Microsoft

AZ-400.2 Module 01: Implementing Continuous Integration in an Azure DevOps Pipeline

Lesson 01: Continuous Integration Overview

Lesson 1 Overview

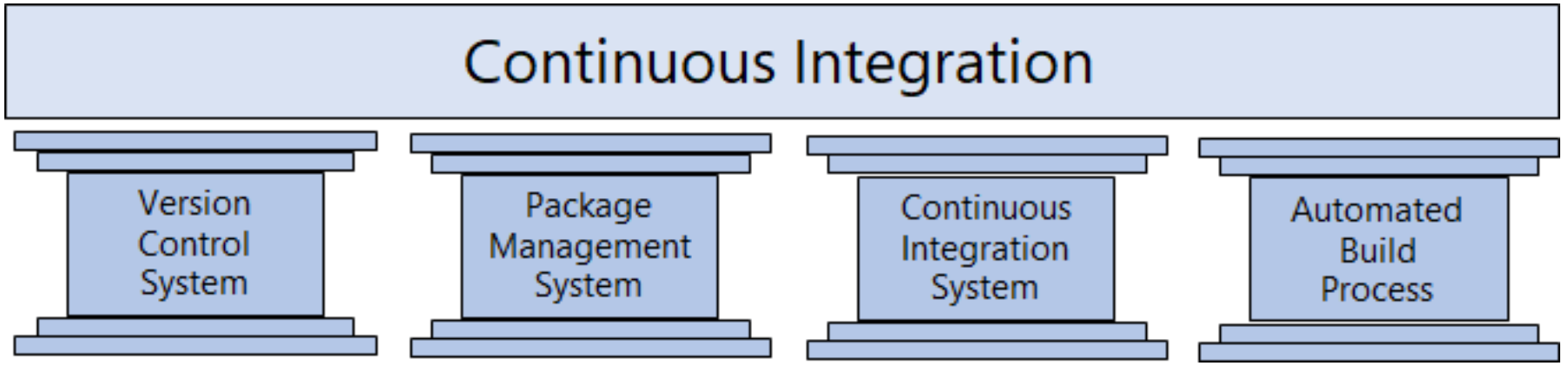
- Introduction to Continuous Integration
- The Four Pillars of Continuous Integration
- Benefits of Continuous Integration
- Continuous Integration Implementation Challenges
- Implementing Continuous Integration in Azure DevOps
- Using Variables to Avoid Hard-coded Values
- Build Number Formatting and Build Status
- Build Authorizations, Timeouts, and Badges
- Configuring Build Retention
- Lab - Enabling Continuous Integration with Azure Pipelines

Introduction to Continuous Integration

- Continuous Integration (CI) is the process of automating the build and testing of code.
- CI encourages developers to share their code and unit tests by merging their changes into the shared version control repository.
- When a change is detected it triggers an automated build system. The code is built using a build definition. Developers respond to any issues or bugs.
- CI keeps the master branch clean ensuring bugs are caught earlier in the development cycle, which makes them less expensive to fix.

The Four Pillars of Continuous Integration

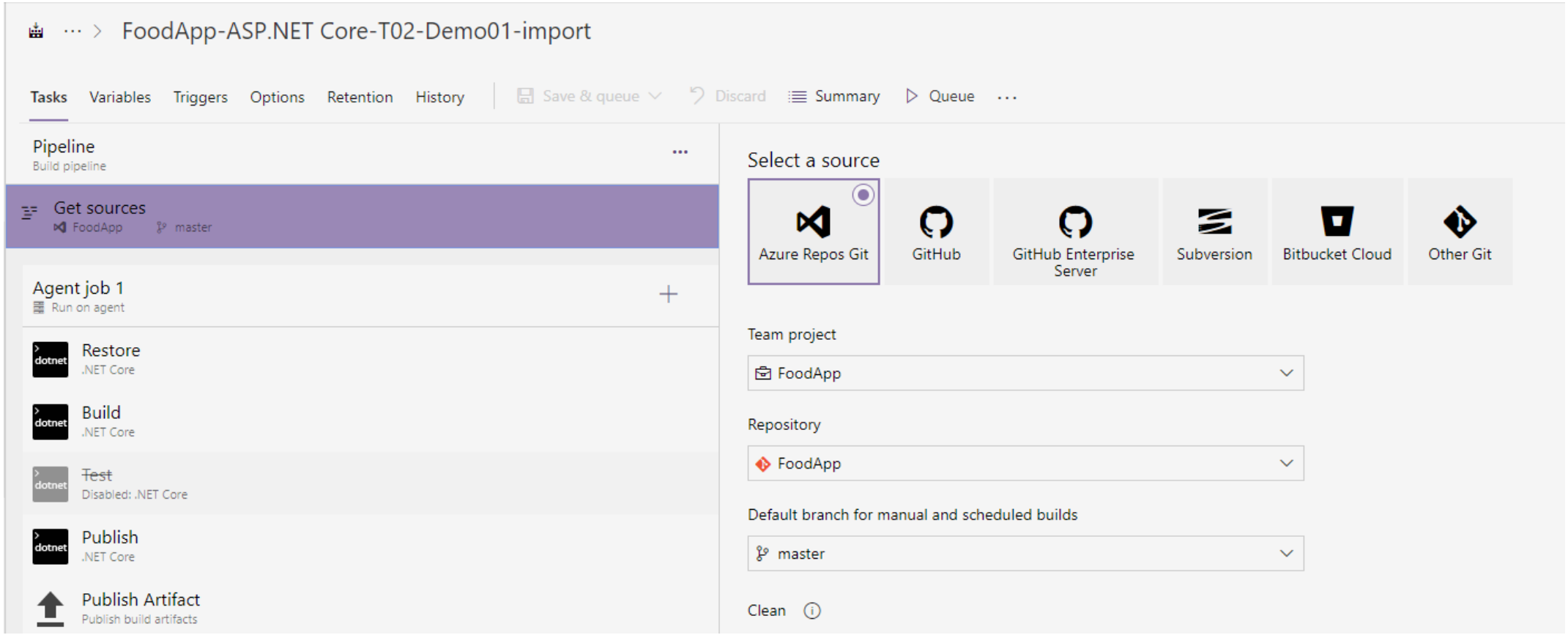
- A **Version Control System** manages changes to your source code over time.
- A **Package Management System** is used to install, uninstall and manage software packages.
- A **Continuous Integration System** merges all developer working copies to a shared mainline several times a day.
- An **Automated Build Process** creates a software build including compiling, packaging, and running automated tests.



Benefits of Continuous Integration

- Improving code quality based on rapid feedback
- Triggering automated testing for every code change
- Reducing build times for rapid feedback and early detection of problems (risk reduction)
- Better management of technical debt and code analysis
- Reducing long, difficult, and bug-inducing merges
- Increasing confidence in codebase health long before production deployment
- Rapid feedback to the developer

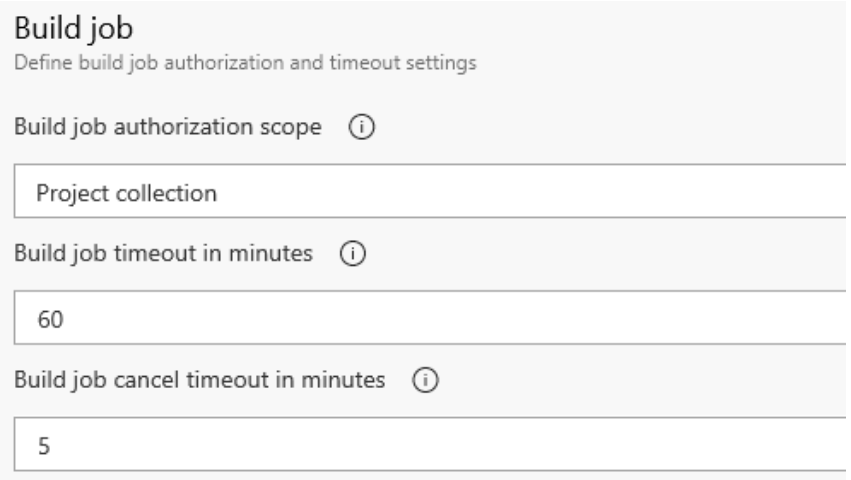
Demonstration: Implementating Continuous Integration in Azure DevOps



Authorization and Timeouts

Authorization and Timeouts (scope, job timeout, cancel job timeout)

- Specify the authorization scope for a build job.
- Project Collection if the build needs access to multiple projects.
- Current Project if you want to restrict this build to have access only the resources in the current project.



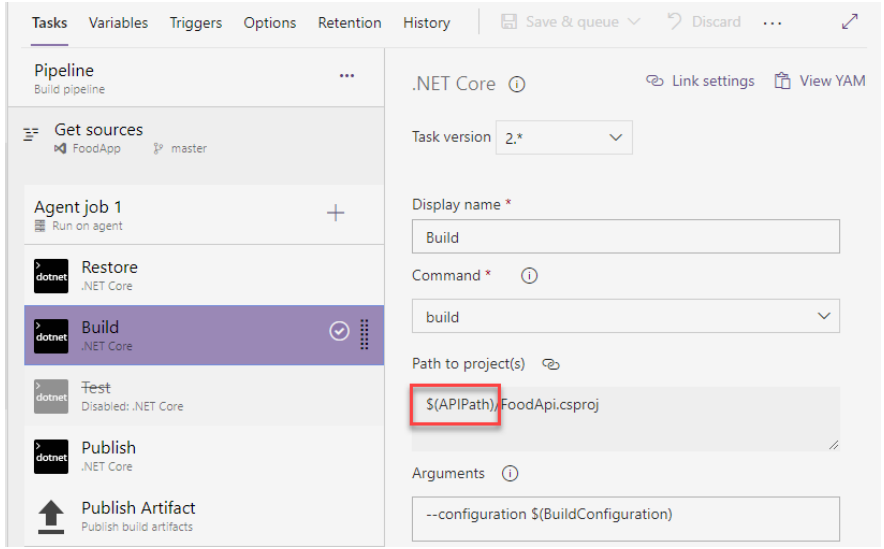
Using Variables

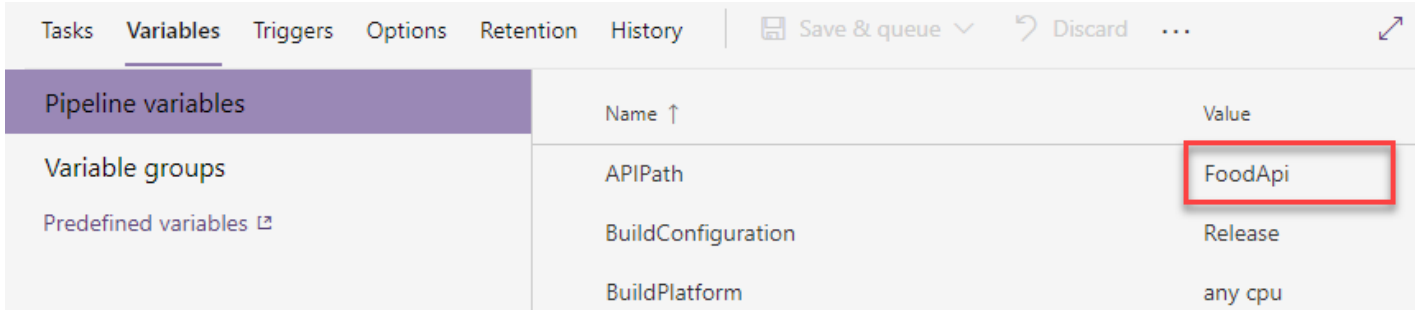
Help to reduce complexity when working with

- Environments and
- Ressources

Several Variable Types

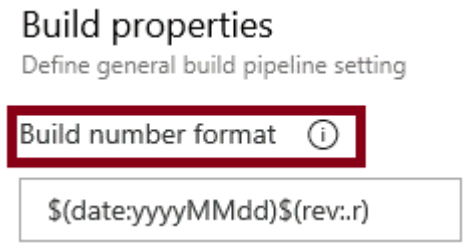
- Predefined
- Custom
  - Global / Step specific
- Variable Groups
  - Make Accessibe through different pipelines



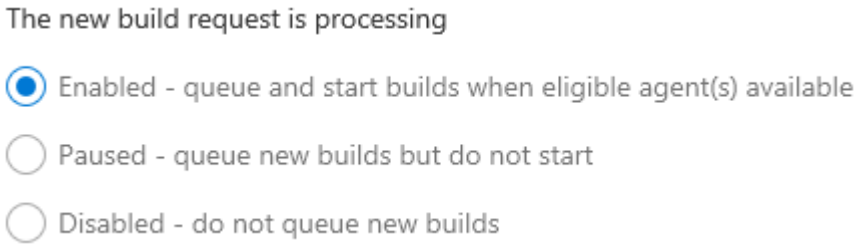


Build Number Formatting and Build Status

- Build number formatting

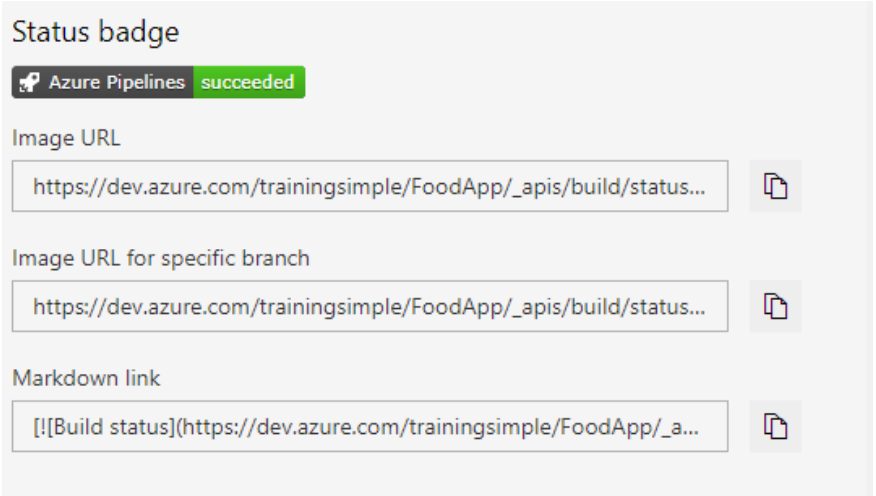


- Build status (enabled, paused, disabled)



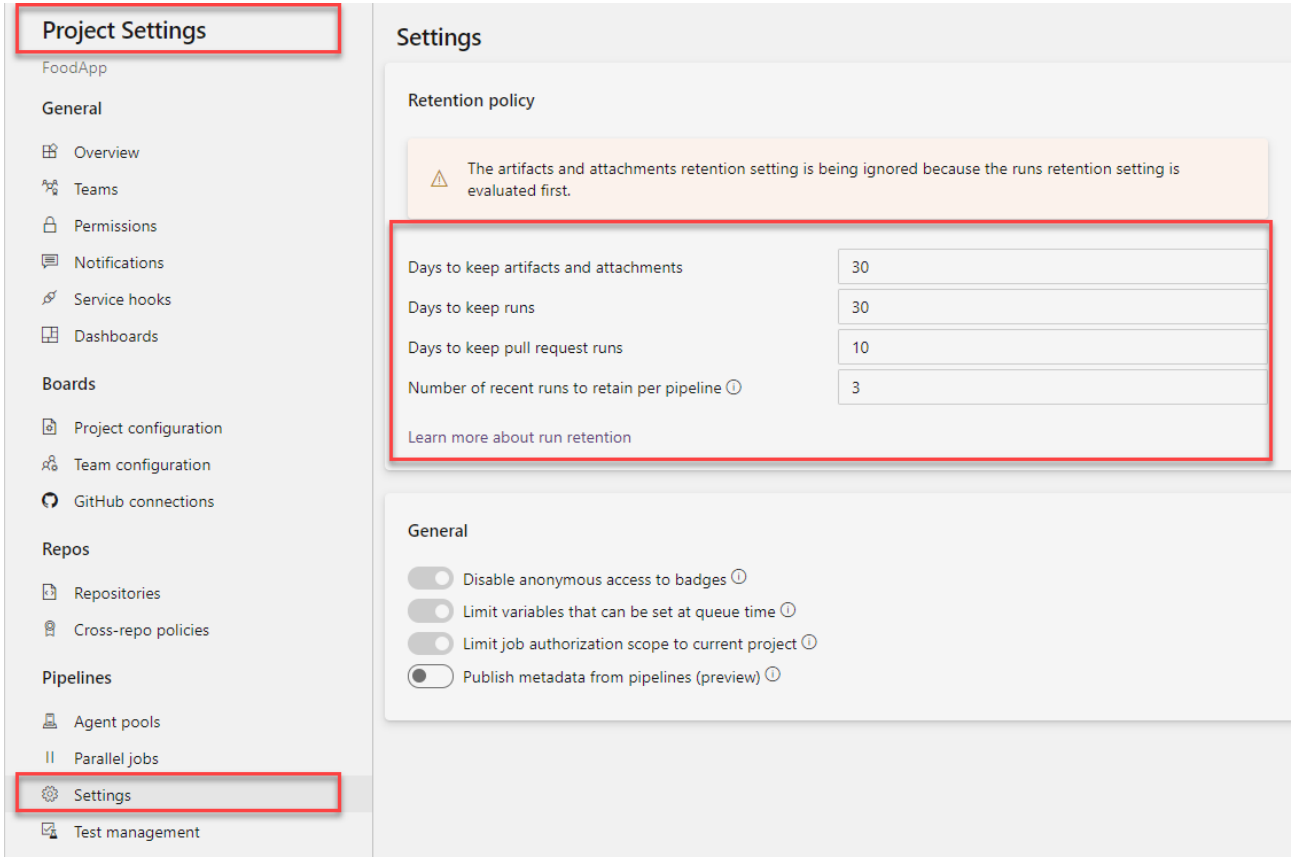
Badges

- Indicate the State of a Build - can be added to GitHub



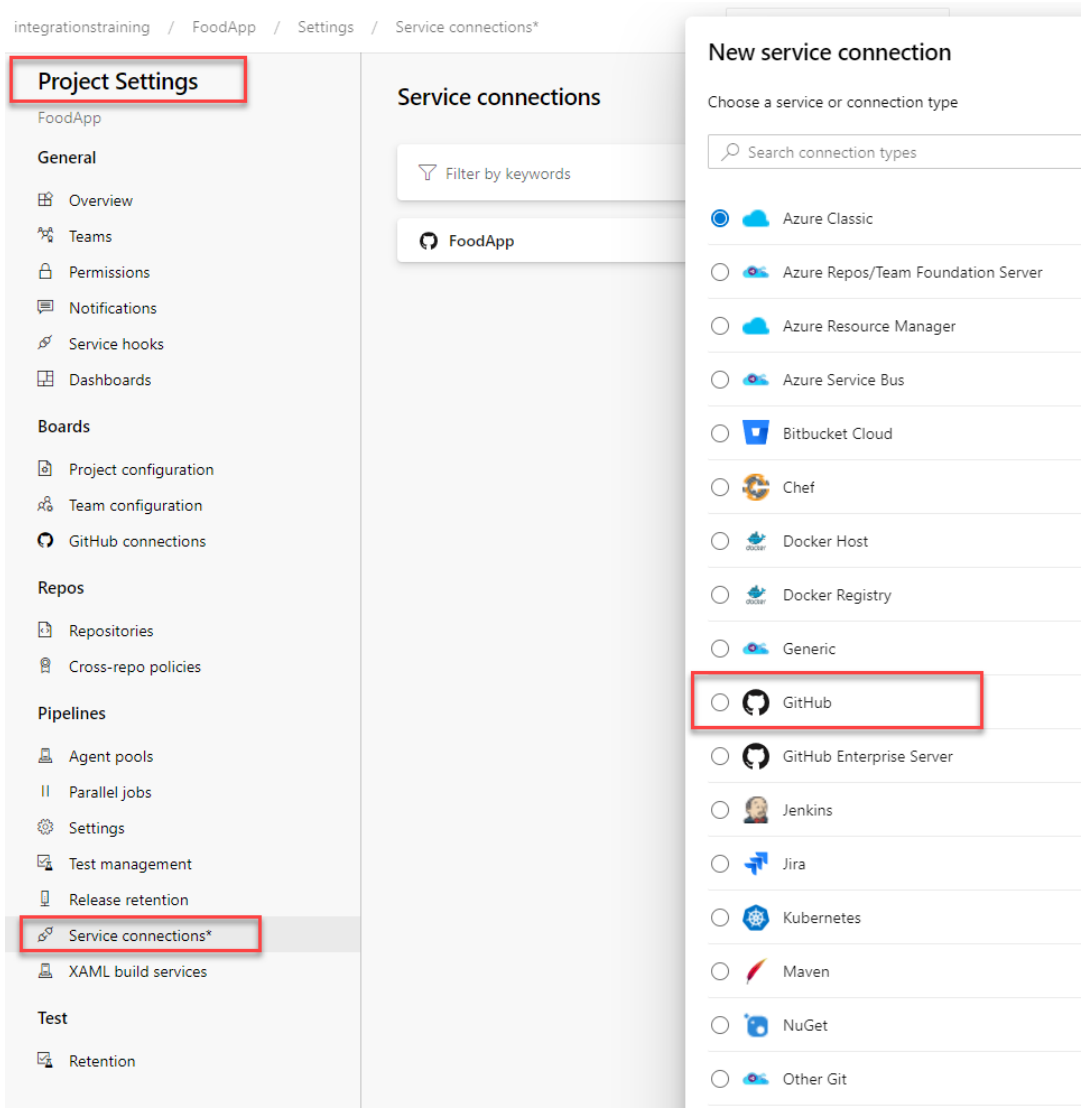
Configuring Build Retention

Retention policies are used to configure how long runs and releases are to be retained by the system



Service Connections

- Service Connections allow consumption of external ressources such as Azure, GitHub, …
- Azure Pipelines can automatically build and validate every pull request and commit to your GitHub repository



Lab: Enabling Continuous Integration with Azure Pipelines

In this hands-on lab, you will learn how to configure continuous integration with Azure Pipelines. You will perform the following tasks:

- Creating a basic build pipeline from a template
- Tracking and reviewing a build
- Invoking a continuous integration build

✔ Note that you must have already completed the prerequisite labs in the Welcome section.

Lesson 02: Implementing a Build Strategy

Lesson 2 Overview

- Automated Build Workflows
- Implementing Build Triggers
- Working with Hosted Agents
- Implementing a Hybrid Build Process
- Configuring Agent Demands
- Implementing Multi-Agent Builds
- Build-Related Tooling
- Creating a Jenkins Build Job and Triggering CI

Automated Build Workflows

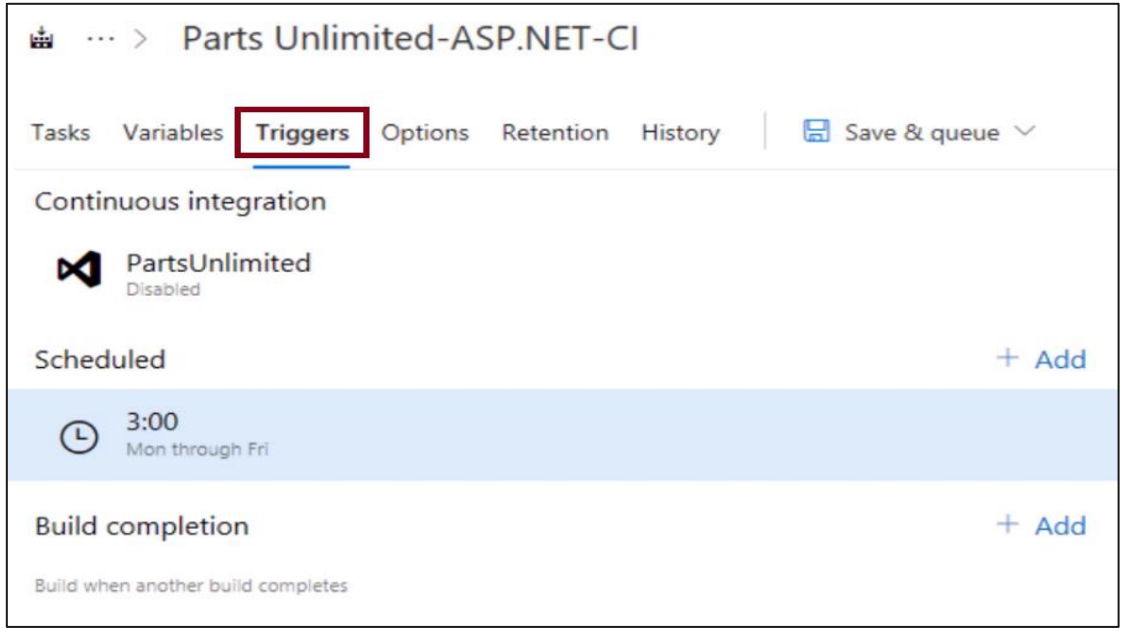
- Azure DevOps can automate a custom workflow that's as large and complex as you need
- Agile teams normally require more than one type of build
- Builds are typically triggered automatically when code is committed
- Builds can also be scheduled – such as a daily build

Implementing Build Triggers

ARambazamba.FoodApp

master ARambazamba/FoodApp / azure-pipelines.yml *

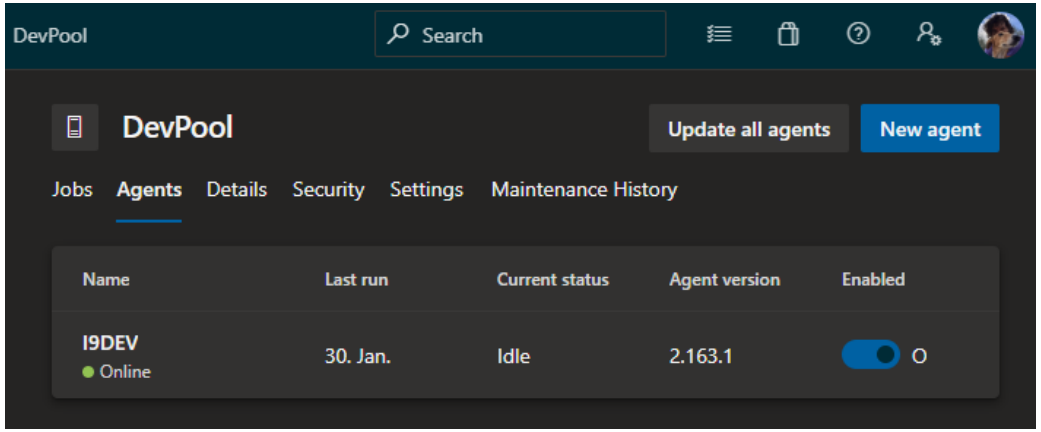
```yaml
trigger:
  # Which branch triggers the pipeline
  branches:
    include:
    - master
    - development
  # Which path triggers the pipeline -> Monorepo
  paths:
    include:
      - FoodApi/*
```



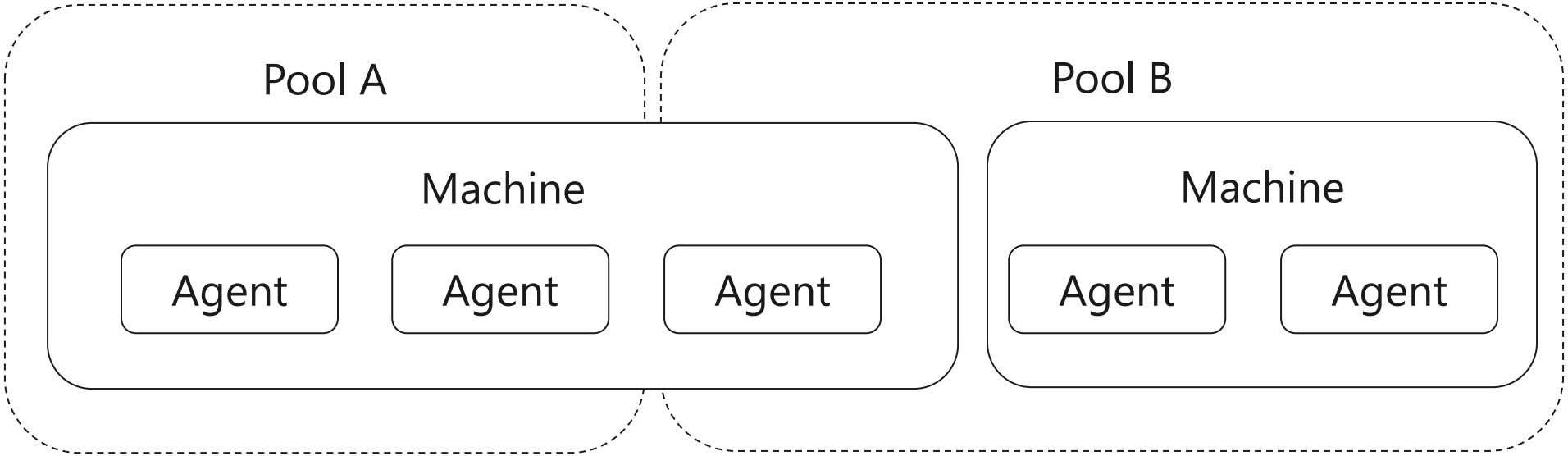
Lesson 03: Evaluate Use of Hosted vs Private Agents

Hosted vs Private Agents

- An agent is installable software that runs one build or deployment job at a time
- Two types of agents:
  - **Microsoft-hosted agents -** Automatically take care of maintenance and upgrades. Each time you run a pipeline, you get a fresh virtual machine. The virtual machine is discarded after one use.
  - **Self-hosted agents –** You take care of maintenance and upgrades. Give you more control to install dependent software needed. You can install the agent on Linux, macOS, Windows machines, or even in a Linux Docker container.



Agent Pools

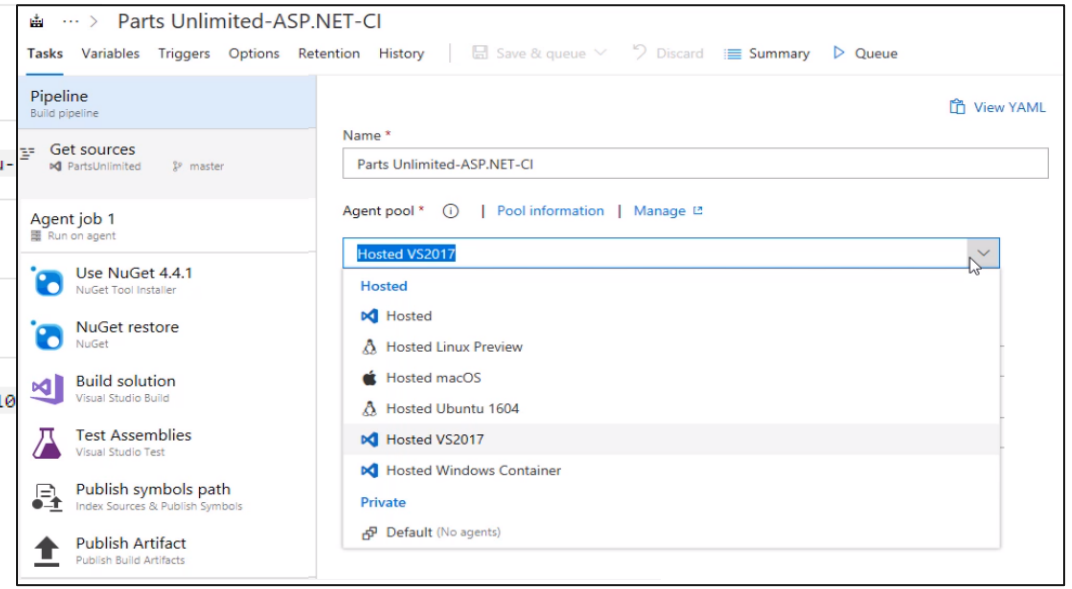


- You can organize agents into agent pools - Defines the sharing boundary
- In Azure Pipelines, agent pools are scoped to the Azure DevOps organization; so you can share an agent pool across projects

Working with Hosted Agents

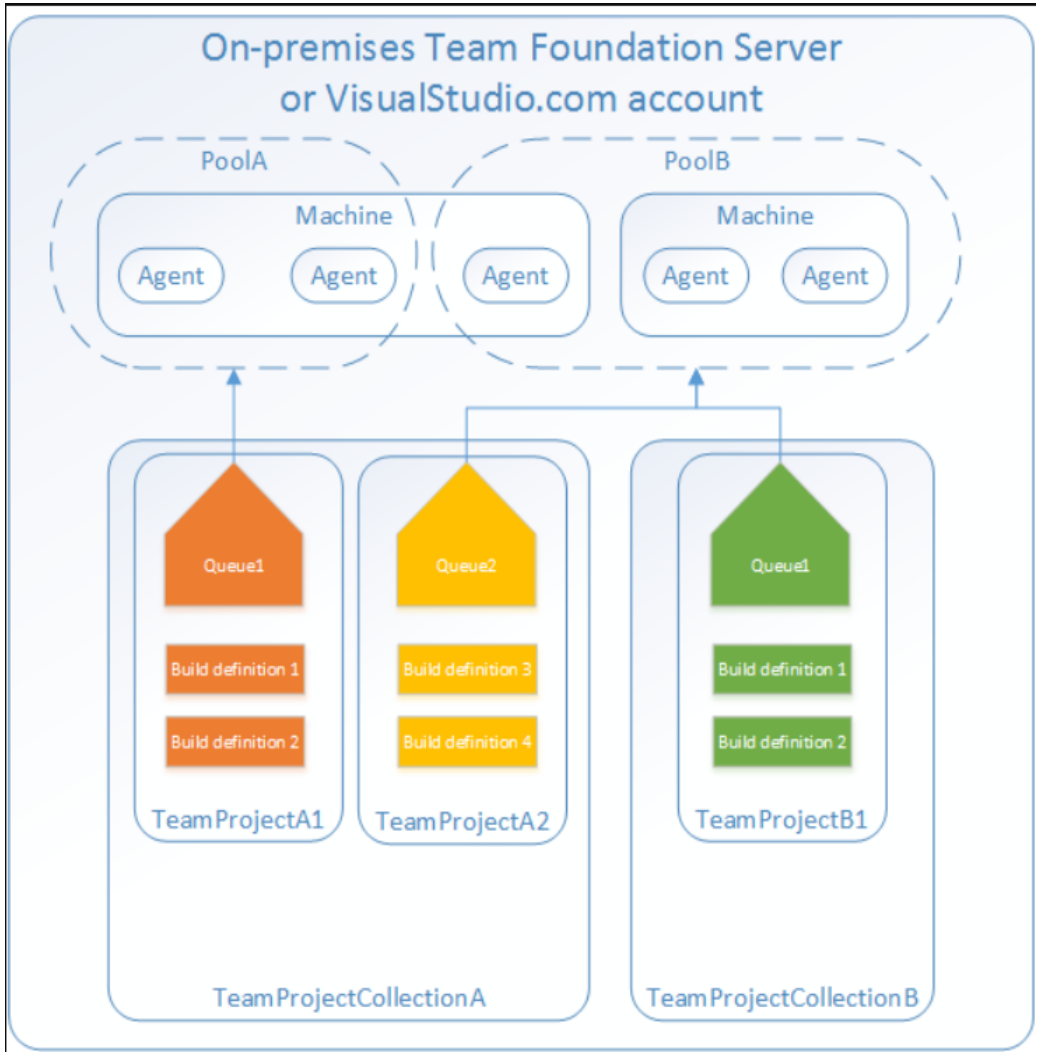
- Azure Pipelines provides a Microsoft-hosted agent pool named Azure Pipelines that offers several virtual machine images to choose from

Image	Classic Editor Agent Specification	YAML VM Image Label
Windows Server 2019 with Visual Studio 2019	*windows-2019*	`windows-latest` OR `windows-2019`
Windows Server 2016 with Visual Studio 2017	*vs2017-win2016*	`vs2017-win2016`
Ubuntu 18.04	*ubuntu-18.04*	`ubuntu-latest` OR `ubuntu-`
Ubuntu 16.04	*ubuntu-16.04*	`ubuntu-16.04`
macOS X Mojave 10.14	*macOS-10.14*	`macOS-10.14`
macOS X Catalina 10.15	*macOS-10.15*	`macOS-latest` OR `macOS-10`



Typical Situations for Agent Pools

- You're a member of a project and you want to use a set of machines owned by your team for running build and deployment jobs
- You're a member of the infrastructure team and would like to set up a pool of agents for use in all projects
- You want to share a set of agent machines with multiple projects, but not all of them



Security of Agent Pools

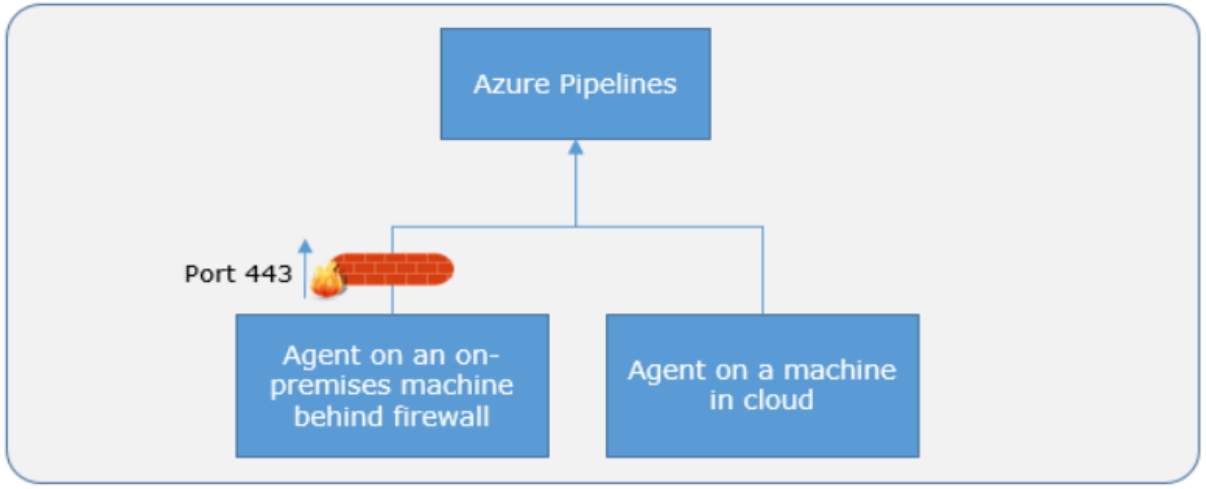
- Roles are defined on each agent pool, and membership in these roles governs what operations you can perform on an agent pool

Role	Purpose
Reader	Can view the organization agent pool as well as agents. You typically use this to add operators that are responsible for monitoring the agents and their health.
Service Account	Can use the organization agent pool to create a project agent pool in a project. If you follow the guidelines above for creating new project agent pools, you typically do not have to add any members here.
Administrator	In addition to all the above permissions, members of this role can register or unregister agents from the organization agent pool. They can also refer to the organization agent pool when creating a project agent pool in a project. Finally, they can also manage membership for all roles of the organization agent pool. The user that created the organization agent pool is automatically added to the Administrator role for that pool.

Lesson 08: Setup Private Agents

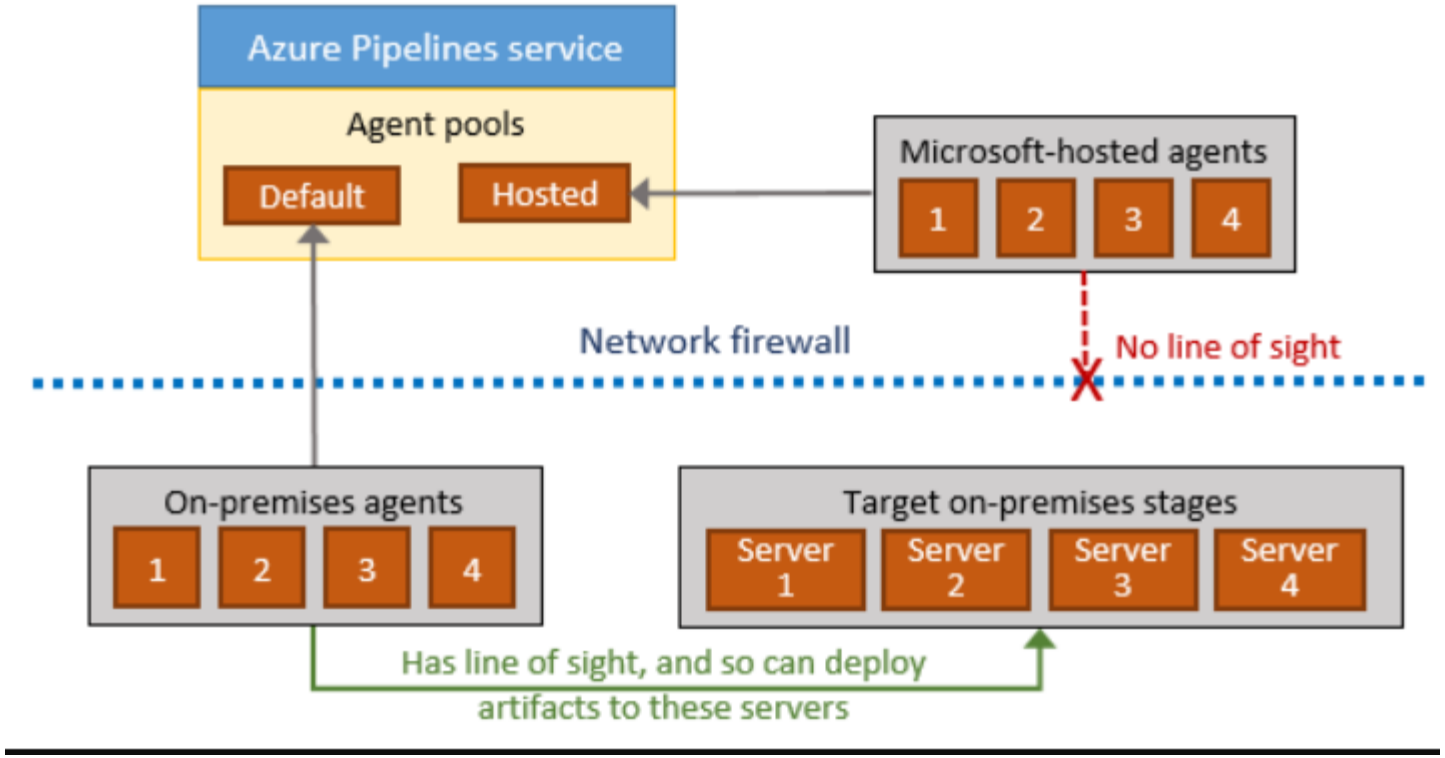
Communication with Azure Pipelines

- The agent determines which job it needs to run, and to report the logs and job status
- Communication is always initiated by the agent
- All the messages from the agent to Azure Pipelines are over HTTPS



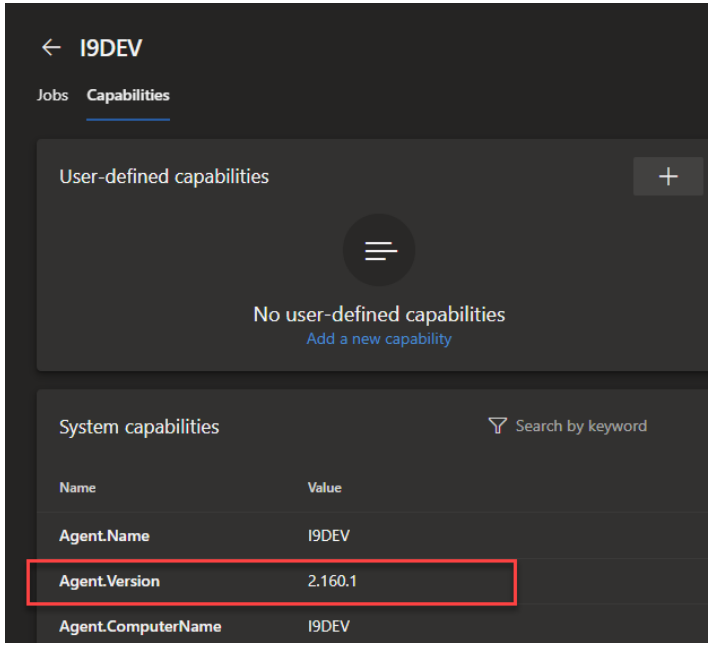
Communication to Deploy to Target Servers

- Agent must have "line of sight" connectivity to servers
- Microsoft-hosted agent pools, by default, have connectivity to Azure websites and servers running in Azure
- You may need to manually configure connectivity



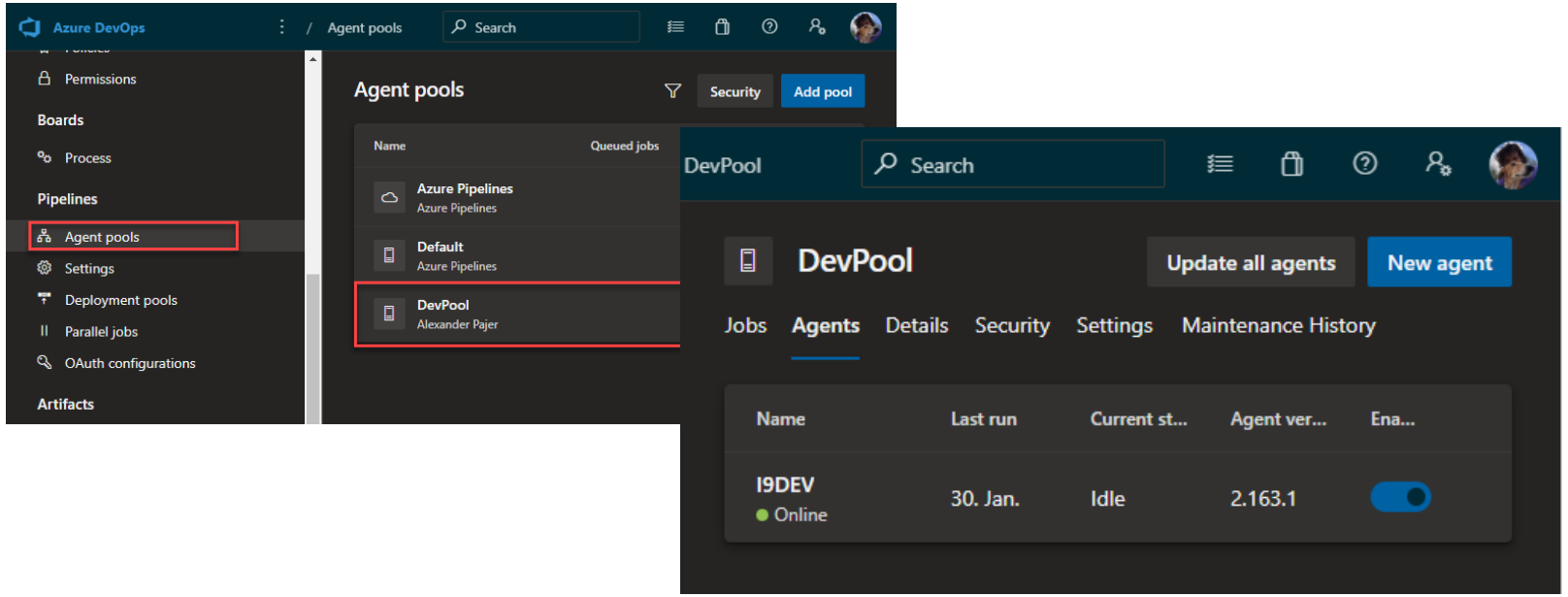
Other Considerations

- Authentication
  - To register an agent, you need to be a member of the administrator role in the agent pool
- Personal Access Tokens
  - Generate and use a PAT to connect an agent with Azure Pipelines
- Interactive vs Service processes
- Agent version and upgrades



Implementing a Hybrid (Self Hosted) Build Process

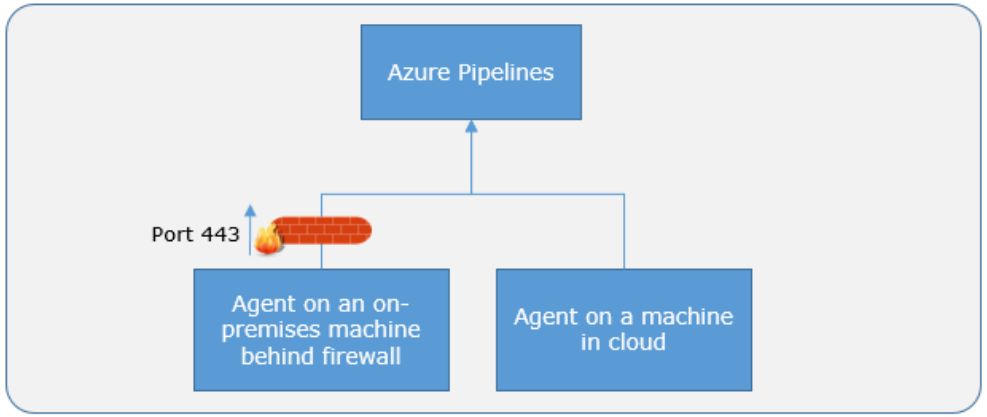
- Agents available for Windows, Linux, macOS
- Docker - Windows / Ubuntu Container
- Can be executed behind proxy using port 8888

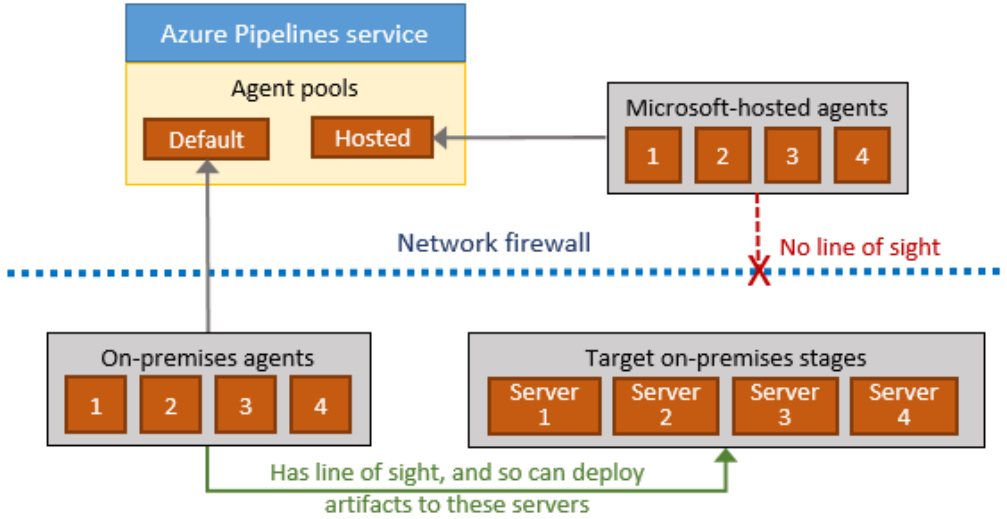


Hybrid Build Security

Why Hybrid?

- Security Issues (On-Prem),
- Special Demands
  - i.e. SharePoint Server Side Build -> Buildtools für Visual Studio

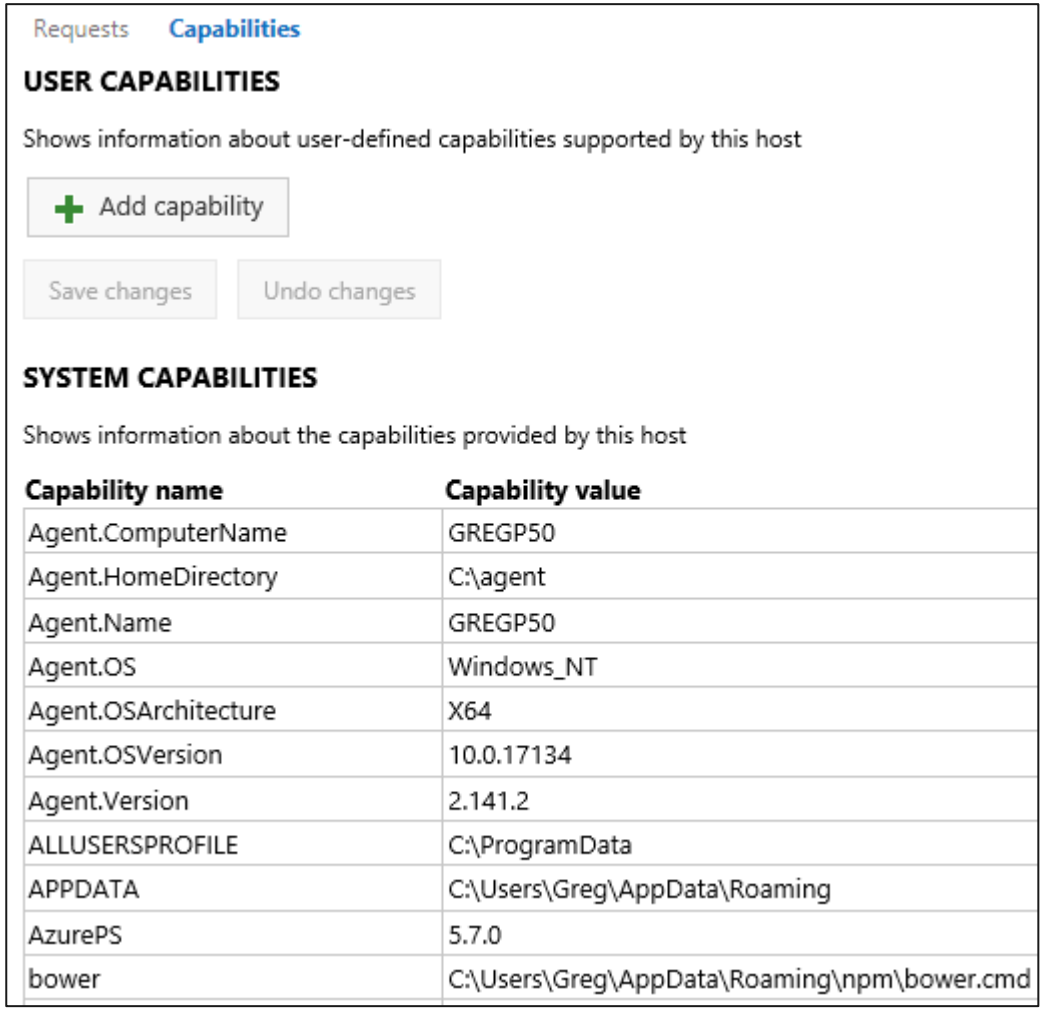




Configuring Agent Demands

Every self-hosted agent has a set of capabilities that indicate what it can do

- User Capabilities
- System Capabilities
- Agents can have different authorization and timeout settings



Capability name	Capability value
Agent.ComputerName	GREGP50
Agent.HomeDirectory	C:\agent
Agent.Name	GREGP50
Agent.OS	Windows_NT
Agent.OSArchitecture	X64
Agent.OSVersion	10.0.17134
Agent.Version	2.141.2
ALLUSERSPROFILE	C:\ProgramData
APPDATA	C:\Users\Greg\AppData\Roaming
AzurePS	5.7.0
bower	C:\Users\Greg\AppData\Roaming\npm\bower.cmd

Implementing Multi-Agent Builds

Adding multiple jobs to a pipeline lets you:

- Break your pipeline into sections that need different agent pools, or self-hosted agents
- Publish artifacts in one job and consume them in one or more subsequent jobs
- Build faster by running multiple jobs in parallel
- Enable conditional execution of tasks

✔ You can configure the number of parallel jobs

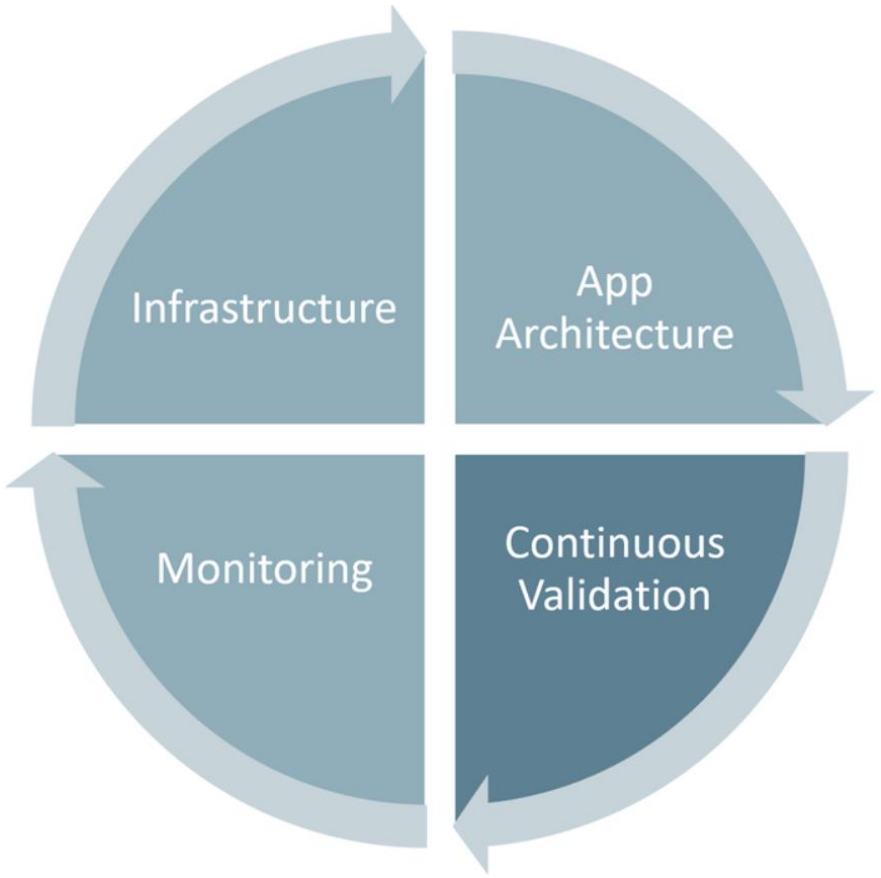
Microsoft

AZ-400.2 Module 02: Managing Code Quality and Security Policies

Lesson 01: Introduction to Security

Introduction to Security

- Securing applications is a continuous process that encompasses secure infrastructure, designing an architecture with layered security, continuous security validation, and monitoring for attacks
- Security is everyone's responsibility

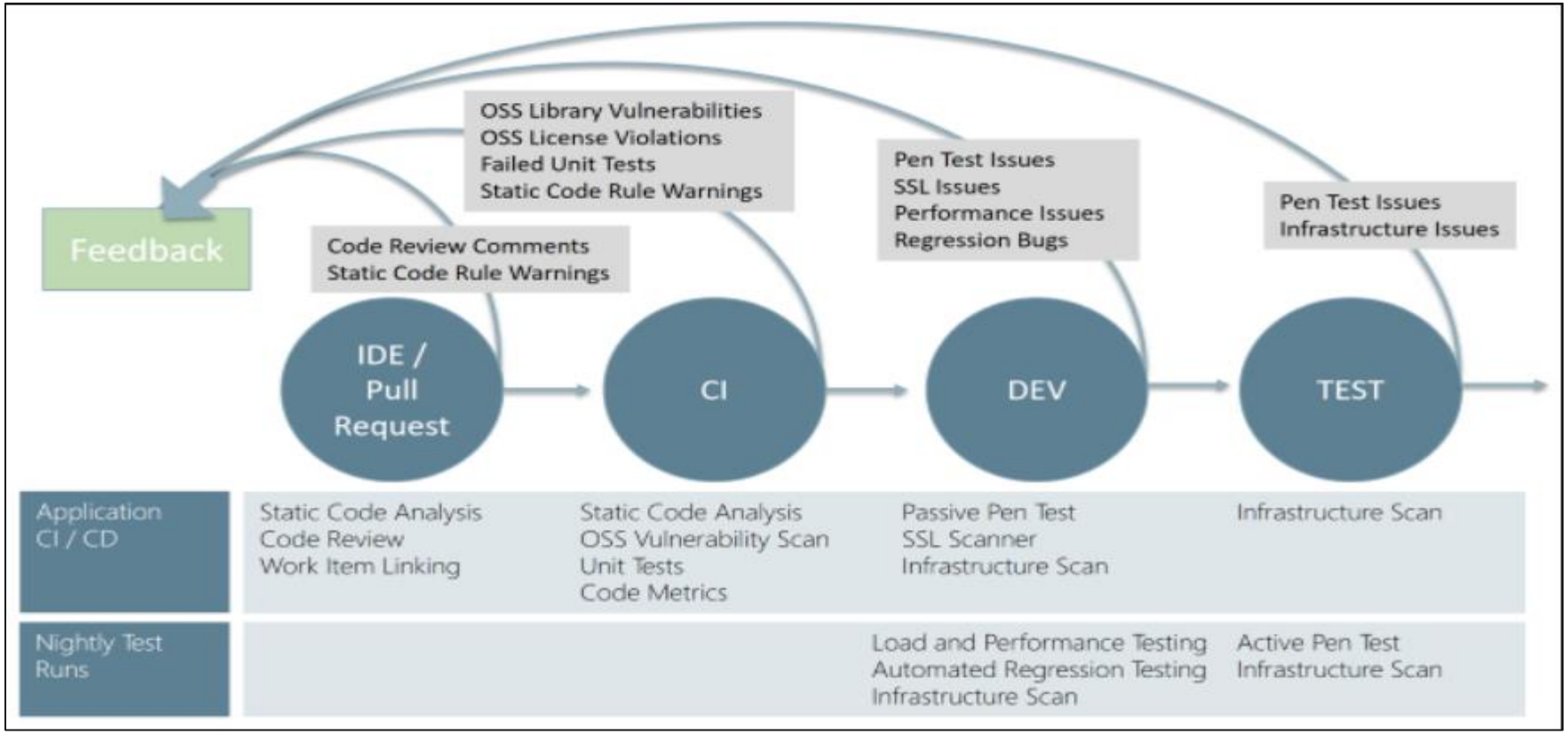


Continuous Integration - Security Testing

- The CI build should be executed as part of the pull request (PR-CI) process and once the merge is complete
- Several tools are available:
  - Visual Studio Code Analysis and the Roslyn Security Analyzers
  - SonarCloud - Bug & Vulnarablity Scanner
  - WhiteSource Bolt - Open Source Security Scanner
  - OWASP ZAP - Penetration Testing Tool

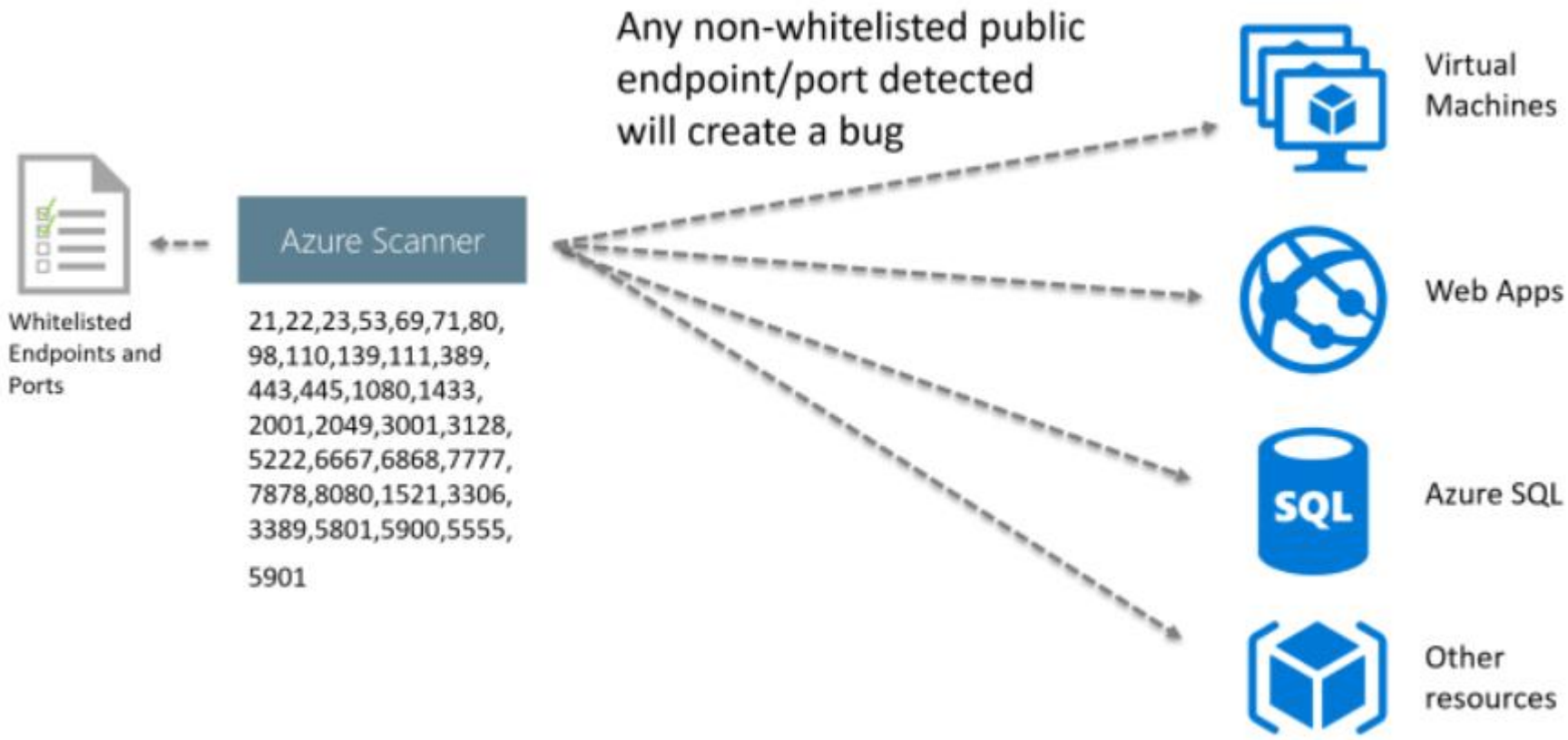
Key Validation Points

- Continuous security validation should be added at each step from development through production



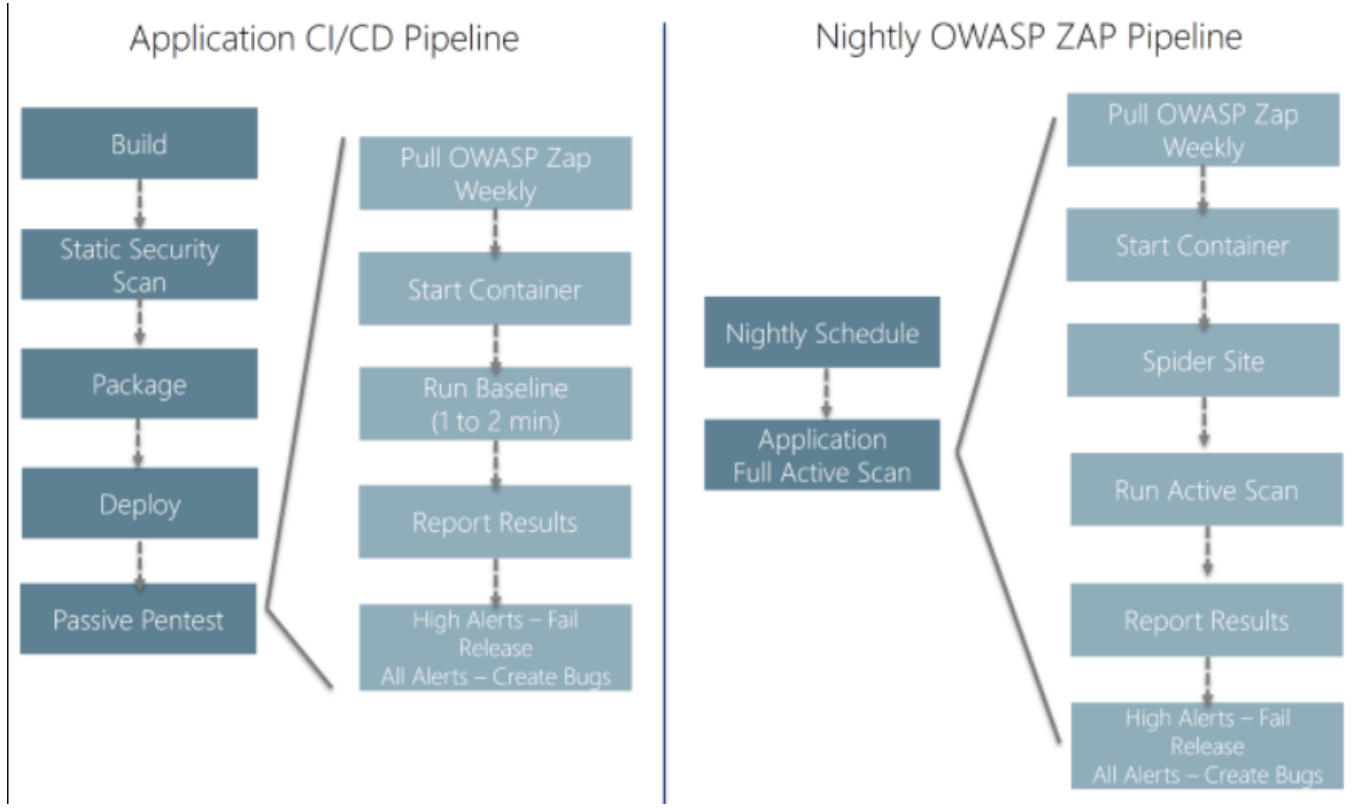
Infrastructure Vulnerabilities

- Be sure to validate the infrastructure
- Use the Azure Security Center and Azure Policies



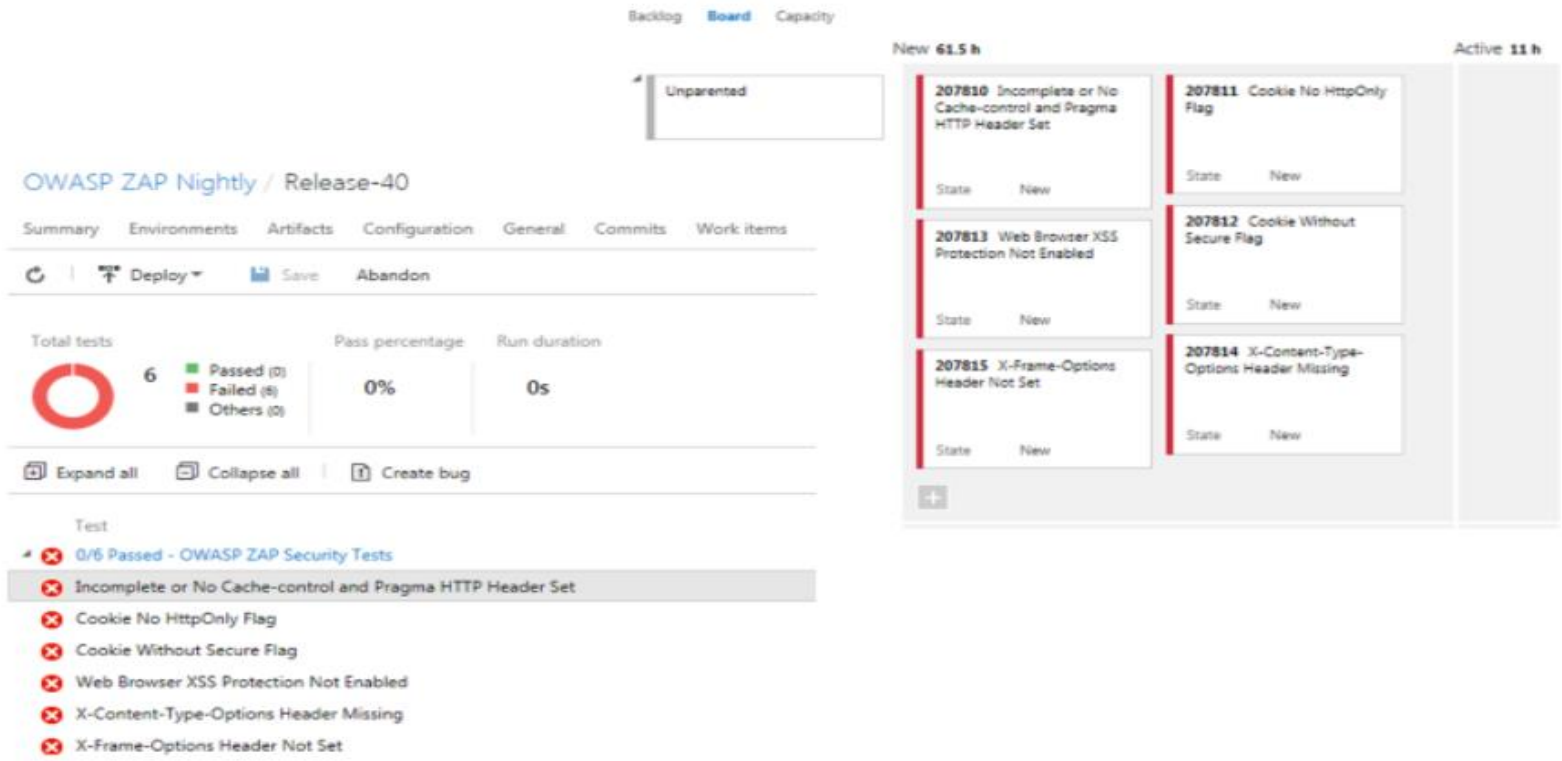
Application Deployment to DEV and TEST

- OWASP ZAP can be used for penetration testing
- Testing can be active or passive
- Conduct a quick baseline scan to identify vulnerabilities
- Conduct nightly more intensive scans



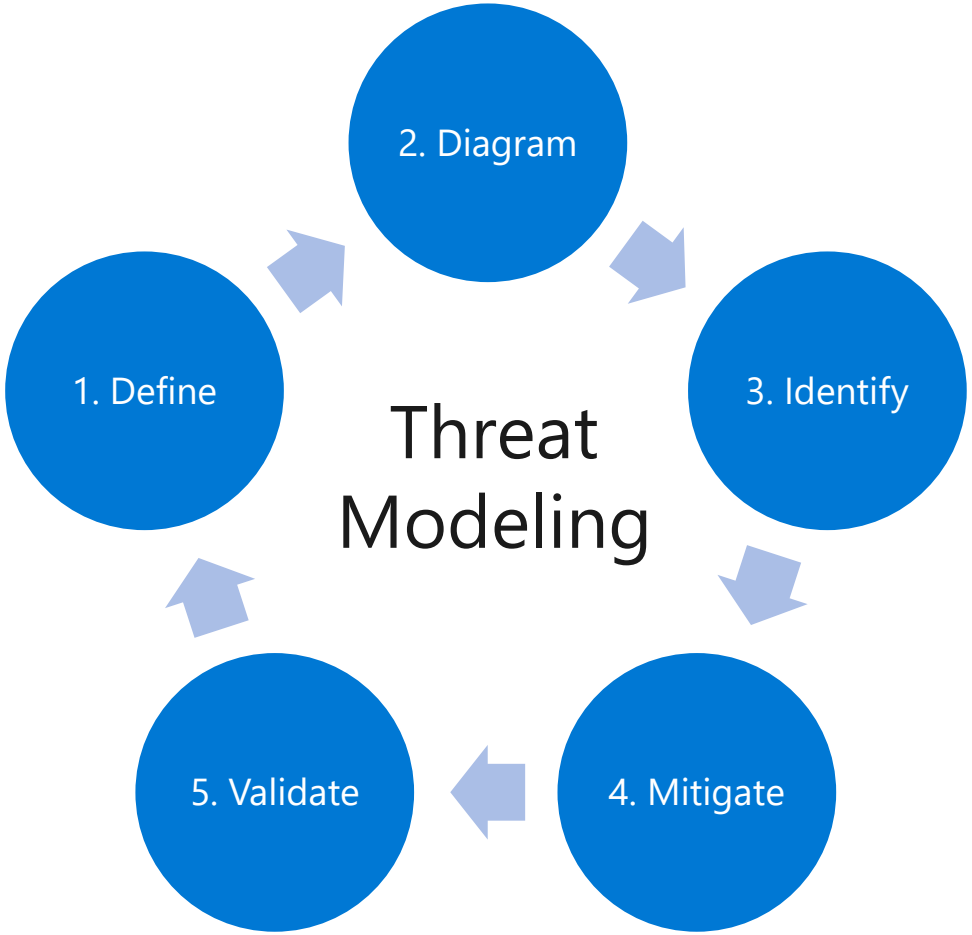
Results and Bugs

- OWASP ZAP provides a report with results and bugs
- Use a holistic and layered approach to security



Threat Modeling

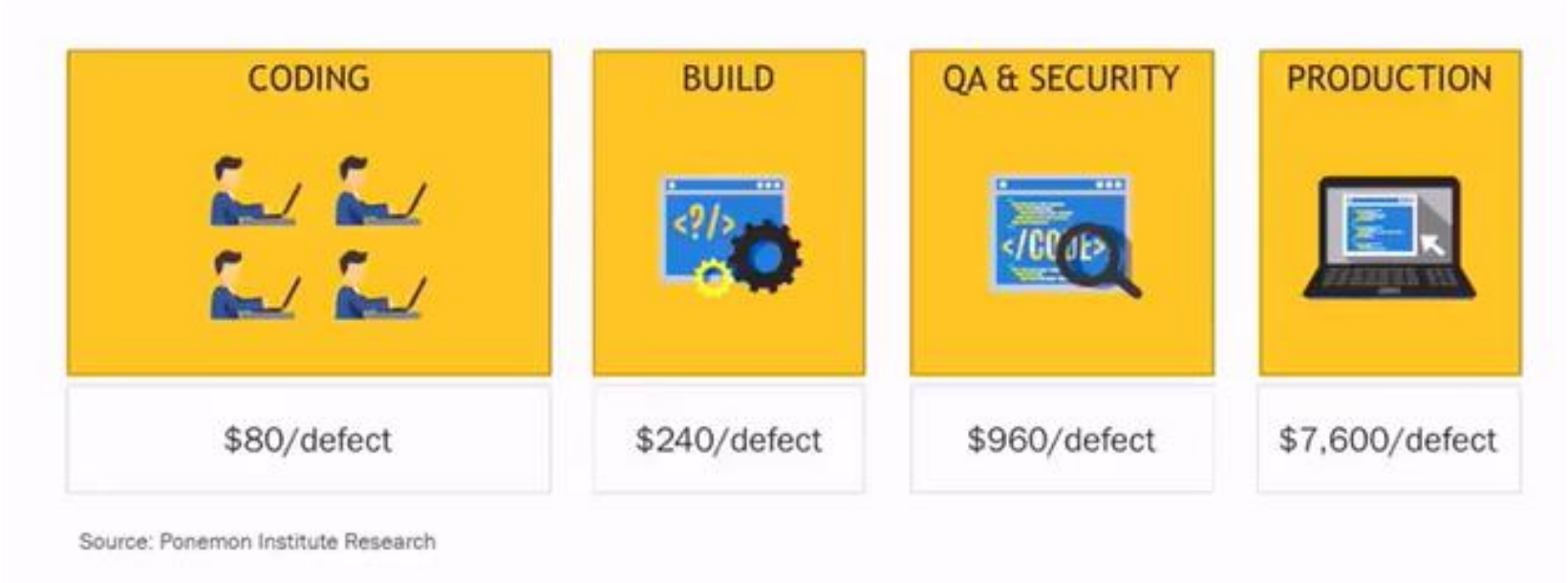
- Define security requirements
- Create an application diagram
- Identify threats
- Mitigate threats
- Validate that threats have been mitigated



Threat Modeling

Lesson 05: Implement Tools for Managing Security and Compliance

Implement Continuous Security Validation

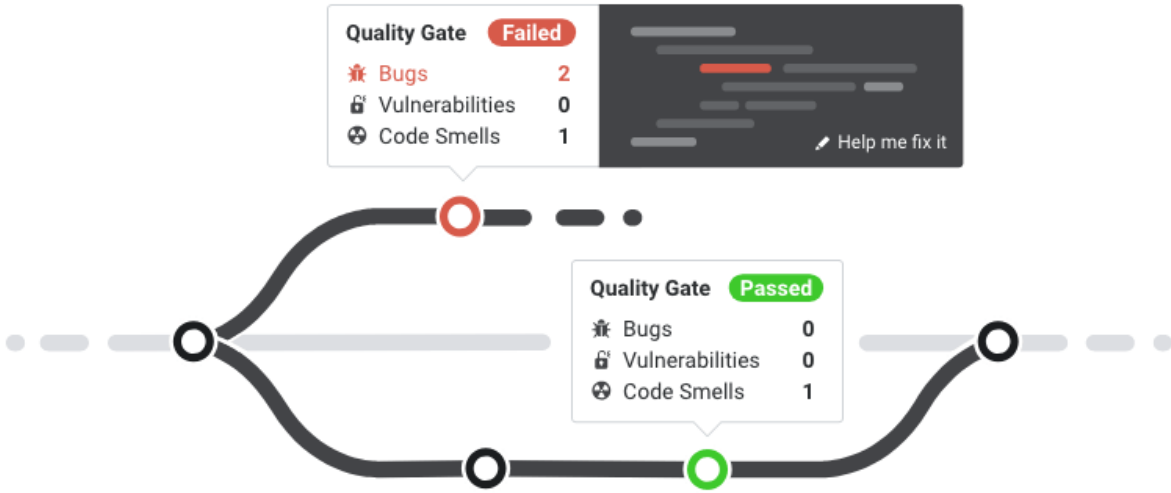


- Reduce cost by moving DevSecOps to the left
- Use automated tooling and processes to identify problems

SonarCloud vs SonarQube

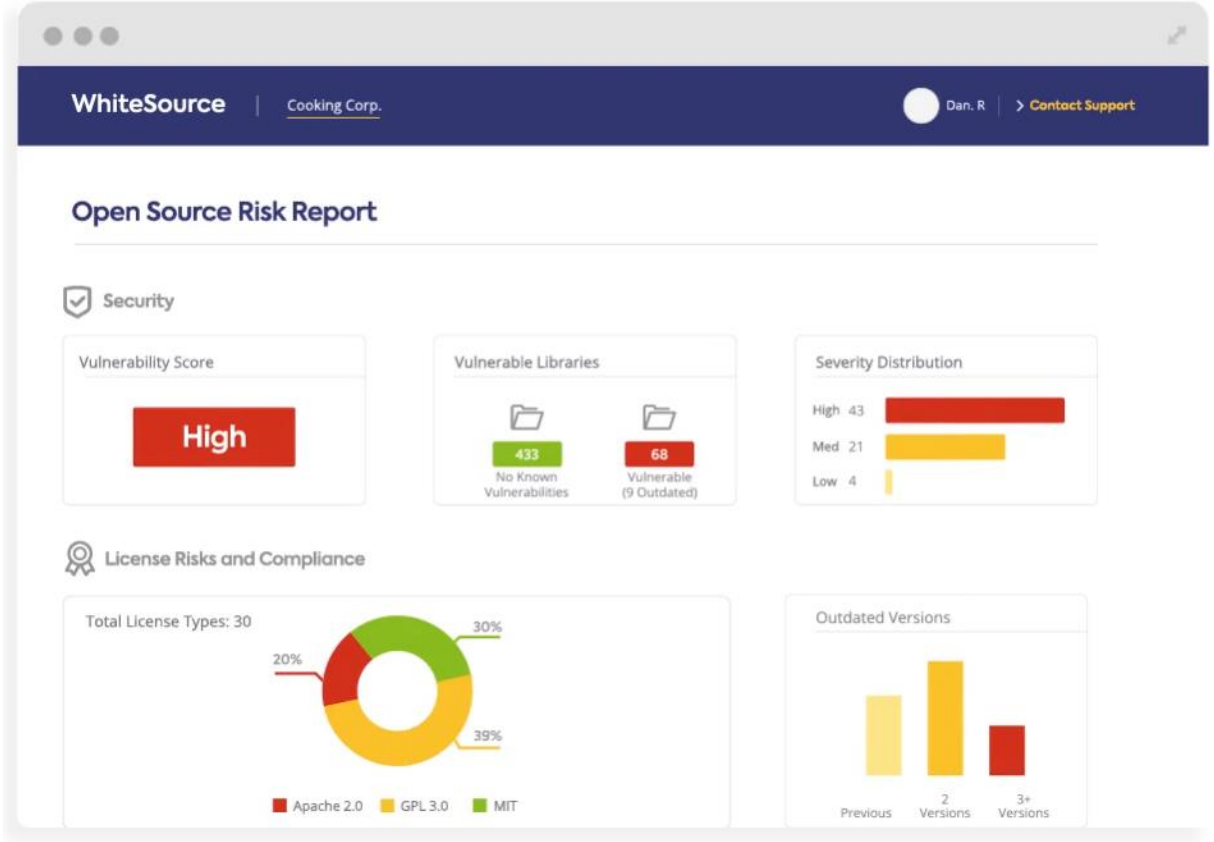
Technical debt – measure between the codebase's current state and an optimal state

- SonarQube is meant to be integrated with on-premise solutions like GitHub Enterprise or BitBucket Server for example
- SonarCloud is meant to be integrated with cloud solutions like Azure Dev Ops or BitBucketCloud



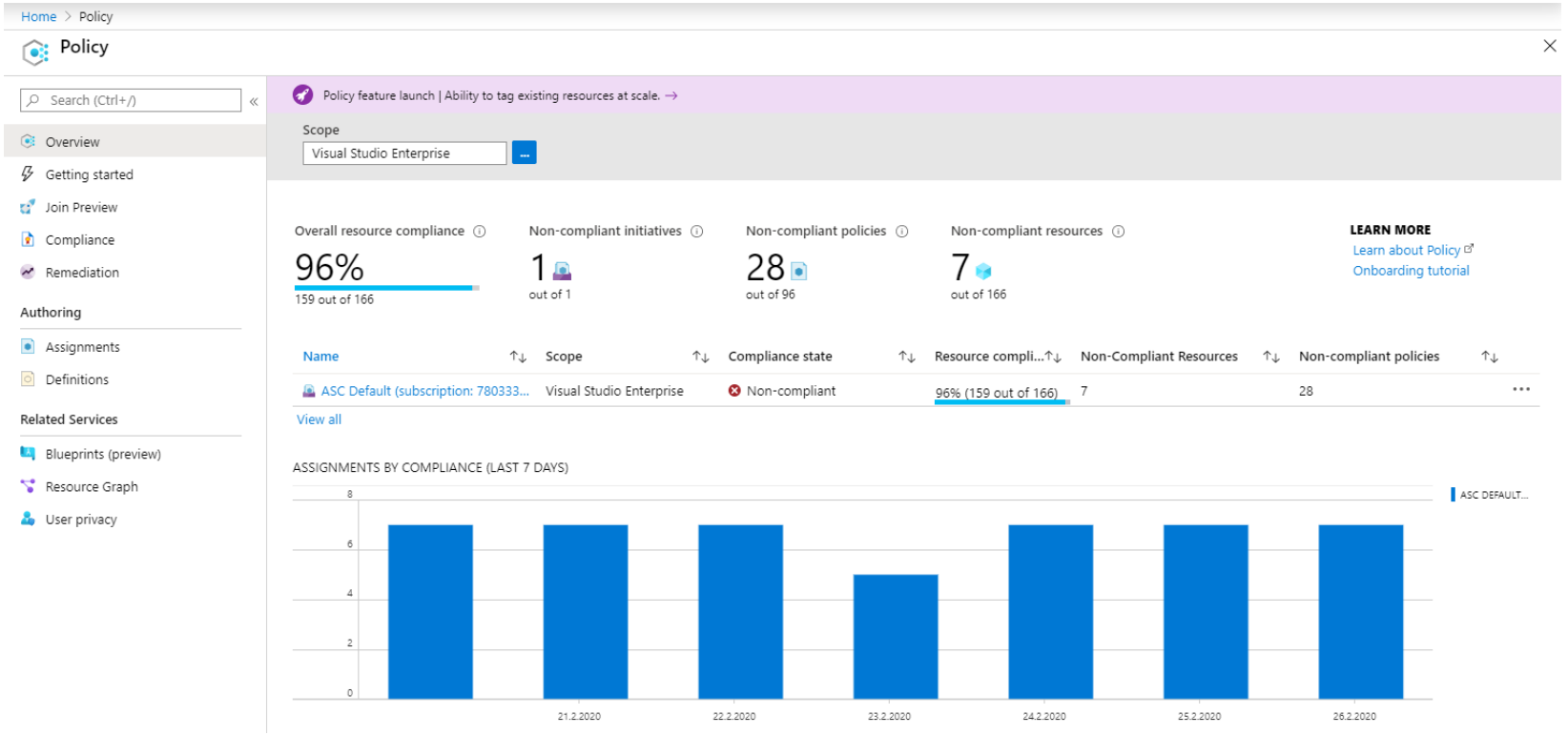
WhiteSouce

- Open Source Security Validation Tool

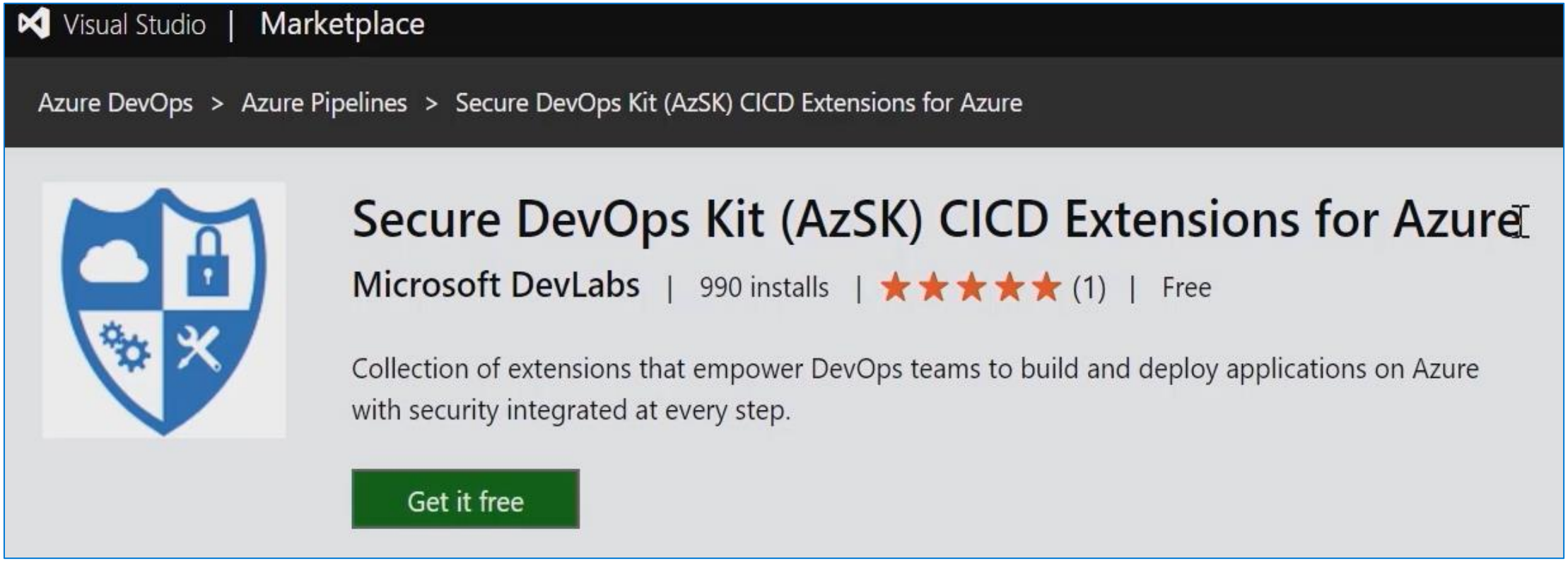


Azure Policy & Governance

- Enables enforcement restrictions and policies on ressources
- Used together with delegated administration



Securing Infrastructure with AzSK



Lab: SonarCloud

In this lab, Driving continuous quality of your code with SonarCloud, you will you will learn how to integrate Visual Studio Team Services with SonarCloud. You will learn how to:

- Setup a VSTS project and CI build to integrate with SonarCloud
- Analyze SonarCloud reports
- Integrate static analysis into the VSTS pull request process

✔ Note that you must have already completed the prerequisite labs in the Welcome section.

Lab: WhiteSource

In this lab, Managing Open-source security and license with WhiteSource, you can use WhiteSource Bolt with Azure DevOps to automatically detect alerts on vulnerable open source components, outdated libraries, and license compliance issues in your code. You will learn how to:

- Detect and remedy vulnerable open source components.
- Generate comprehensive open source inventory reports per project or build.
- Enforce open source license compliance, including dependencies' licenses.
- Identify outdated open source libraries with recommendations to update.

✔ Note that you must have already completed the prerequisite labs in the Welcome section.

Lesson 01: Managing Code Quality

Lesson 1 Overview

- Code Quality Defined
- Sources and Impacts of Technical Debt
- Using Automated Testing to Measure and Monitor Technical Debt
- Configuring SonarCloud in a Build Pipeline
- Reviewing SonarCloud Results and Resolving Issues
- Integrating Other Code Quality Tools
- Code Quality Tooling
- Managing Technical Debt with Azure DevOps and SonarCloud

Code Quality Defined

Short deadlines, a lack of coding standards, and poor technical skills can lead to code that is NOT:

- Clear and readable
- Documented
- Efficient
- Maintainable
- Extensible
- Secure

Code Quality Tools

Tool	What it is used for
White Source	Licence Checks
Black Duck	Checkt Open Source Code auf Risiko
Chef	Chef is a powerful automation platform that transforms virtual machine infrastructure on Azure into code.
Octopus Tentacle	Build Pipelines, automated deployment asp.net, Java, node.js on windows, Mac, Linux
Sonar Qube, Sonar Cloud	Insepct sourcecode, bugs, security, > 20 Sprachen
Apache Maven PMD	Code quality checks: Unused variable, empty catch block
Jira	Ticker System
Cobertura	Java: Code coverage testing and publish code
Bullseye coverage	BullseyeCoverage is an advanced C++ code coverage tool used to improve the quality
Coverlet	Coverlet is a cross platform code coverage framework for .NET
JaCoco	JaCoCo - Java Code Coverage Library
SourceGear Vault 10	SourceGear Vault Pro is a version control and bug tracking solution for professional development teams
OWASP ZAP	Owasp Zed Attack Proxy (ZAP): Security tool

Sources and Impacts of Technical Debt

- Technical Debt describes the future penalty that you incur today by making easy or quick choices in software development practices.
- Common sources of technical debt are:
  - Lack of coding style and standards, Lack of or poor design of unit test cases
  - Ignoring or not understanding object orient design principles
  - Monolithic classes and code libraries, Poorly envisioned use of technology, architecture and approach
  - Over-engineering code & Insufficient comments and documentation
  - Not writing self-documenting code
  - Taking shortcuts to meet deadlines
  - Leaving dead code in place

Using Automated Testing to Measure Technical Debt

Technical debt:

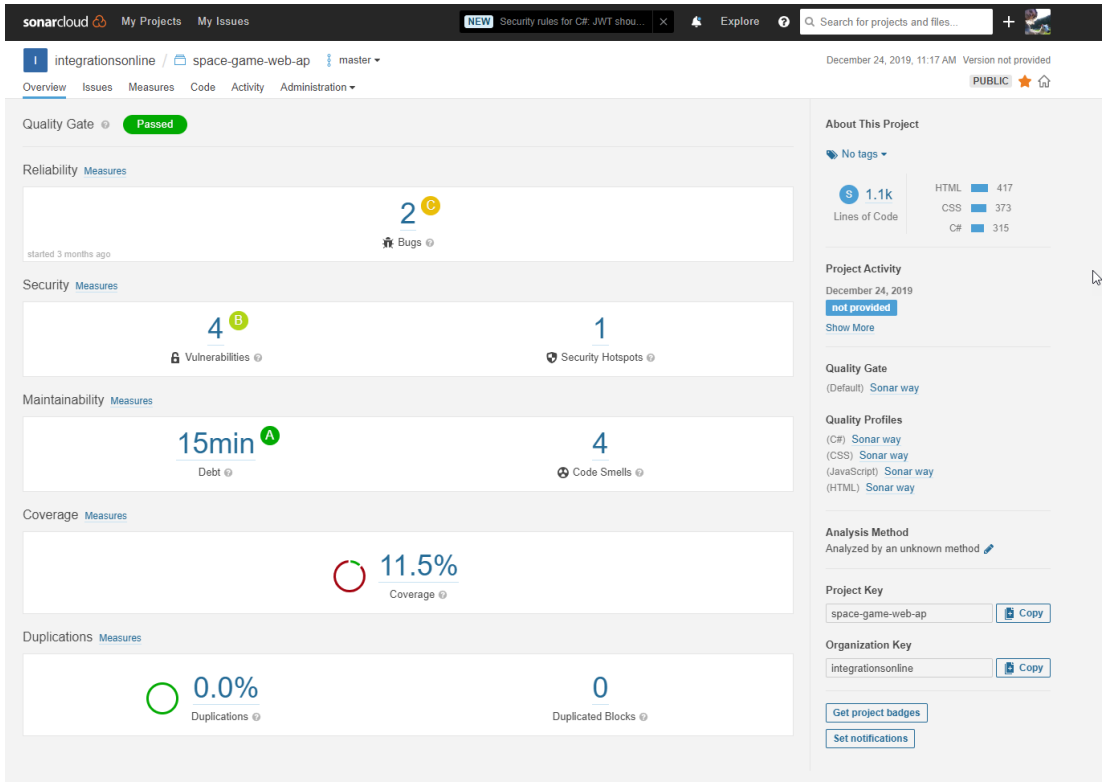
- Adds problems during development that makes it more difficult to add customer value
- Saps productivity and frustrates development teams
- Makes code both hard to understand and fragile
- Increases the time to make changes, and to validate those changes
- Starts small and grows over time

✔ One way to minimize the accumulation of technical debt, is to use automated testing and assessment

Sonar Cloud

Eliminate bugs and vulnerabilities.

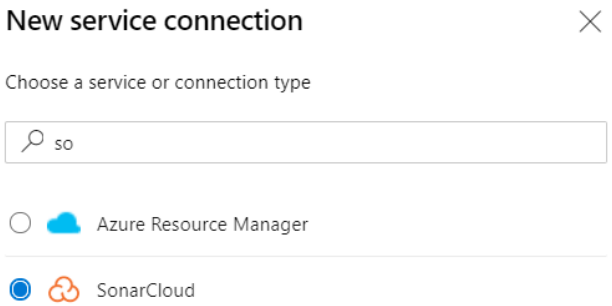
Champion quality code in your projects

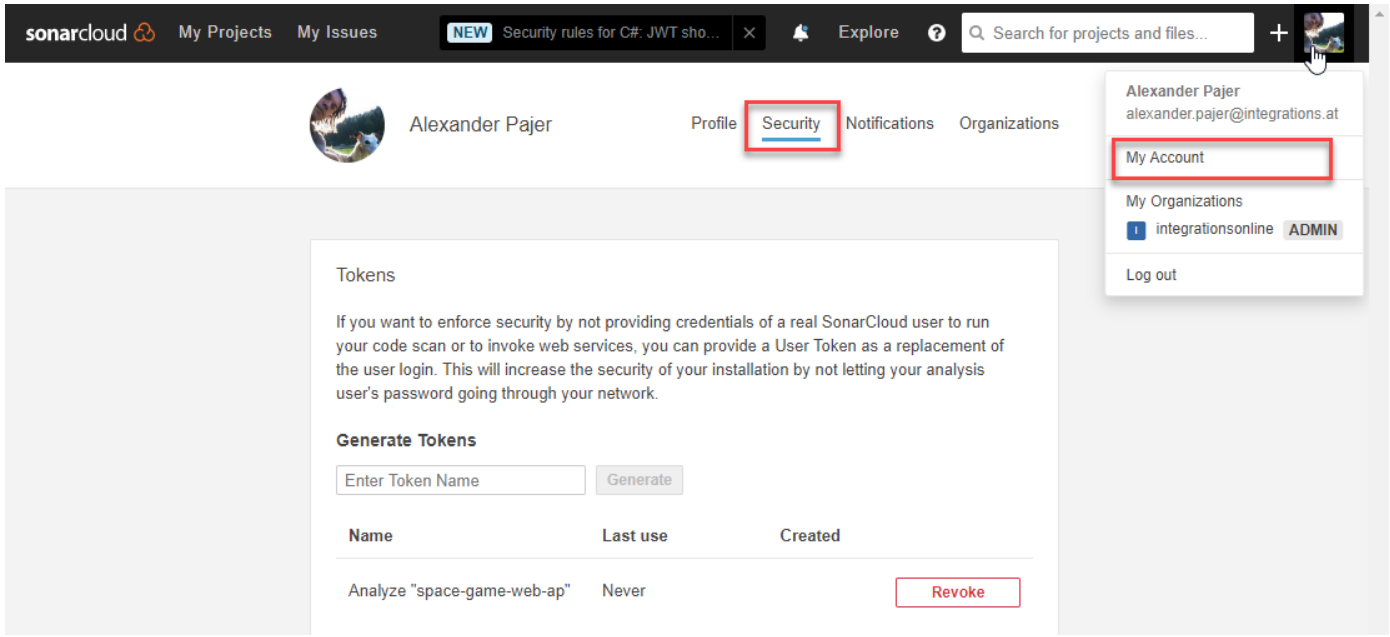


Sonar Cloud Service Connection

Requires:

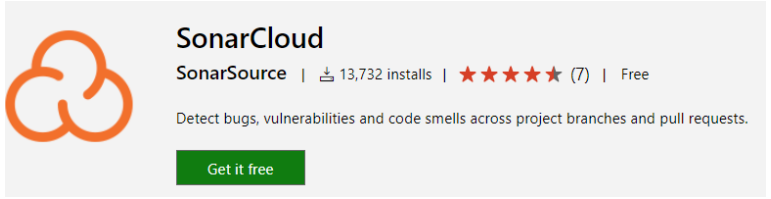
- Login Token generated in Sonarcloud
- Organ Name
- Project Key

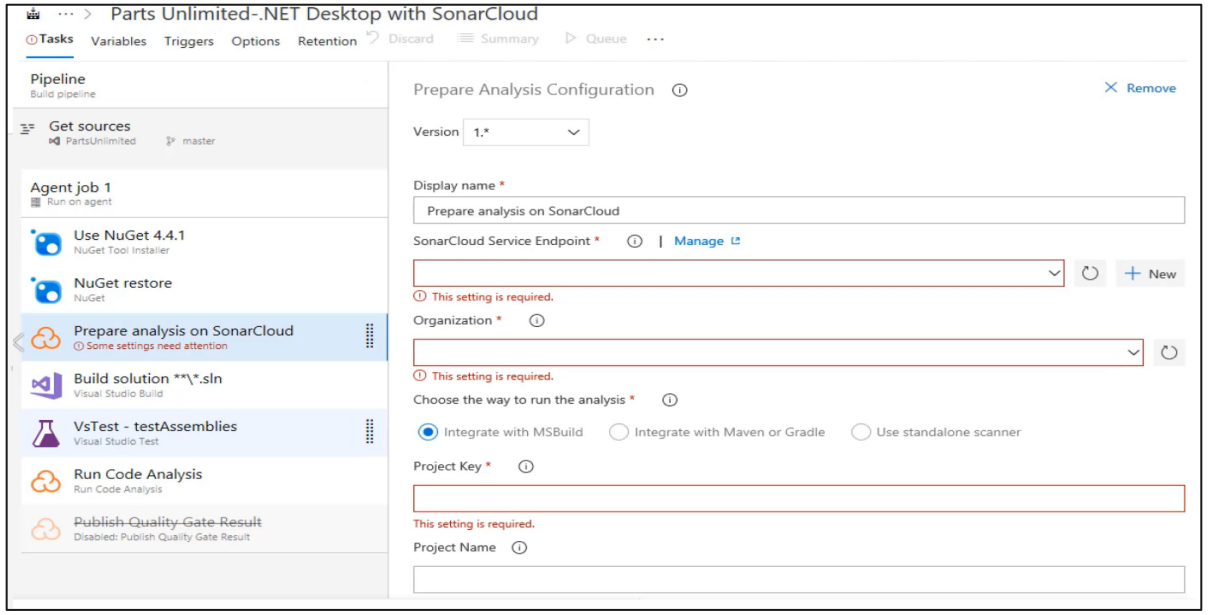




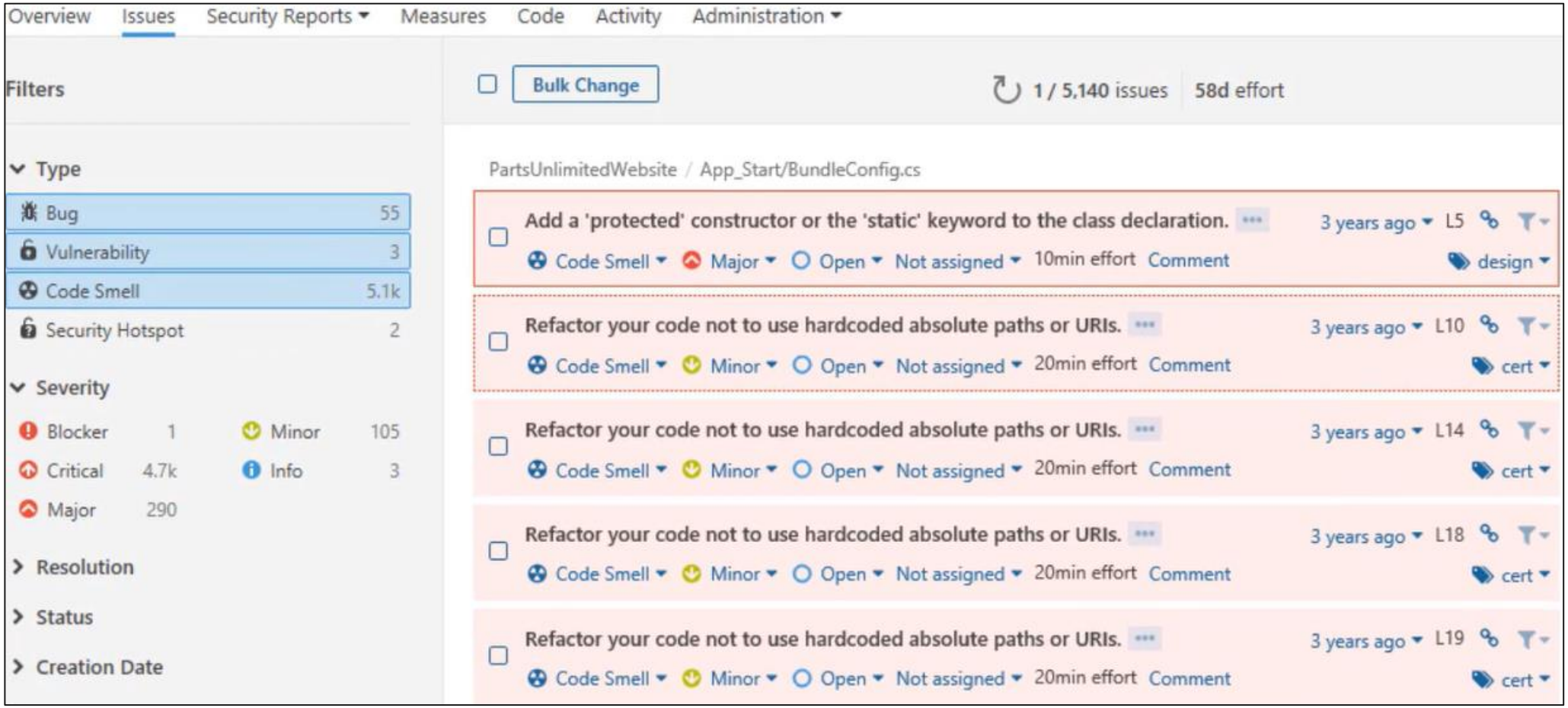
Configuring SonarCloud in a Build Pipeline

- Available as Marketplace Extension
- Requires a Service Connection
- Three Tasks:
  - Prepare Analysis Configuration
  - Run Code Analysis
  - Publish Quality Gate Result
- Typically stored in variables



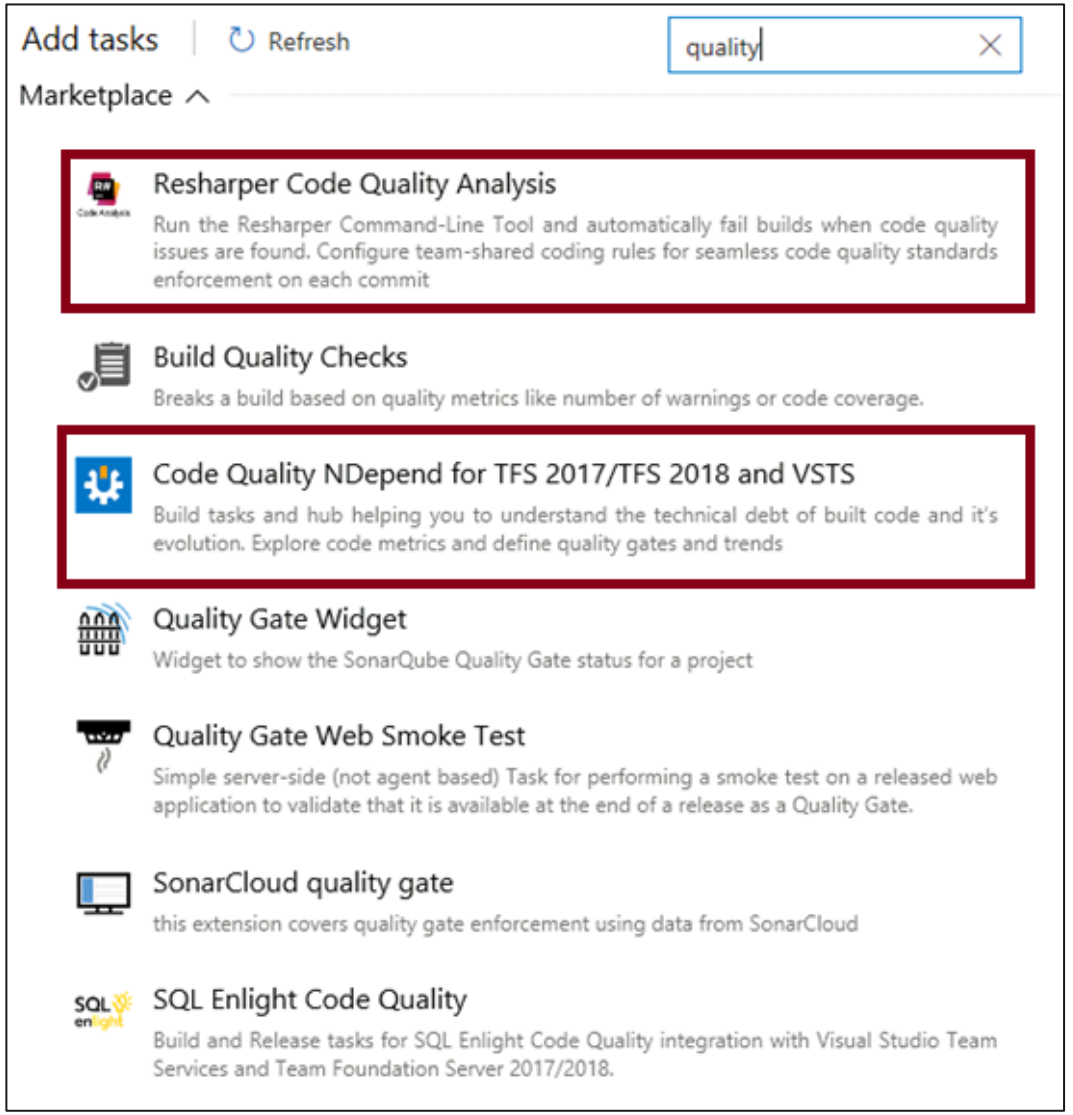


Reviewing SonarCloud Results and Resolving Issues



Integrating Other Code Quality Tools

- NDepend is a Visual Studio extension that assesses the amount of technical debt that a developer has added during a recent development period, typically in the last hour
- Resharper Code Quality Analysis is a command line tool and can be set to automatically fail builds when code quality issues are found



Lab: Managing Technical Debt with Azure DevOps and SonarCloud

In this hands-on lab, you will learn how to manage and report on technical debt using SonarCloud integration with Azure DevOps. You will perform the following tasks:

- Integrate SonarCloud with Azure DevOps and run an analysis
- Analyze the results
- Configure a quality profile to control the rule set used for analyzing your project

✔ Note that you must have already completed the prerequisite labs in the Welcome section.

Lesson 02: Managing Security Policies

Lesson 2 Overview

- Open Source Licensing Challenges
- Avoiding the OWASP Top Ten
- Detecting Open Source Issues with WhiteSource Bolt
- Integrating Other Security Policy Tooling
- Security Policy Tooling
- Checking Vulnerabilities using WhiteSource Bolt and Azure DevOps

Video: Open Source Licensing Challenges

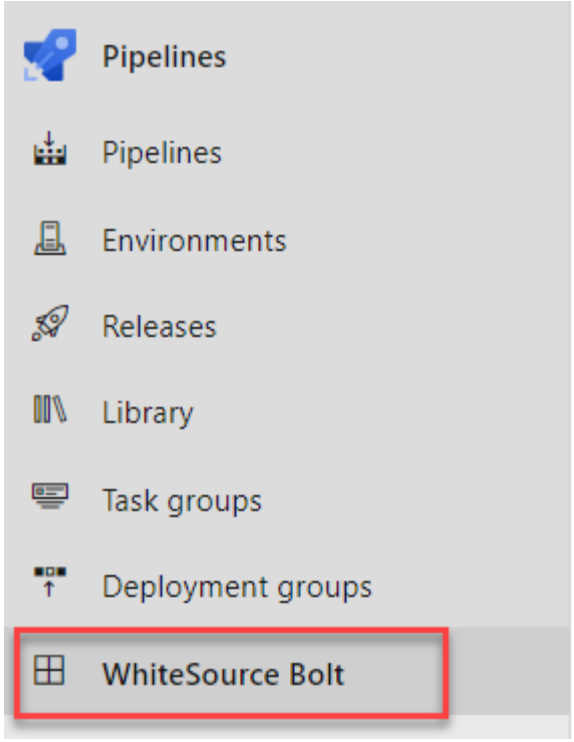
- Open source software is code that everyone can read, modify, enhance, and share
- Incorporating open source code is convenient but can cause issues:
  - Security
  - Quality
  - Old versions
  - Licensing
- Minimize risk by implementing automated systems to manage the code

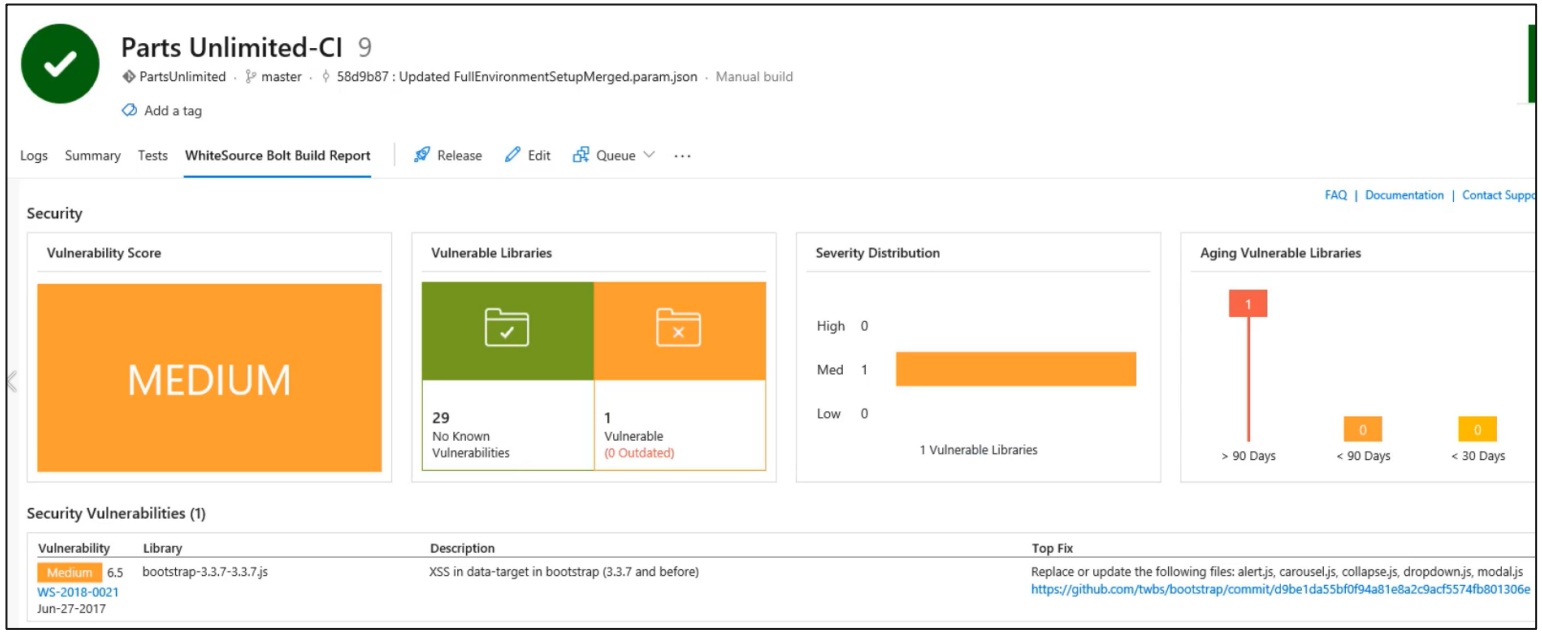
Video: Avoiding OWASP Top Ten

1. Injection Attacks
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control

Detecting Open Source Issues with WhiteSource Bolt

```
- task: WhiteSource Bolt@20
  displayName: 'Run WhiteSource Bolt'
```





Integrating Other Security Policy Tooling

- Micro Focus Fortify searches for violations of security-specific coding rules and guidelines
- Checkmarx CxSAST is designed for identifying, tracking and fixing technical and logical security flaws
- BinSkim is a static analysis tool that scans binary files
- OWASP Zed Attack Proxy Scan is an open-source web application for professional penetration testers

Lab: Checking Vulnerabilities using WhiteSource Bolt with Visual Studio Team Services

In this hands-on lab, you will learn how to check for open source vulnerabilities using WhiteSource Bolt in conjunction with Azure DevOps. You will learn how to:

- Integrate WhiteSource Bolt with you Azure DevOps build process
- Detect and remedy vulnerable open source components
- Generate comprehensive open source inventory reports per project or build
- Enforce open source license compliance, including licenses for dependencies
- Identify outdated open source libraries with recommendations to update

✔ Note that you must have already completed the prerequisite labs in the Welcome section.

Module 2: Review Questions

1. You want to run a penetration test against your application. Which tool could you use?
2. What is code smells? Give an example of a code smell.
3. You are using Azure Repos for your application source code repository. You want to create an audit of open source libraries that you have used. Which tool could you use?
4. Name three attributes of high-quality code.
5. You are using Azure Repos for your application source code repository. You want to perform code quality checks. Which tool could you use?

Microsoft

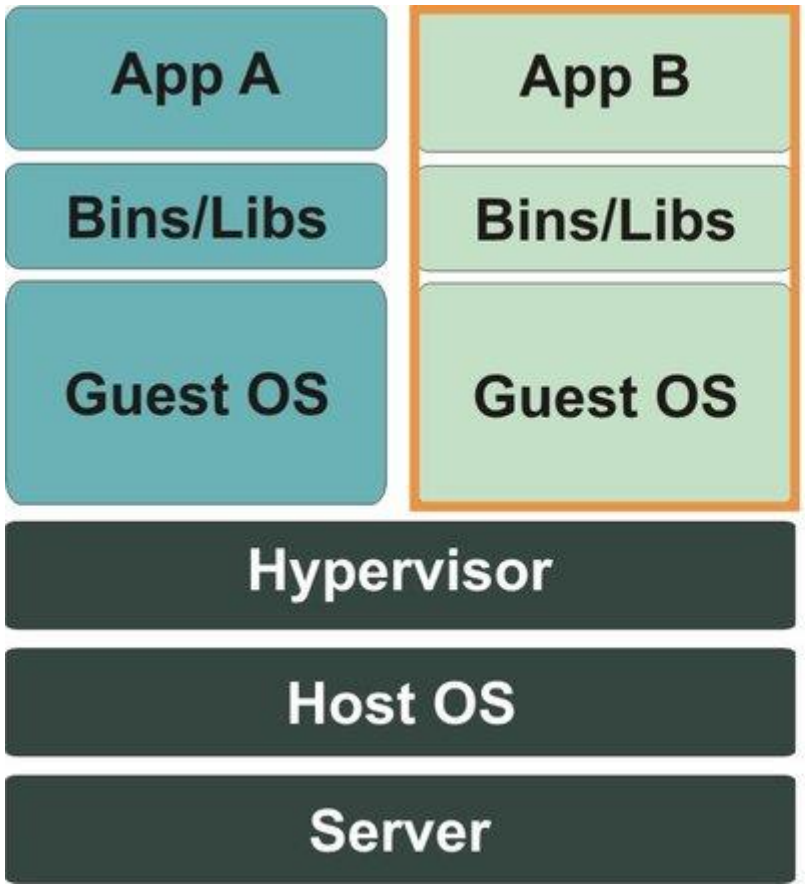
AZ-400.2 Module 03: Implementing a Container Build Strategy

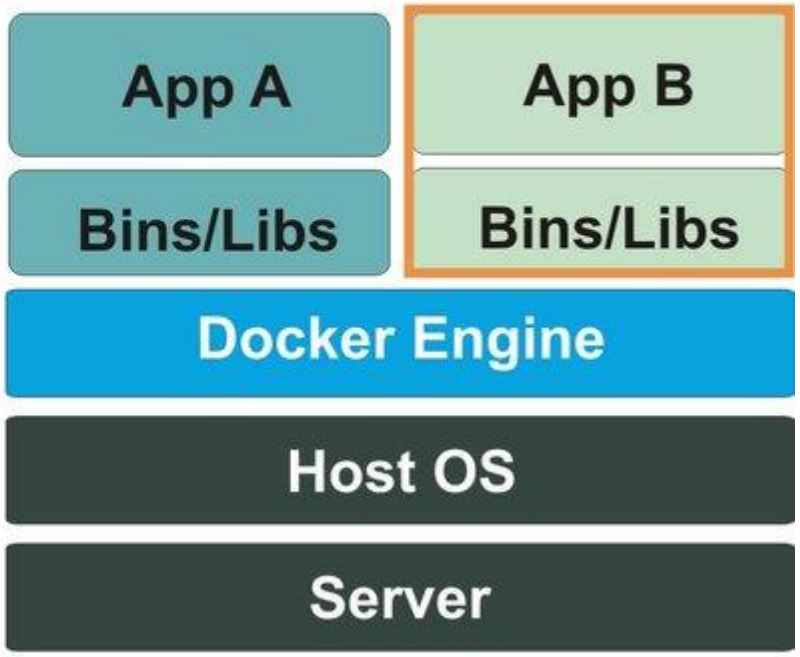
Lesson 01: Implementing a Container Build Strategy

Lesson 1 Overview

- Overview of Containers
- Containers vs Virtual Machines
- Docker Containers and Development
- Microservices and Containers
- Azure Container-Related Services
- Dockerfile Core Concepts
- Creating Multi-Stage Builds
- Creating an Azure Container Registry
- Adding Docker Support to an Existing Application
- Additional Container-Related Resources
- Modernizing an Existing .NET Application with Azure and Docker Images

Containers vs Virtual Machines





Discussion: Containers vs Virtual Machines

In your development environment,

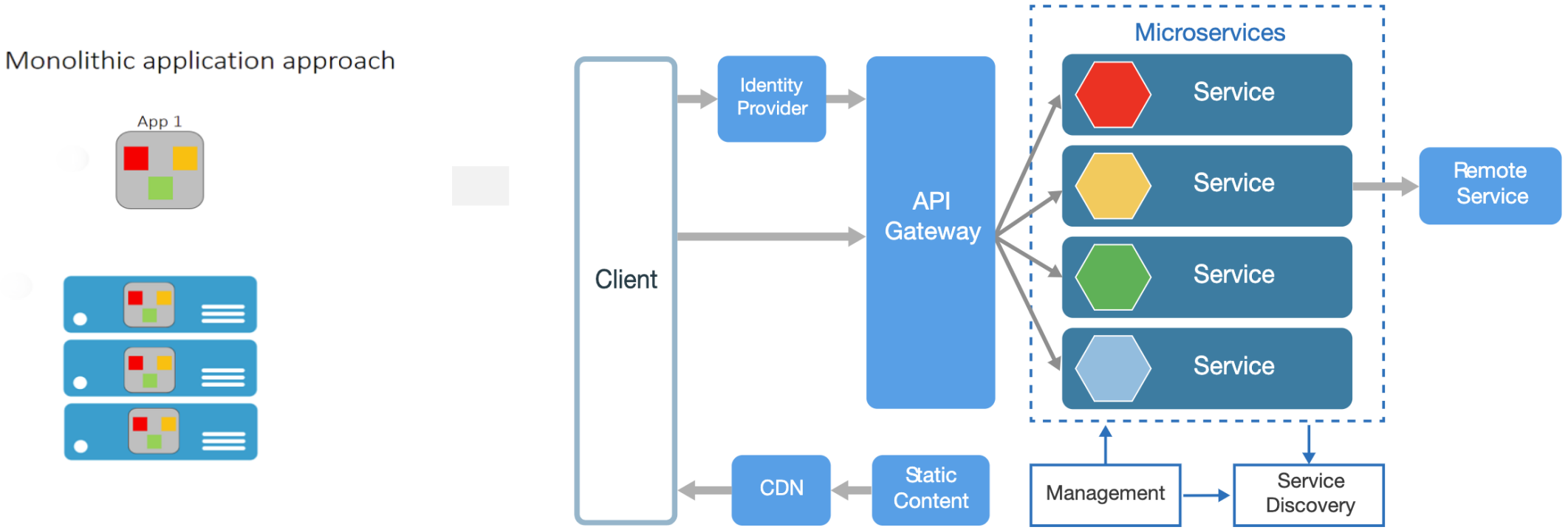
- Do you currently use virtualization of any type?
- Do you prefer to deploy containers or virtual machines?

Docker Containers and Development

- Docker is a software containerization platform with a common toolset, packaging model, and deployment mechanism
- Docker greatly simplifies containerization and distribution of applications that can be run anywhere
- A Docker image can be created that will deploy identically across any environment in seconds
- DockerHub has more than 180,000 applications in the public community repository
- Docker organized the Open Container Initiative (OCI)

Microservice Architecture

With Microservices every part of the application is deployed as a fully self-contained component



Azure Container Related Services

- Storing the Image:
- **Azure Container Registry** lets you store and manage container images in a central registry

- Hosting
- **Azure Container Instances** let you focus on creating your applications rather than provisioning and management of the infrastructure
- **Azure Kubernetes Service** is the de facto standard for container orchestration
- **Azure App Service p**rovides a managed service for both Windows and Linux based web applications

Dockerfile Core Concepts

- Text files that contain the commands needed by docker build to assemble an image
- Repeat the steps that are used to build the application
- Uses Keywords like:
  - FROM
  - WORKDIR
  - RUN
  - COPY
  - ENTRYPOINT
  - EXPOSE

```
FROM mcr.microsoft.com/dotnet/core/sdk:3.1
WORKDIR /app

COPY *.csproj ./
RUN dotnet restore

COPY . ./
RUN dotnet publish -c Release -o out
ENTRYPOINT ["dotnet", "out/SkillsApi.dll"]
EXPOSE 8080/tcp
ENV ASPNETCORE_URLS https://*:5000
```

Multiple Stage Builds

- Keep the image size as small as possible
- Layers are additional instructions added to the Dockerfile
- Multi-stage builds helps optimize the files, improves their readability, and makes them easier to maintain
- Each FROM instruction starts a new stage
- The stages are numbered in order, starting with stage 0
- Stages are named using an AS clause
- Naming stages lets you build them separately

Multistage Build

Uses 2 images

- Build Image with SDK (build)
- Runtime Image (base)

```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 8080/tcp
ENV ASPNETCORE_URLS https://*:5000

FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
WORKDIR /src
COPY ["*.csproj", "."]
RUN dotnet restore "SkillsApi.csproj"
COPY . .
RUN dotnet build "SkillsApi.csproj" -c Release -o /app

FROM build AS publish
RUN dotnet publish "SkillsApi.csproj" -c Release -o /app
FROM base AS final
WORKDIR /app
COPY --from=publish /app .
ENTRYPOINT ["dotnet", "SkillsApi.dll"]
```

Create an Azure Container Registry

Container Registry

Microsoft

Azure Container Registry is a private registry for hosting container images. Using the Azure Container Registry, you can store Docker-formatted images for all types of container deployments. Azure Container Registry integrates well with orchestrators hosted in Azure Container Service, including Docker Swarm, DC/OS, and Kubernetes. Users can benefit from using familiar tooling capable of working with the open source Docker Registry v2.

Use Azure Container Registry to:

- Store and manage container images across all types of Azure deployments
- Use familiar, open-source Docker command line interface (CLI) tools
- Keep container images near deployments to reduce latency and costs
- Simplify registry access management with Azure Active Directory
- Maintain Windows and Linux container images in a single Docker registry

Lab: Existing .NET Applications with Azure and Docker Images

In this hands-on lab, you will learn how to modernize an existing ASP.NET application with migration to Docker images managed by the Azure Container Registry. You will learn how to:

- Migrate the LocalDB to SQL Server in Azure
- Using the Docker tools in Visual Studio 2017, add Docker support for the application
- Publish Docker Images to Azure Container Registry (ACR)
- Push the new Docker images from ACR to Azure Container Instances (ACI)

Module 3: Review Questions

1. You are reviewing an existing Dockerfile. How would you know if it's a multi-stage Dockerfile?
2. You are designing a multi-stage Dockerfile. How can one stage refer to another stage within the Dockerfile?
3. What is the line continuation character in Dockerfiles?
4. You are using Azure to manage your containers. Which container orchestration styles are supported?
5. When the Open Container Initiative defined a standard container image file format, which format did they choose as a starting point?