Center
GitHub Actions
Monitor
Azure Monitor
Application Insights
DevOps
Collaborate

Agenda

- Introduction
- Patterns, practices, and demos
- Getting started



Are developer satisfaction and high-performing teams related?

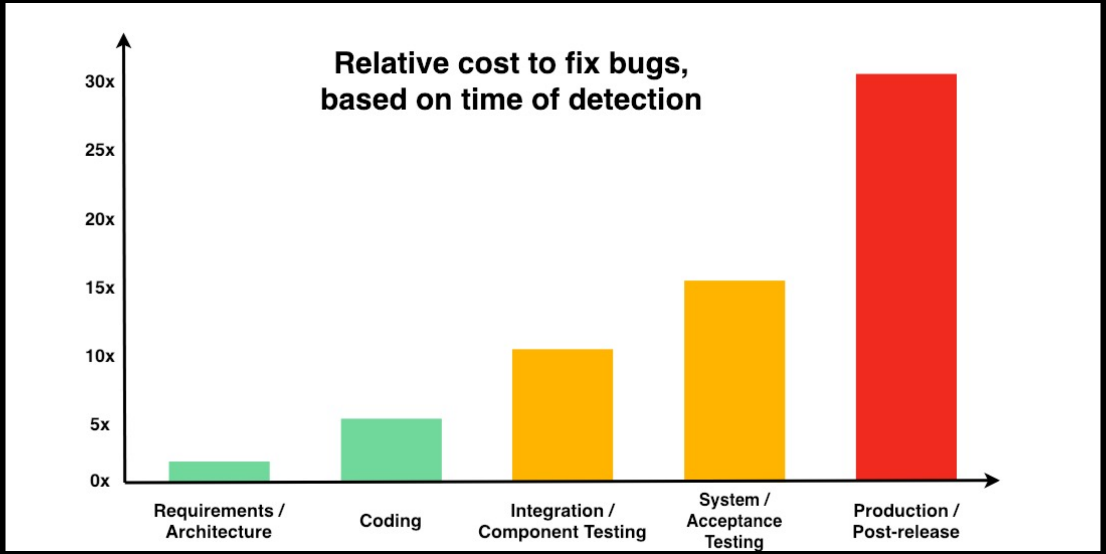
"We compared the proportion of promoters (those who scored 9 or 10) in the high-performing cluster against the proportion of promoters in the low-performing cluster. We found that employees in high-performing teams were 2.2 times more likely to recommend their organization to a friend as a great place to work, and 1.8 times more likely to recommend their team to a friend as a great working environment."

2.2x

Employees in high-performing teams were 2.2 times more likely to recommend their organization as a great place to work.

Source: [2016 State of DevOps Report]

Reduce cost by shifting left



"Most defects end up costing more than it would have cost to prevent them. Defects are expensive when they occur, both the direct costs of fixing the defects and the indirect costs because of damaged relationships, lost business, and lost development time." — *Kent Beck, Extreme Programming Explained*

Patterns & practices

Foundational patterns and practices for pipelines

Pattern: "a named strategy for solving a recurring problem" (Linda Rising)

Poka Yoke

[practice]

Poka Yoke

Japanese term that means "mistake-proofing" or "inadvertent error prevention". A poka-yoke is any mechanism in a process that helps an equipment operator avoid (yokeru) mistakes (poka) and defects by preventing, correcting, or drawing attention to human errors as they occur.

- Less time spent on training workers;
- Elimination of many operations related to quality control;
- Unburdening of operators from repetitive operations;
- Promotion of the work improvement-oriented approach and actions;
- A reduced number of rejects;
- Immediate action when a problem occurs;
- 100% built-in quality control;
- Preventing bad products from reaching customers;
- Detecting mistakes as they occur;
- Eliminating defects before they occur.

Source: Poka-yoke - Wikipedia

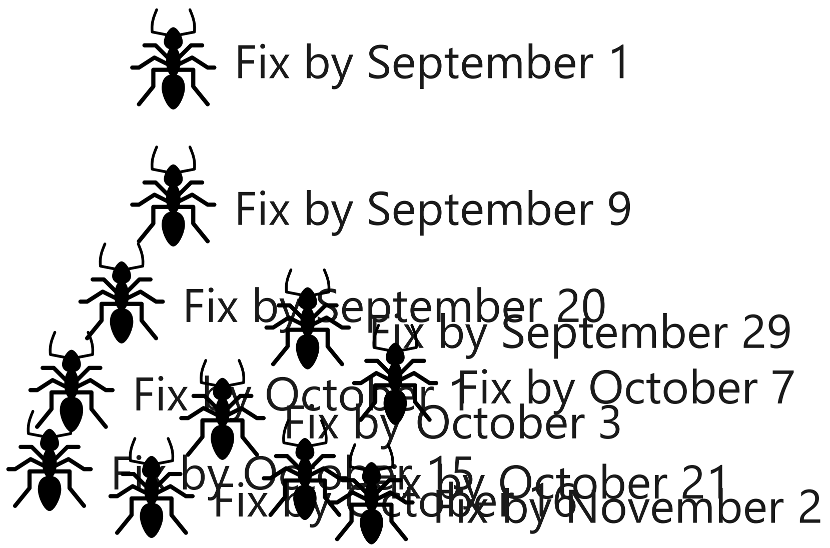
Stop managing bugs, start fixing

[practice]

Stop managing bugs, start fixing

Improve efficiency and reduce context switching

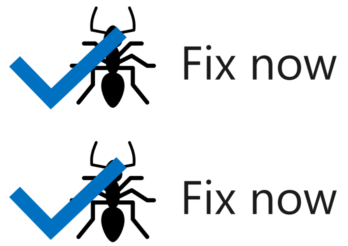
Approach #1: File more bugs



- Harder for developer to remember context
- Potential for other code to build on bad code

Approach #2: Fix before merge

```
This code change introduces 2
new issues. Please fix them
before merging.
```

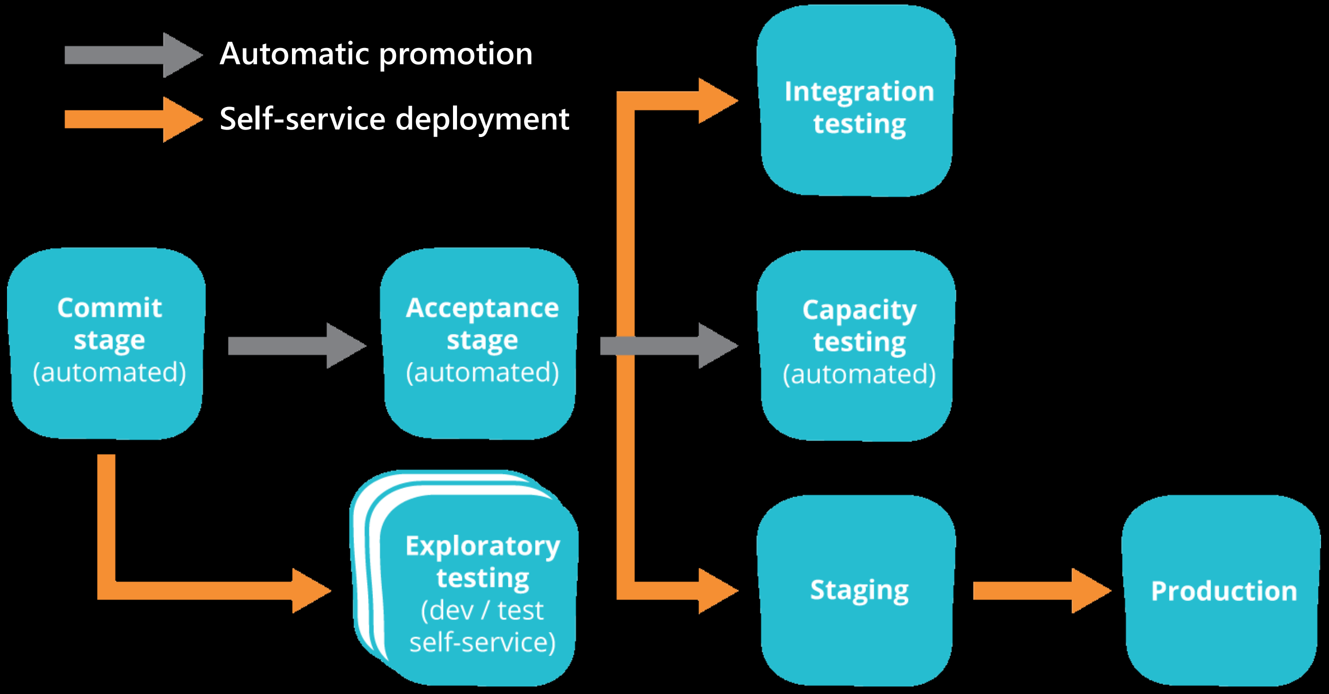


- Developer easily remembers context
- No potential for others to build on bad code
- No opportunity for attacker

Deployment pipeline

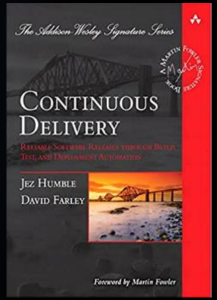
[pattern]

Typical stages in the deployment pipeline



Source: Patterns - Continuous Delivery

Deployment pipeline



"every change in version control triggers a process (usually in a CI server) which creates deployable packages and runs automated unit tests and other validations such as static code analysis"

Source: [Continuous Delivery]

Only build packages once (build once, deploy many)

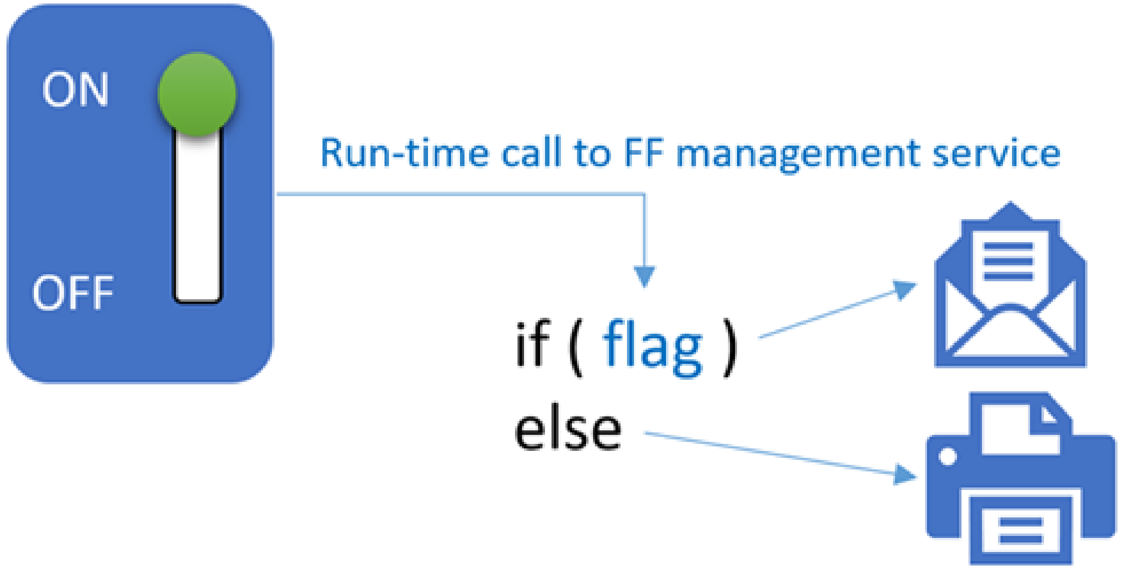
Deploy the same way to every environment

Smoke test your deployments

Keep your environments similar

What are Feature Flags?

It is a technique to enable (expose) or disable (hide) a feature in a solution. It allows us to release and test features, even before they are complete and ready for release. It's an alternative to maintaining multiple source-code branches and a low-risk companion for releasing, managing and fine-tuning features in production.



Why use Feature Flags

- Decouple deployment and exposure
  - Logical deployments to prevent blocking
  - Deploy code in logical chunks without resorting to long-lived branching. Even if a feature is not fully ready, it can be flagged ‘off’, but still deployed.
- Faster development: Spend less time addressing merge conflicts and refactoring old code. Spend more time delivering features that users want.

Yes, it is really good!
(…and everyone knows this)

- Staged Roll-outs and Opt-ins
- Mitigate Risk
  - Quickly turn things off and isolate the problem, without affecting other development environments
  - No more hot-fixes
- Flat Branching – Allows easy working in the Master and reduces long-lived branches
- First step to using this with Telemetry to measure key metrics (Experimentation)
- Giving the engineer the power back!
- Reduce dependency on environments

Where to place your flags

- At the edge?
- In the core?

But there are some complications…

- Adding a new layer of logic and maintenance is hard to do. **Therefore, we want to use a platform that will make this easier.**
- Removing stale flags
- Establishing a multi-instance support system to reduce the risk to customers if a feature is rolled-back
- Increase complexity for testing
- Change in culture and mindset of development

Introducing App Configuration

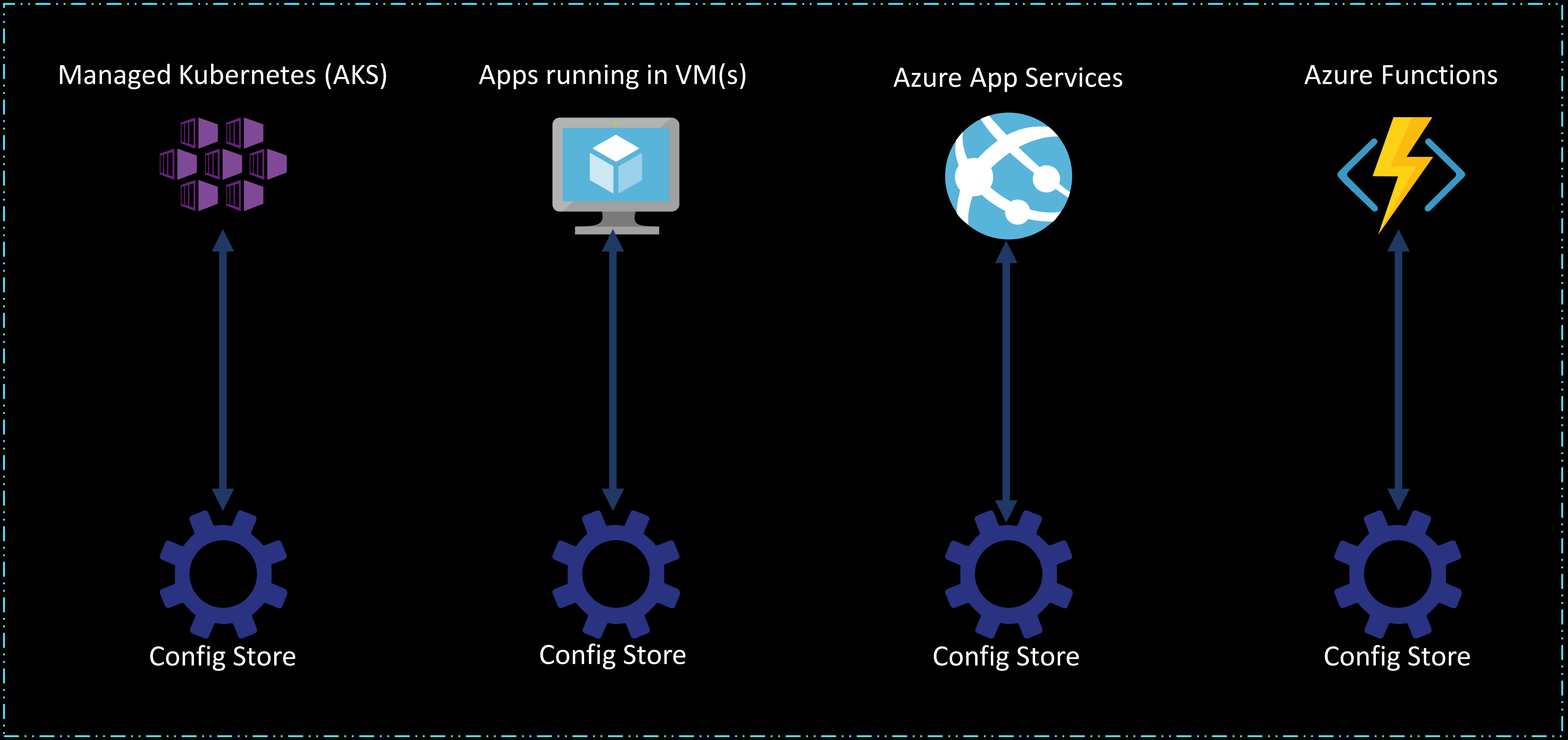
What

"Provides a service to centrally manage application settings and feature flags"

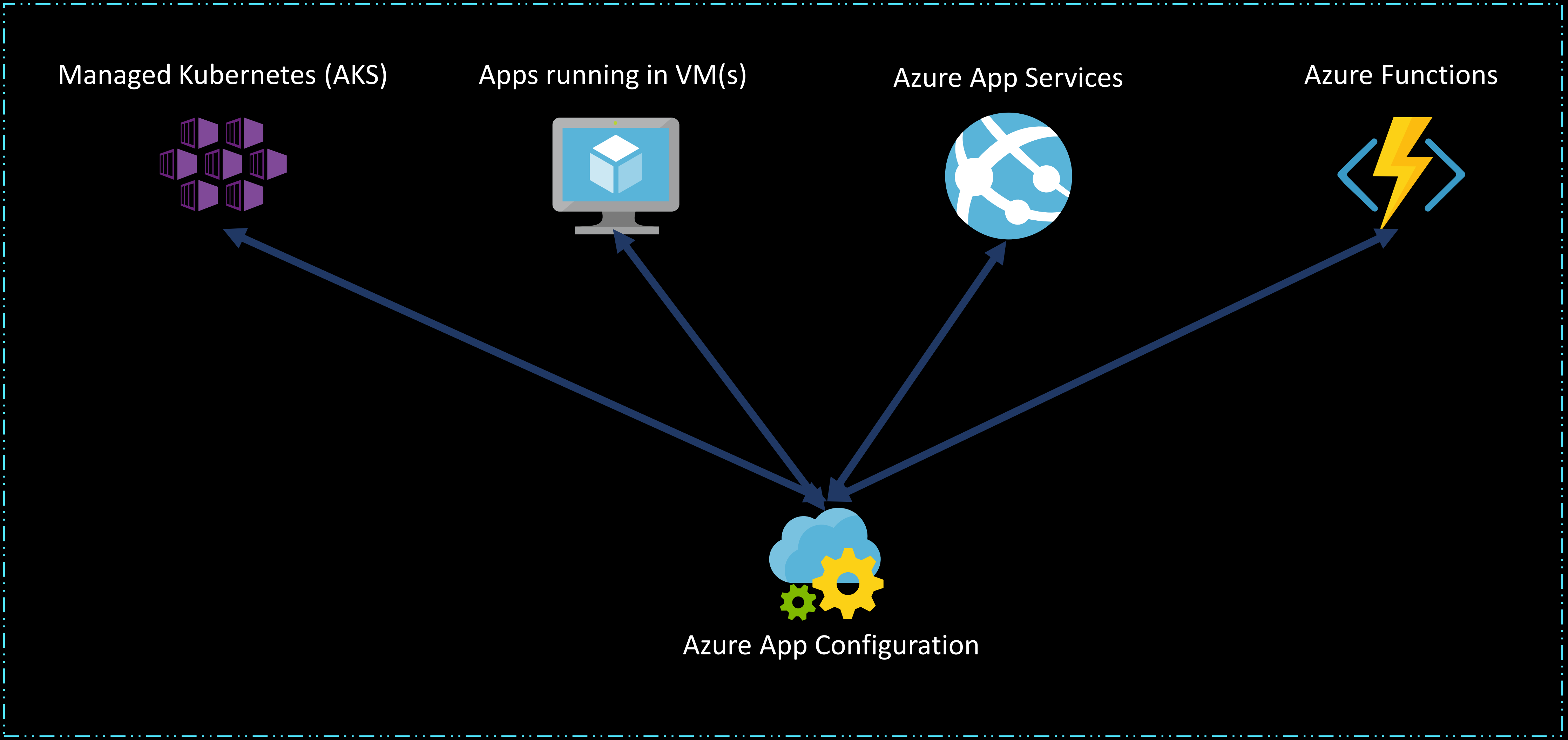
Why

"An application's configuration settings should be kept external to its executable and read in from its runtime environment or an external source."

Traditional App Configuration



With Azure App Configuration



Key Features

Azure App Configuration

- A fully managed service that can be set up in minutes
- Flexible key representations and mappings
- Tagging with labels
- Point-in-time replay of settings
- Dedicated UI for feature flag management
- Comparison of two sets of configurations on custom-defined dimensions
- Enhanced security through Azure-managed identities
- Encryption of sensitive information at rest and in transit
- Native integration with popular frameworks

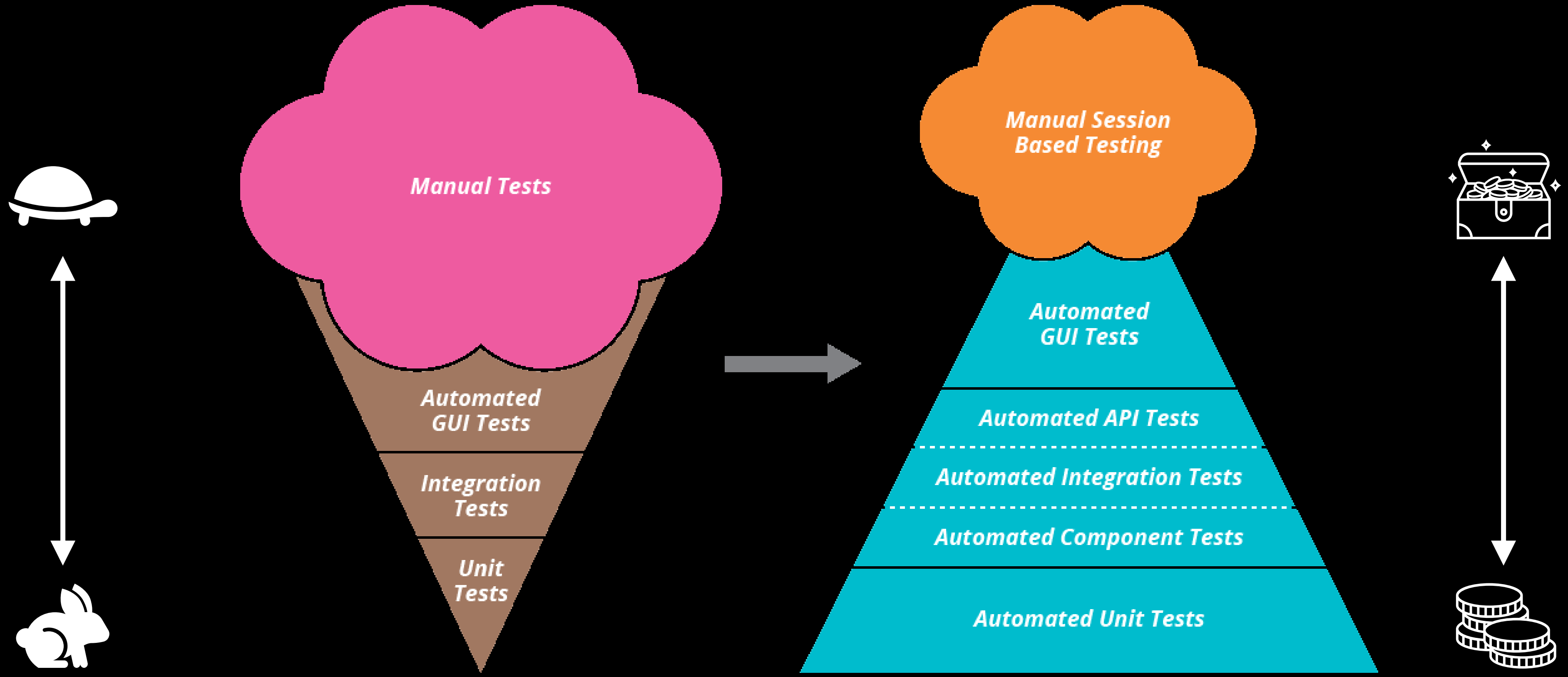
Available Libraries

Azure App Configuration

	How to connect	Quickstart
.NET Core	App Configuration provider for .NET Core	.NET Core quickstart
ASP.NET Core	App Configuration provider for .NET Core	ASP.NET Core quickstart
.NET Framework and ASP.NET	App Configuration builder for .NET	.NET Framework quickstart
Java Spring	App Configuration provider for Spring Cloud	Java Spring quickstart
JavaScript/Node.js	App Configuration client for JavaScript	Javascript/Node.js quickstart
Python	App Configuration client for Python	Python quickstart
Other	App Configuration REST API	None

Test management
[practice]

Ice cream anyone?



Source: [TestPyramid (martinfowler.com)]

Introducing Playwright

Capable automation for the modern web platform

Reliable automation with auto-wait APIs and no timeouts

Easy to get started and toolchain integration

Playwright

Reliable end-to-end testing for modern web apps

Any browser • Any platform • One API

- Cross-browser
- Cross-platform
- Cross-language
- Test Mobile Web

Resilient • No flaky tests

- Auto-wait
- Web-first assertions
- Tracing

Full isolation • Fast execution

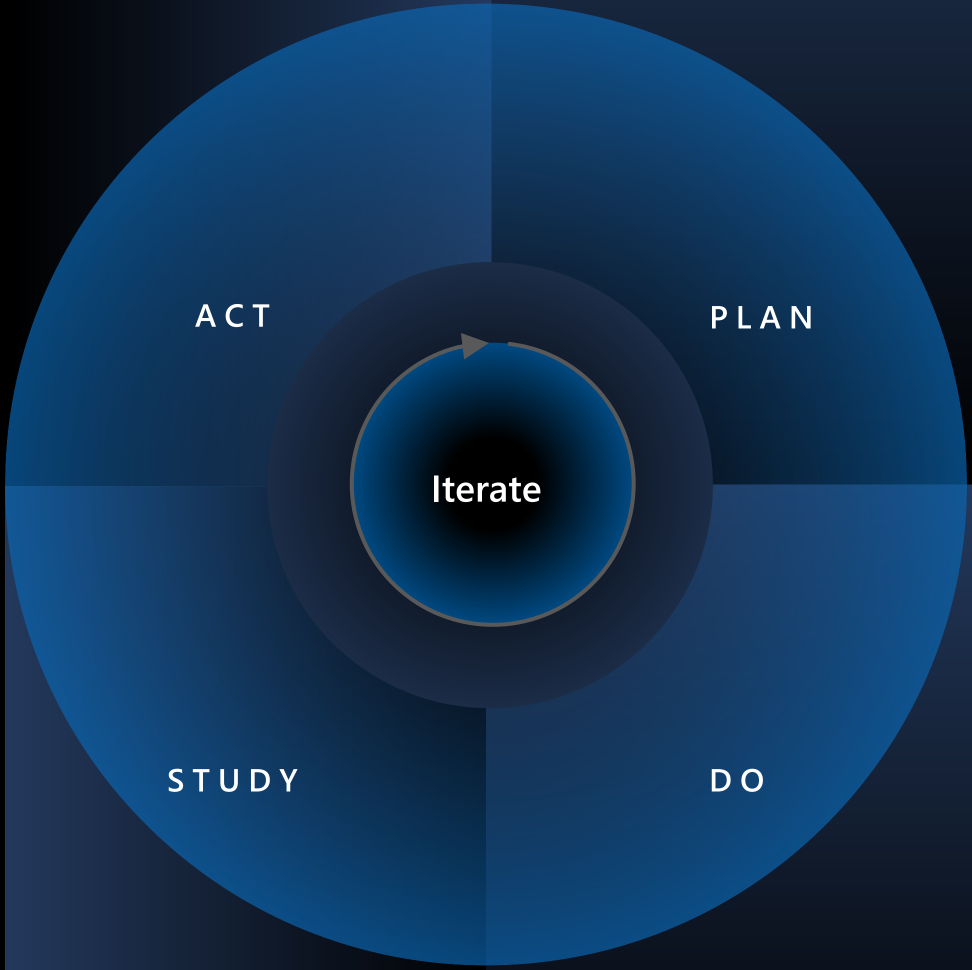
- Browser contexts
- Log in once

Powerful Tooling

- Codegen
- Playwright inspector
- Trace Viewer

How do you get started?

No spot investment, a never-ending journey



Source: PDSA Cycle - The W. Edwards Deming Institute