Azure DevOps Testing - Microsoft Playwright

Radhika Jayaprakash

Cloud Solution Architect

December 2024

Agenda

Azure DevOps - Testing Overview

Load Testing

Chaos Testing

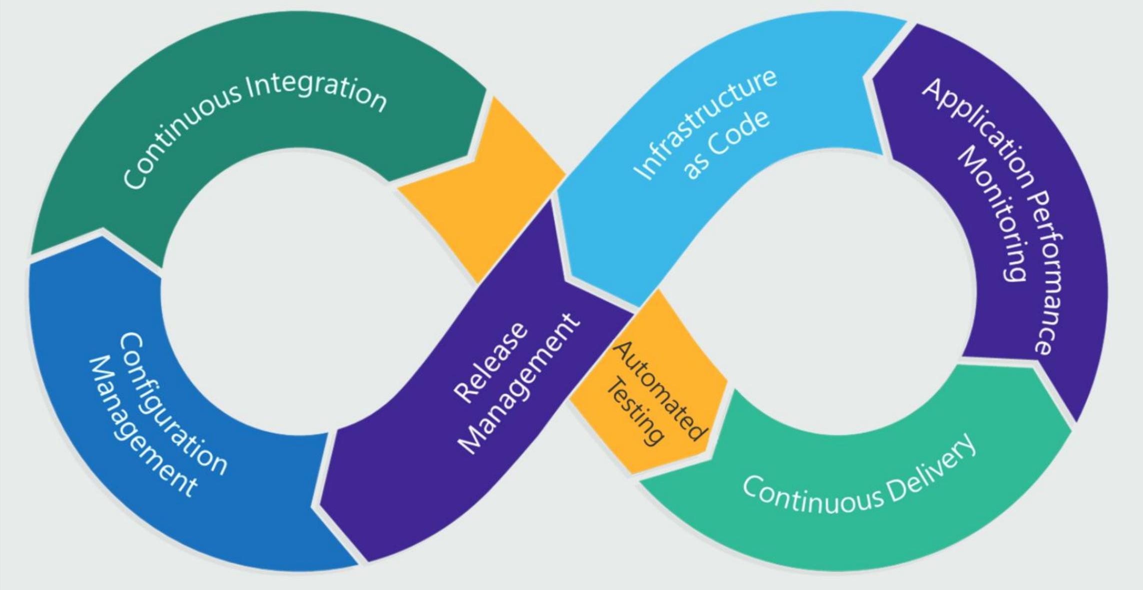
Microsoft Playwright – Basics

Microsoft Playwright – End-End Testing

Microsoft Playwright Services

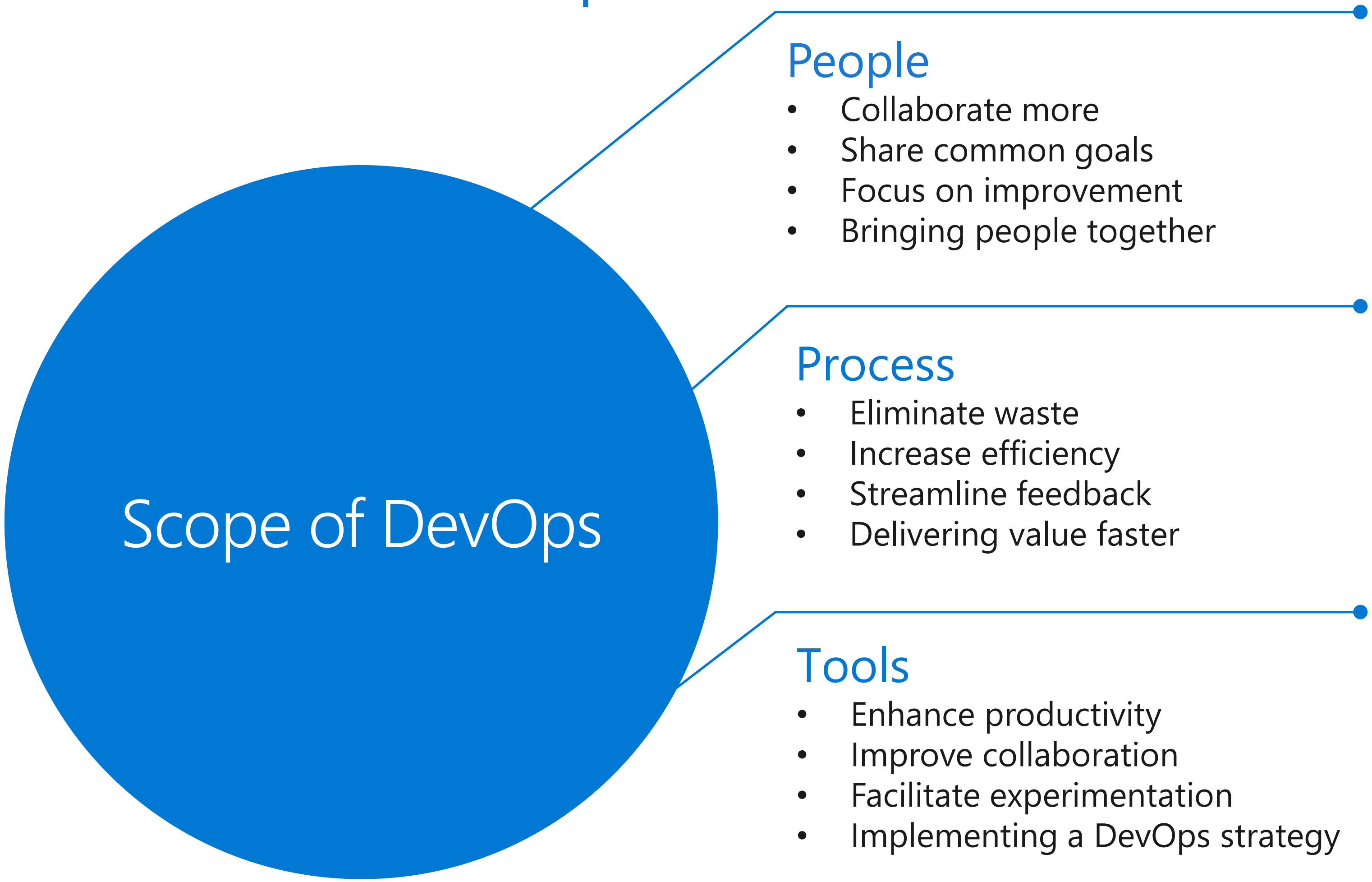
What is DevOps?

Increase flow of value	Shorten cycle times	Continuously Improve



DevOps is the union of people, process, and tools to enable continuous delivery of value to our customers.

What is DevOps?



Scope of DevOps

People

- Collaborate more
- Share common goals
- Focus on improvement
- Bringing people together

Process

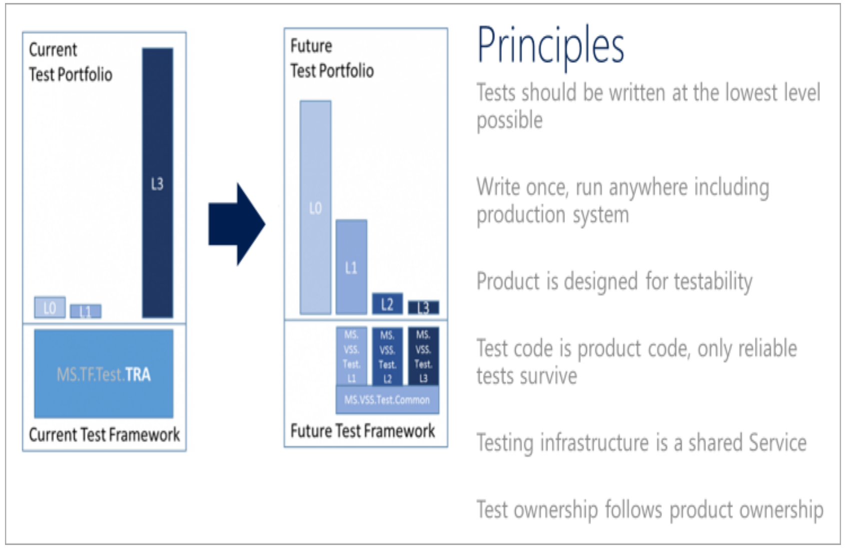
- Eliminate waste
- Increase efficiency
- Streamline feedback
- Delivering value faster

Tools

- Enhance productivity
- Improve collaboration
- Facilitate experimentation
- Implementing a DevOps strategy

DevOps Test Taxonomy

- **L0** and **L1** tests are ***unit tests***
  - Tests that depend on code. L0 is a broad class of fast, in-memory unit tests.
- **L2** are **functional tests**
  - Might require the assembly + other dependencies (SQL or the file system)
- **L3** functional tests run against testable service **deployments**.
  - This test category requires a service deployment, but might use *stubs* for key service dependencies.
- **L4** tests are a restricted class of **integration tests** that run against production (alike).
  - L4 tests require a full product deployment.



Principles

Tests should be written at the lowest level possible

Write once, run anywhere including production system

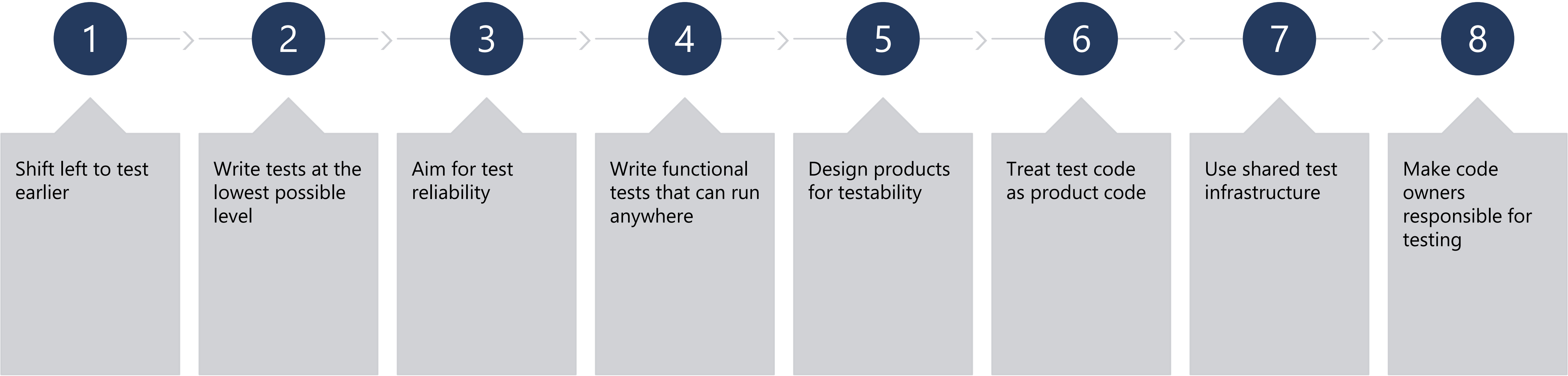
Product is designed for testability

Test code is product code, only reliable tests survive

Testing infrastructure is a shared Service

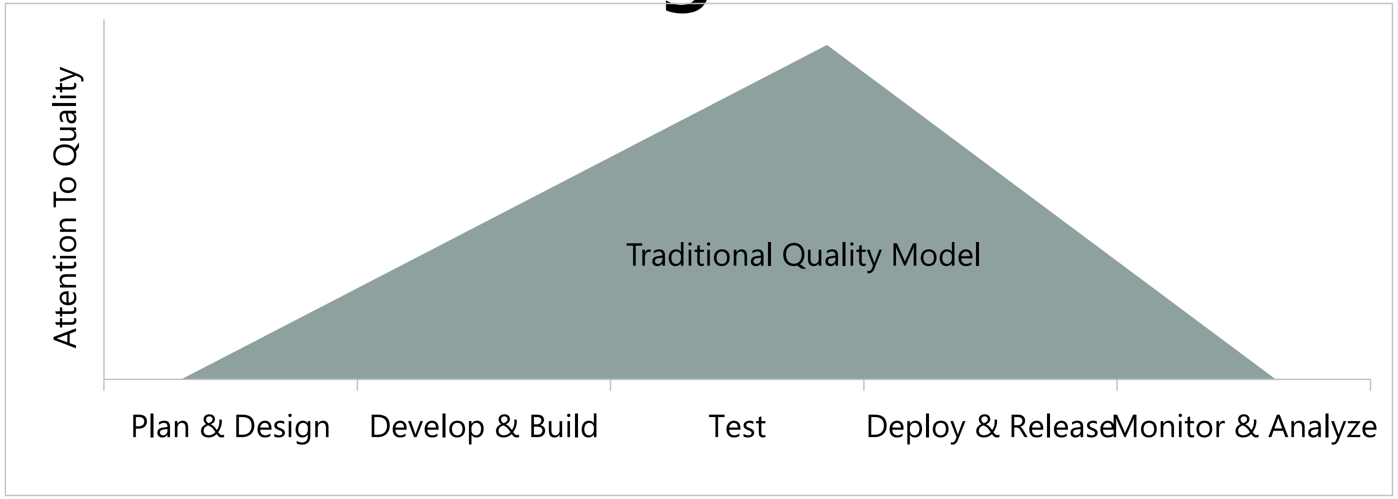
Test ownership follows product ownership

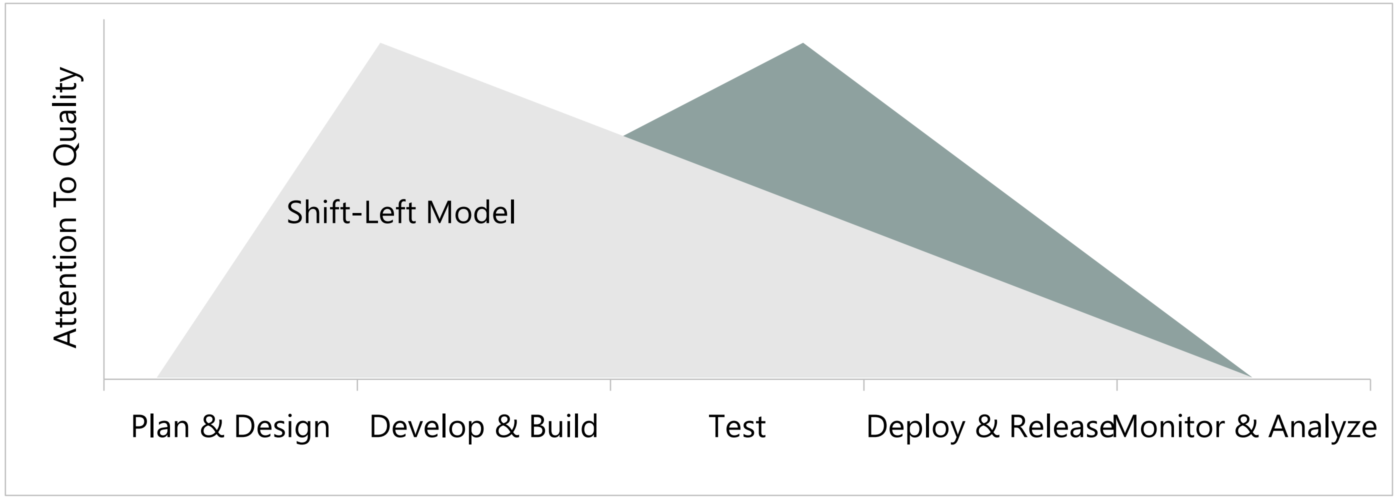
DevOps Test principles



1. Shift left to test earlier
2. Write tests at the lowest possible level
3. Aim for test reliability
4. Write functional tests that can run anywhere
5. Design products for testability
6. Treat test code as product code
7. Use shared test infrastructure
8. Make code owners responsible for testing

Shift-left testing





TDD & BDD

Automated unit test

Mocks & Stubs

CI/CD integration

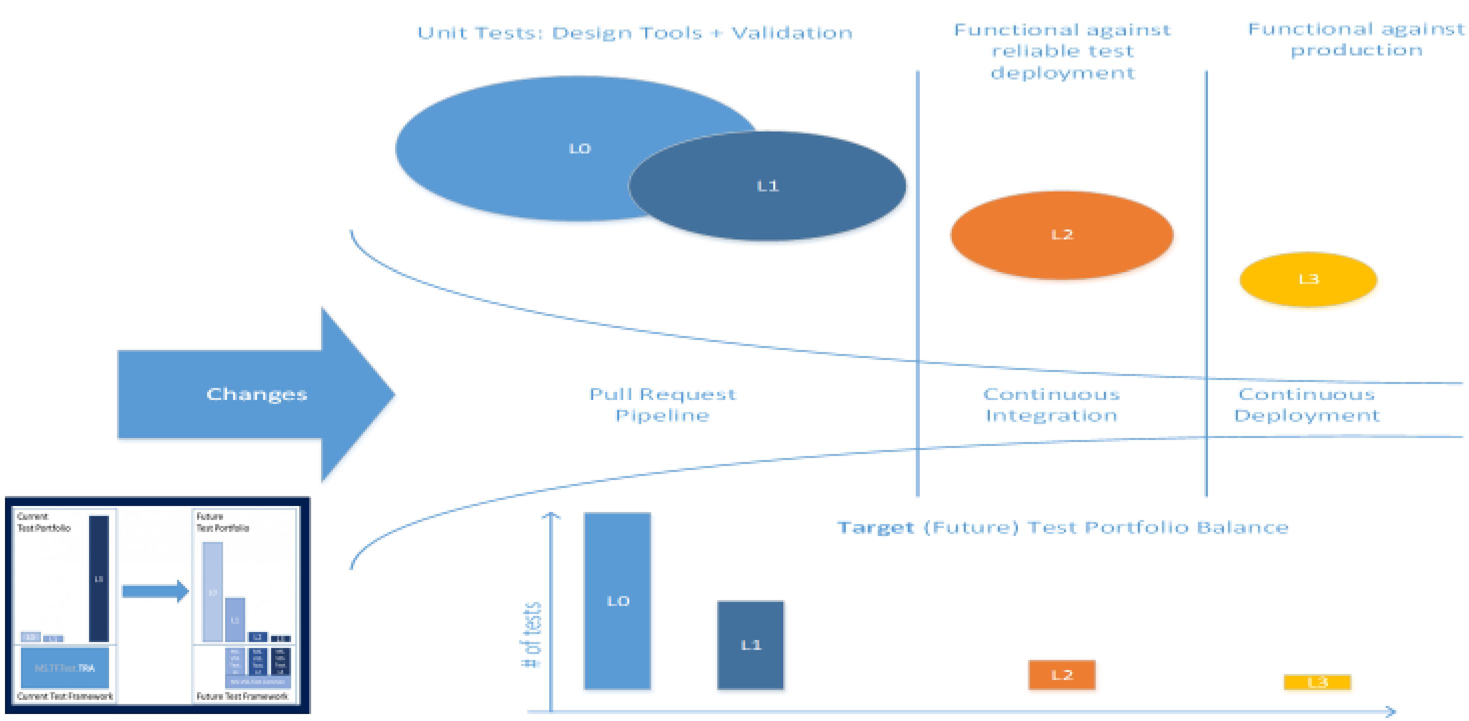
Telemetry generation

Shift left testing is an approach used to speed software testing and facilitate development by moving the testing process to an earlier point in the development cycle. Shifting left is a reference to moving testing to the left on a timeline.

Shift left testing is designed to be a better model for shift left (fast lane) development because traditional testing models that wait until later in the development cycle can bottleneck development.

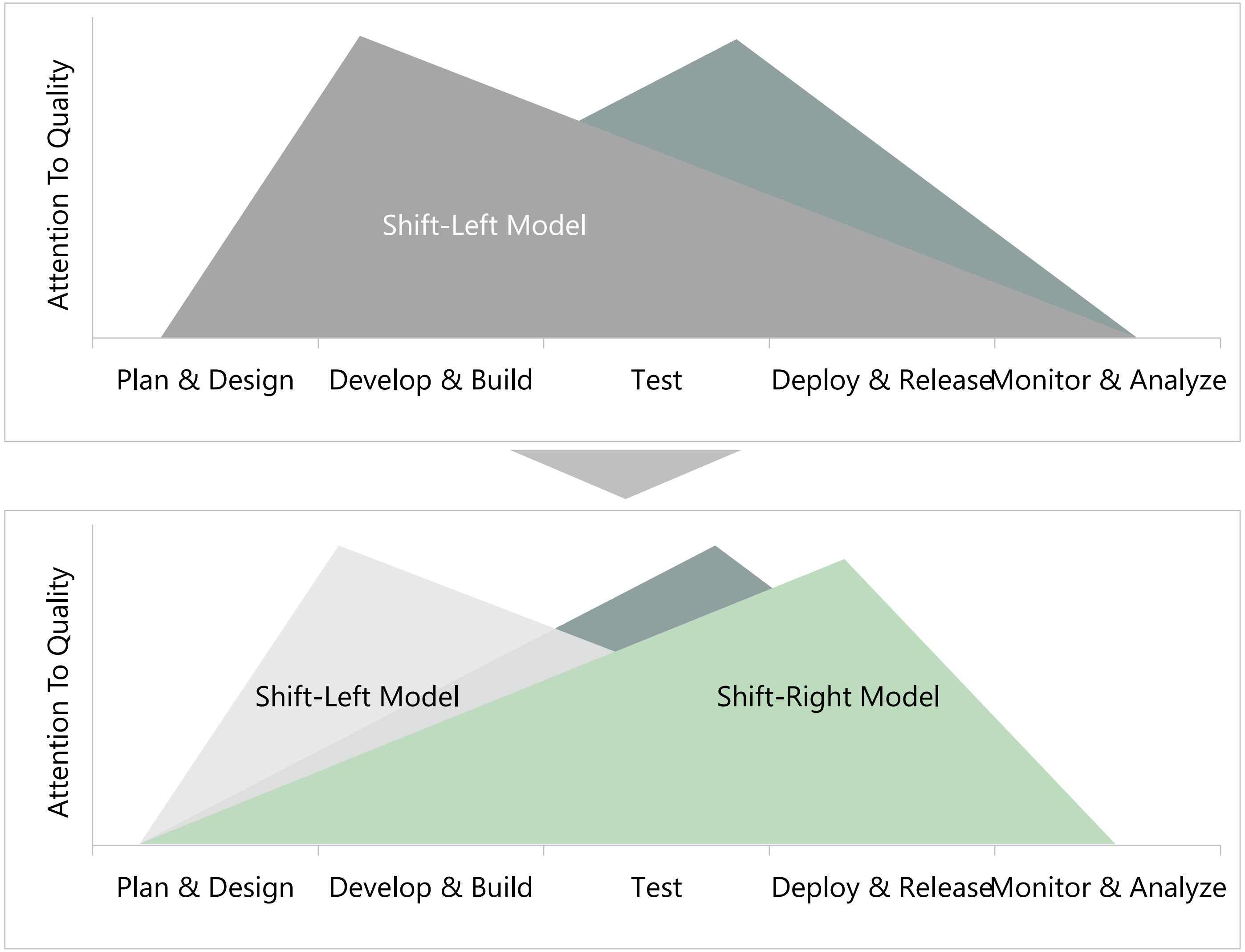
Shift Left to Test easier

“Shift-Left” == Pushing Quality Upstream



- Reduce costs
- Increase efficiency
- Improve collaboration
- Accelerate development
- Mitigate errors
- Strengthen quality
- Enhance security

Shift-right testing



Release rings

Feature Flags

Hypothesis-driven testing

Fault Injection

Insights gained from Telemetry

A common DevOps practice is to not only perform testing early in the development cycle. More often application behavior is also tested in production (“shift right” testing).

Application stability and resiliency can be improved by performing this “shift right” testing, combined with using new capabilities to monitor applications when under stress and/or unexpected (new) elements are introduced.

Shift-right testing increases customer feedback, drives Hypothesis-Driven Testing and enables High Test Coverage.

Introducing Azure DevOps

Azure Boards

Deliver value to your users faster using proven agile tools to plan, track, and discuss work across your teams.

Azure Pipelines

Build, test, and deploy with CI/CD that works with any language, platform, and cloud. Connect to GitHub or any other Git provider and deploy continuously.

Azure Repos

Get unlimited, cloud-hosted private Git repos and collaborate to build better code with pull requests and advanced file management.

Azure Test Plans

Test and ship with confidence using manual and exploratory testing tools.

Azure Artifacts

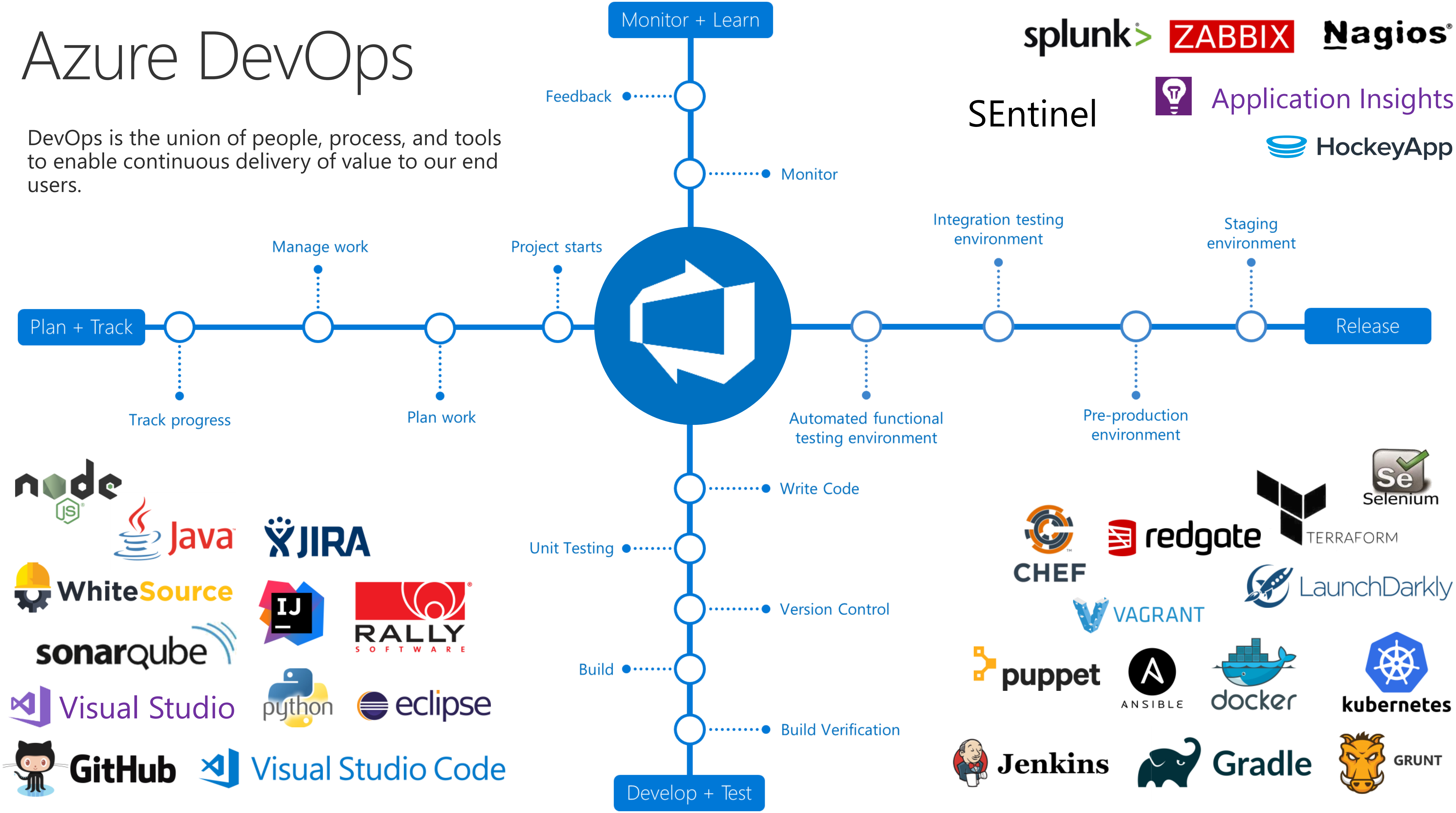
Create, host, and share packages with your team, and add artifacts to your CI/CD pipelines with a single click.

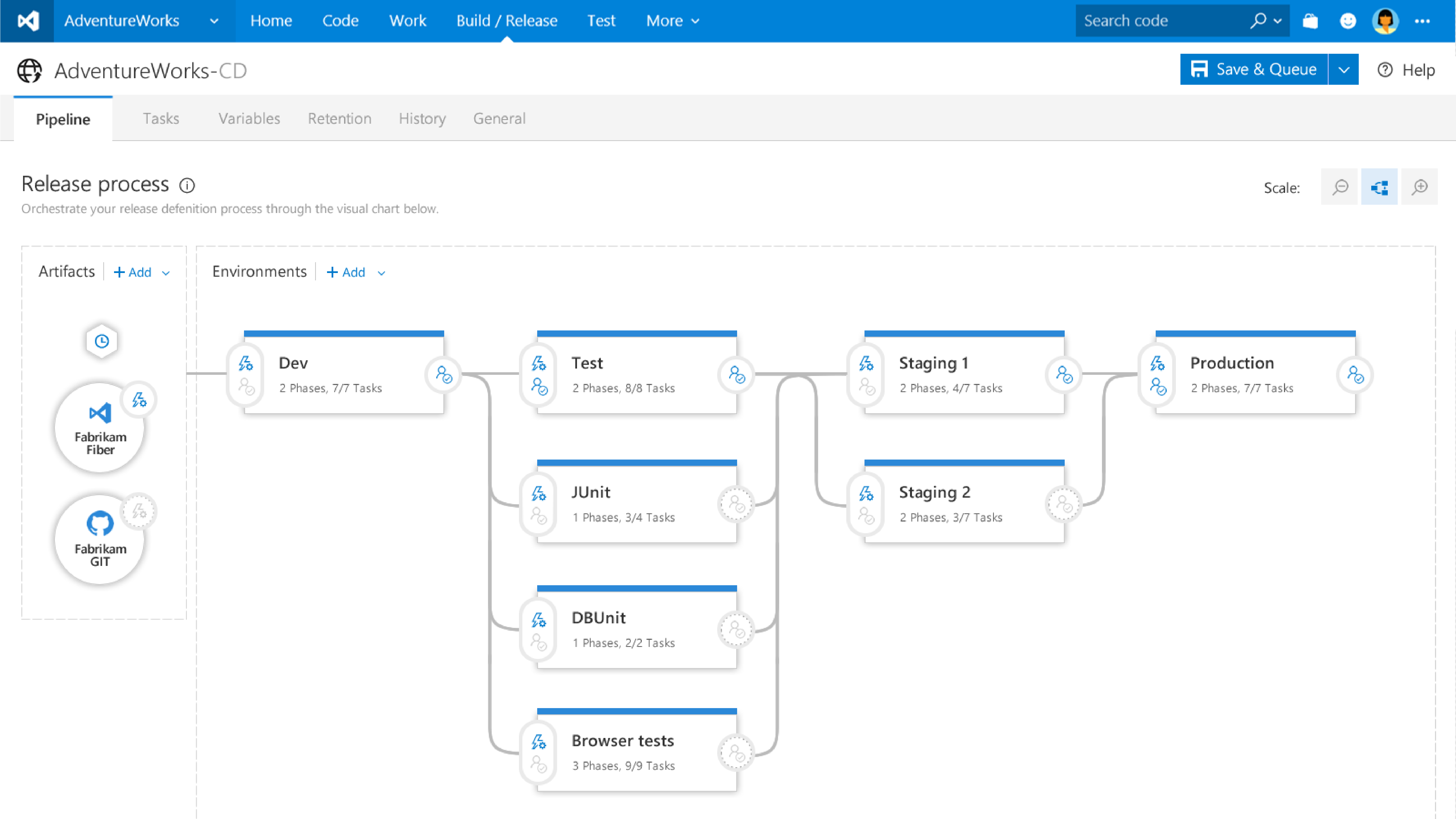
➔

https://azure.com/devops

Azure DevOps

DevOps is the union of people, process, and tools to enable continuous delivery of value to our end users.





AdventureWorks-CD

Release process

Orchestrate your release defenition process through the visual chart below.

Fabrikam Fiber

Fabrikam GIT

Dev
2 Phases, 7/7 Tasks

Test
2 Phases, 8/8 Tasks

JUnit
1 Phases, 3/4 Tasks

DBUnit
1 Phases, 2/2 Tasks

Browser tests
3 Phases, 9/9 Tasks

Staging 1
2 Phases, 4/7 Tasks

Staging 2
2 Phases, 3/7 Tasks

Production
2 Phases, 7/7 Tasks

Azure Load Testing: How to guides on Creating and Running load tests

Azure Load Testing [Getting Started]

Code sample to run a load test to identify performance bottlenecks in a web app, apply changes to resolve the bottleneck and validate the performance improvements.

This code sample show you how to:

- ✓ Deploy the sample app.
- ✓ Create and run a load test.
- ✓ Identify performance bottlenecks in the app.
- ✓ Remove a bottleneck.
- ✓ Rerun the load test to check performance improvements.

The Links below has the instructions on setting up a sample application consists of a Node.js web API, which interacts with a NoSQL database. You'll deploy the web API to Azure App Service web apps and use Azure Cosmos DB as the database, and run an Azure Load test to identify performance bottlenecks, apply changes to resolve the bottleneck and validate the performance improvements

Tutorial: Run a load test to identify performance bottlenecks in a web app | Microsoft Learn

Azure Load Testing [Automating tests using CI/CD]

Code sample to automate regression testing using with CI/CD (Azure DevOps and GitHub Actions)

In this tutorial, you'll set up a CI/CD pipeline that runs a load test for a sample application on Azure. You'll verify the application behavior under load directly from the CI/CD dashboard. You'll then use load test fail criteria to get alerted when the application doesn't meet your quality requirements.

In this tutorial, you'll use a sample Node.js application and JMeter script. The tutorial doesn't require any coding or Apache JMeter skills.

You'll learn how to:

- ✓ Set up the sample application GitHub repository.
- ✓ Configure service authentication for your CI/CD workflow.
- ✓ Configure the CI/CD workflow to run a load test.
- ✓ View the load test results in the CI/CD dashboard.
- ✓ Define load test fail criteria to identify performance regressions.

Instructions for Automating Regression test using :

- Azure DevOps Pipeline
- GitHub Actions

Azure Load Testing [Advanced Use case]

Code sample demonstrating integration with Playwright an Azure Chaos Studio

The Contoso Traders app is a sample application showcasing Playwright, Azure Load Testing, Azure Chaos Studio and more.

This repo contains the source code, deployment templates, and demo scripts for exploring these cloud testing tools. The Contoso Traders app is here: ADD.

Instructions for deploying Contoso Traders Application and running a Azure Load Test :

- Contoso Trader Demo Scripts

Testing Tools

Tool	Purpose
Junit	Test driven development. Unit testing of code written
Selenium	Web User Interface for performing automation testing of the Java application
Cucumber	Behavior driven development. End-User experience , Acceptance testing
Newman	Contract Testing to validate the REST
Azure Load Testing	Performance testing