

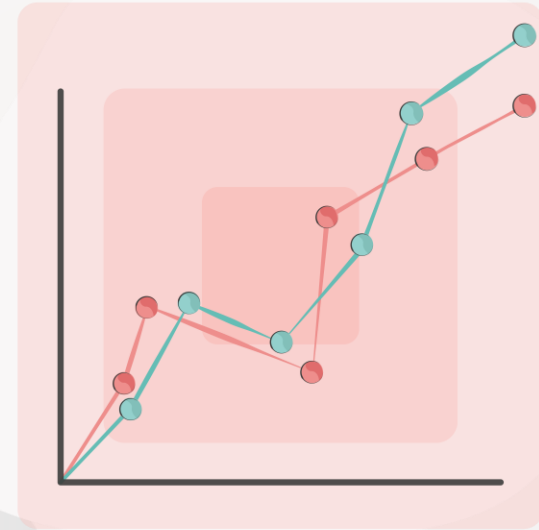


# **Level 5 Data Engineer**

## **Module 5 Topic 4**

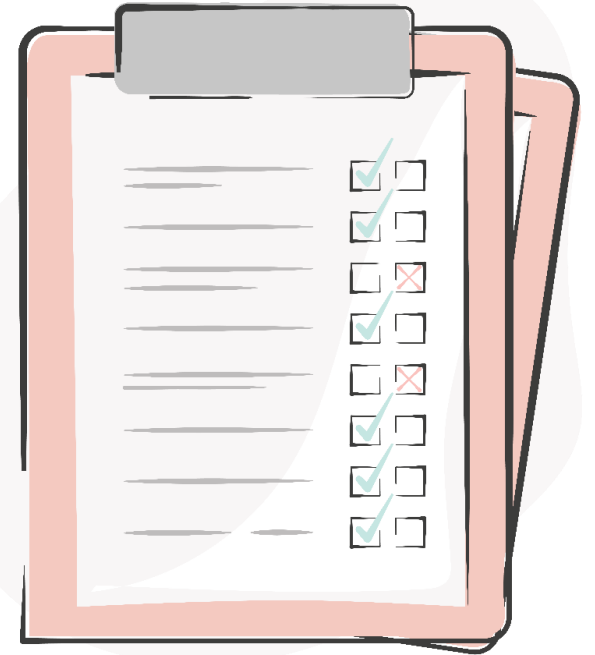
### **Securely deploying cloud data products**

**Welcome to today's  
webinar.**

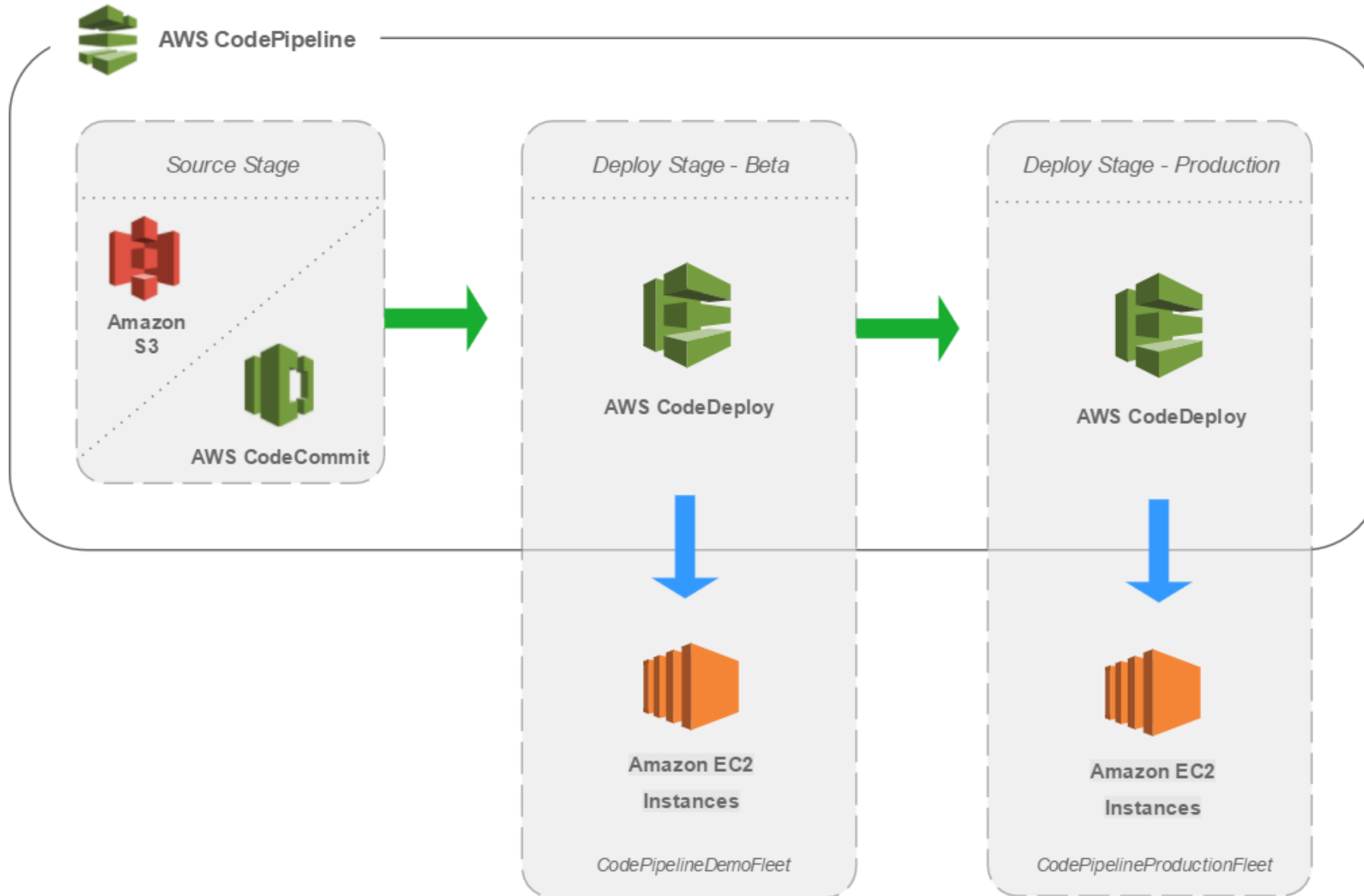


# What we will cover in this topic

- Examine the importance of being able to release to cloud users on demand
- Experiment with your own deployment in the cloud
- Appraise CI/CD best practices
- Examine a DevSecOps Toolchain
- Distinguish the tools required at various stages of the DevSecOps delivery
- Compare Azure security approaches to AWS security approaches



# Release pipelines



# Artifact sources

- Build artifacts
- Package repositories
- Container repositories
- Files
- Source control



# Considerations for deployment to stages

- Do we want/need to deploy every day?
- What is your target environment?
- Is it used by one team or is it used by multiple teams?
- Who are the users? Do they want a new version multiple times a day?
- How long does it take to deploy?
- Is there downtime? What happens to performance? Are users impacted?

# Delivery cadence – three types of triggers

1. Continuous deployment trigger
2. Scheduled trigger
3. Manual trigger

# Release approvals

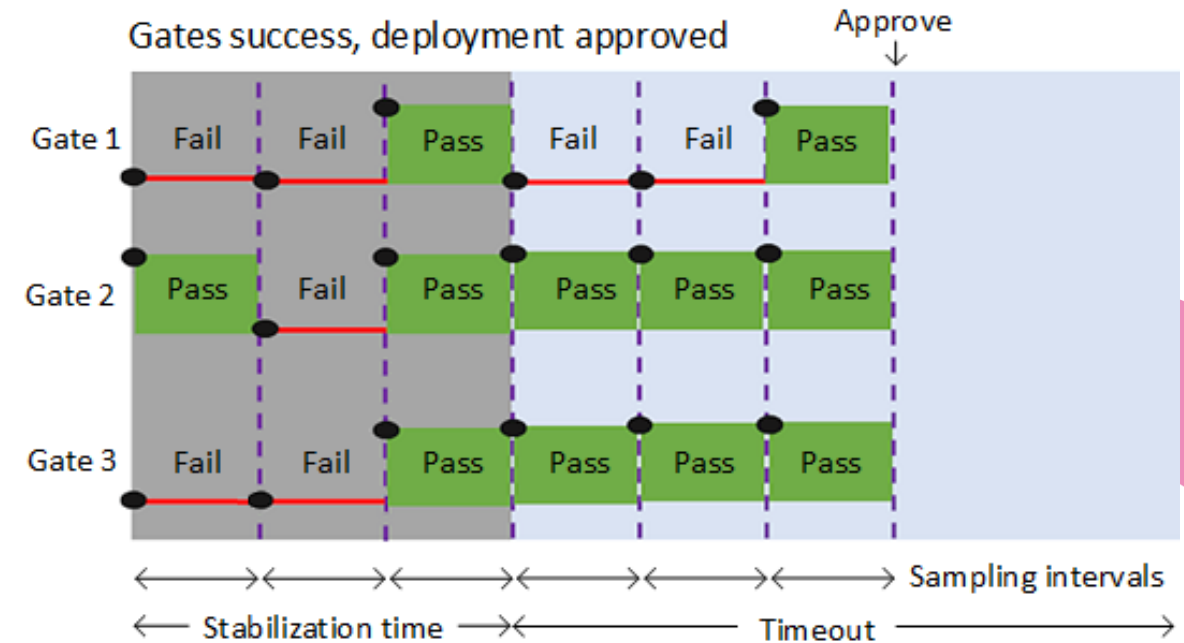
- Release approvals are not to control *\*how\**, but control *\*if\** you want to deliver multiple times a day
- Manual Approvals help in building trust about the automated release process
- Release gates give you additional control over the start and completion of the deployment pipeline. They can usually be set up as a pre-deployment and post-deployment condition and can perform validation with other automated systems until specific requirements are verified.

# Release gates

Release gates give you additional control over the start and completion of the deployment pipeline.

They are often set up as a pre-deployment and post-deployment conditions.

- Incident and issues management
- Notification of users by integration with collaboration systems
- Quality validation
- Security scan on artifacts
- User experience relative to baseline
- Change management
- Infrastructure health





# Using release gates to protect quality

- No new blocker issues
- Code coverage on new code greater than 80%
- No license violations
- No vulnerabilities in dependencies
- No new technical debt introduced
- Compliance checks
- Are there work items linked to the release?
- Is the release started by someone else as the code committer?
- Is the performance not affected after a new release?



# Release process versus release

- The release process involves all the steps that you go through when you move your artifact that comes from one of the artifact sources.
- A release is a package or container that holds a versioned set of artifacts specified in a release pipeline in your CI/CD process. It holds all the information required to carry out all the tasks and actions in the release pipeline, such as the stages (or environments), the tasks for each one, values of task parameters and variables, the release policies such as triggers, approvers, and release queuing options.
- There can be multiple releases from one release pipeline (or release process)

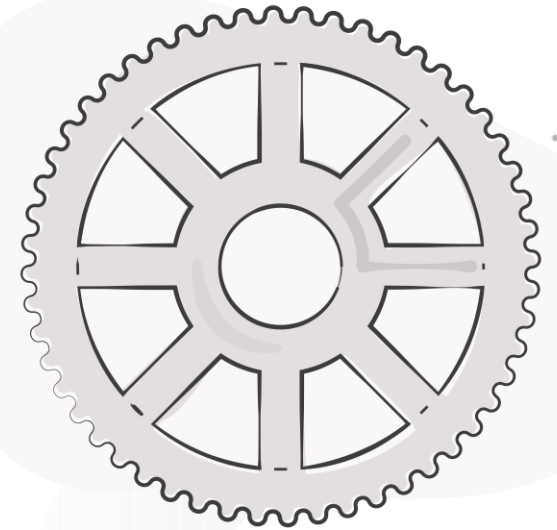
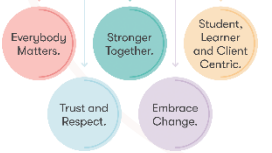
# Azure DevOps release pipelines

- Support pipelines as code (via YAML)
- Most capabilities from classic release pipelines are available

# Activity

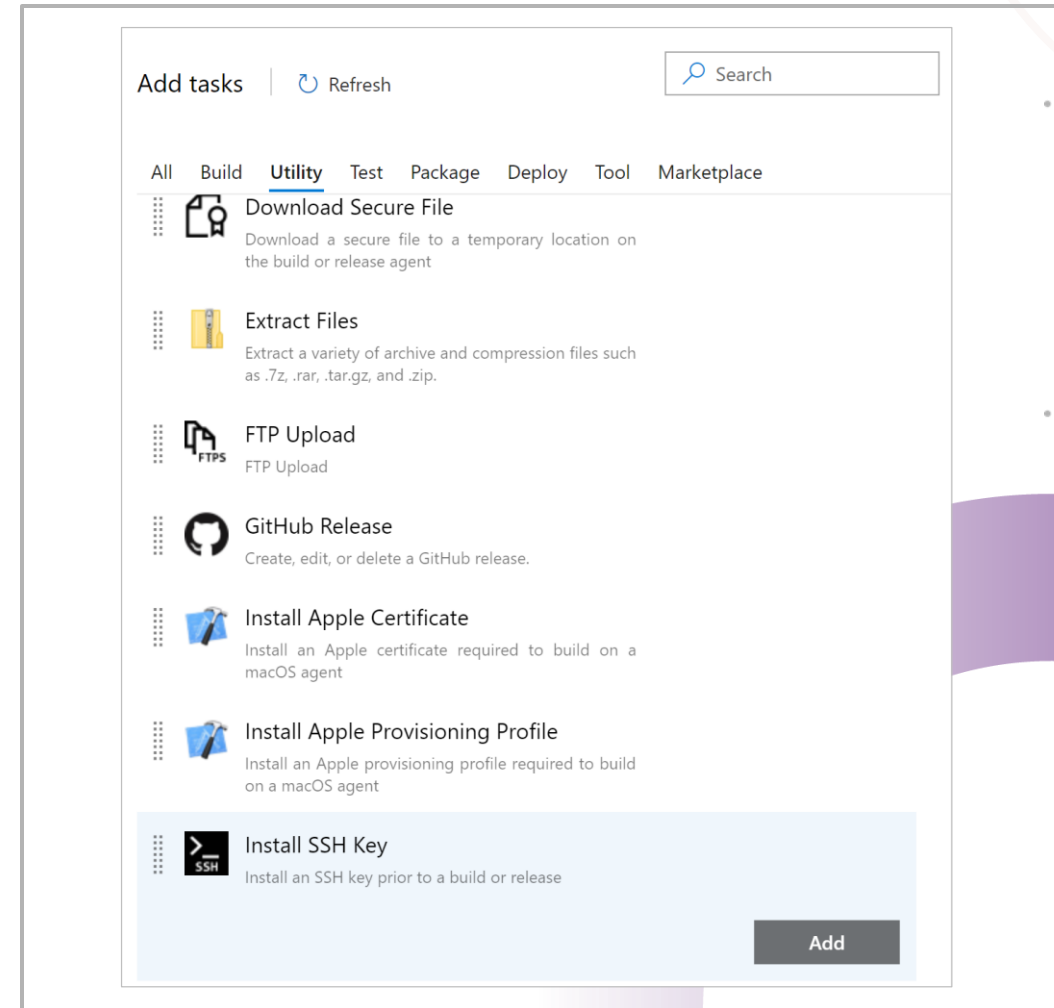
See document “**L5DE M5T4 Activity** - Configuring Agent Pools and Understanding Pipeline Styles “

Building Careers  
Through Education



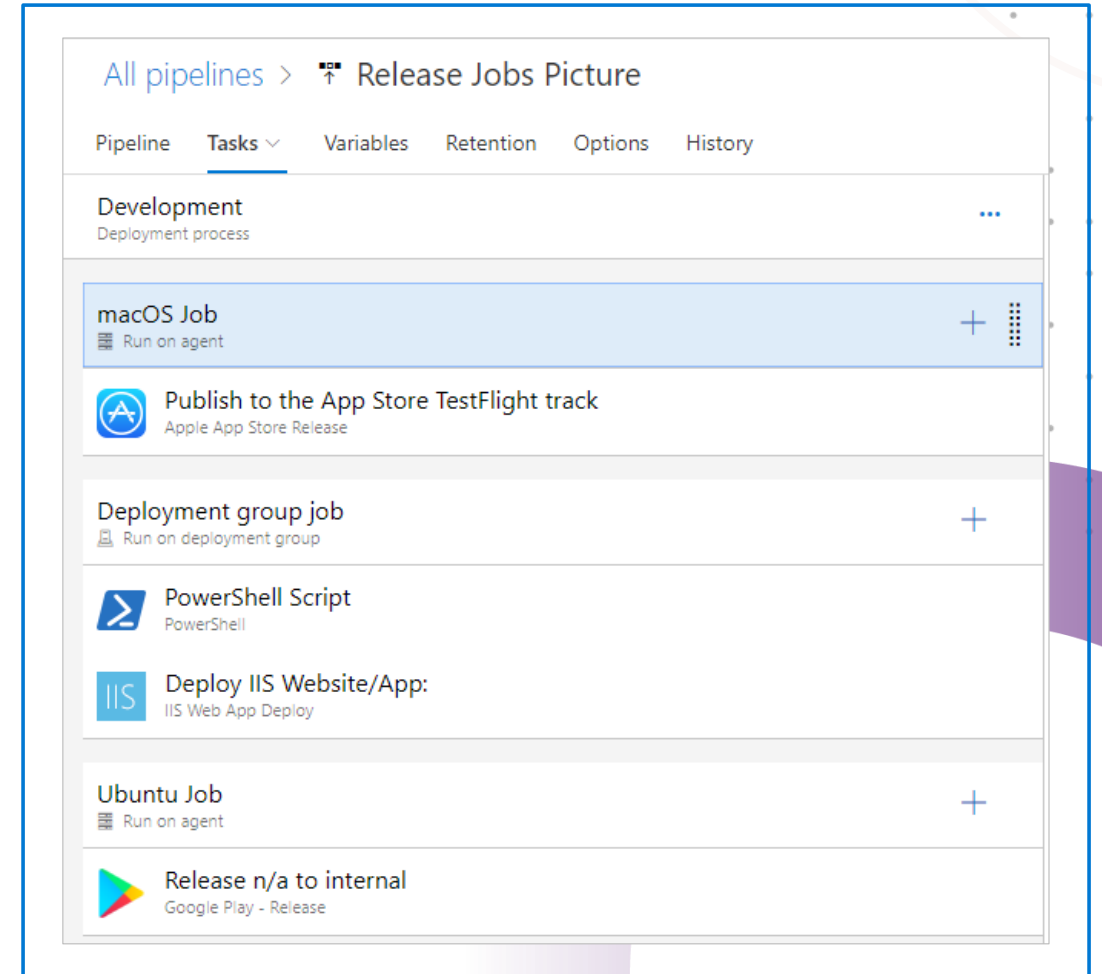
# Build and release tasks

- Units of executable code used to perform designated actions in a specified order
- Tasks building, testing, running utilities, packaging, and deploying
- Extensible model
- Community tasks available in marketplace



# Release jobs

- A job is a series of tasks that run sequentially on the same target
- Can be combined in one pipeline to enable multi-platform deployment:
  - E.g., deploy .NET backend via Windows, iOS app via MacOS and Angular frontend via Linux
- Jobs run on the host machine where the agent is installed



# Multi-configuration and multi-agent

**Multi-configuration:** Run the same set of tasks on multiple configurations:

- Run the release once with configuration setting A on WebApp A and setting B for WebApp B
- Deploy to different geographic regions

**Multi-agent:** Run the same set of tasks on multiple agents using the specified number of agents:

- Deploy same bits to a farm of servers



# Setting up test infrastructure

Test agents are  
needed to run  
tests on target  
servers

Run in  
the cloud

Run on your  
own servers or  
target servers





# Setting up and running availability tests

Create health end points in your application

Use tools, e.g., availability tests in Application Insights, to test health endpoints

Two common types of availability tests:

- URL ping test
- Multi-step web test



# Automate inspection of health

Stay informed about your process and releases:



Release gates

Events,  
subscriptions,  
and  
notifications

Service  
hooks

Reporting

# Events, subscriptions, and notifications

- Actions in Azure DevOps trigger events

Users can subscribe to events and get notified:

- Calling a SOAP URL
- Receiving an email
- Notifications can be managed centrally and personally

# Service hooks

Service hooks enable you to perform tasks on other services when events happen

Out of the box integrations

Build and release	Collaborate	Customer support	Plan and track	Integrate
AppVeyor	Campfire	UserVoice	Trello	Azure Service Bus
Bamboo	Flowdock	Zendesk		Azure Storage
Jenkins	HipChat			Web Hooks
MyGet	Hubot			Zapier
Slack				

# The future of DevSecOps

## Attractive Opportunities in DevSecOps Market



# DevOps to DevSecOps transition

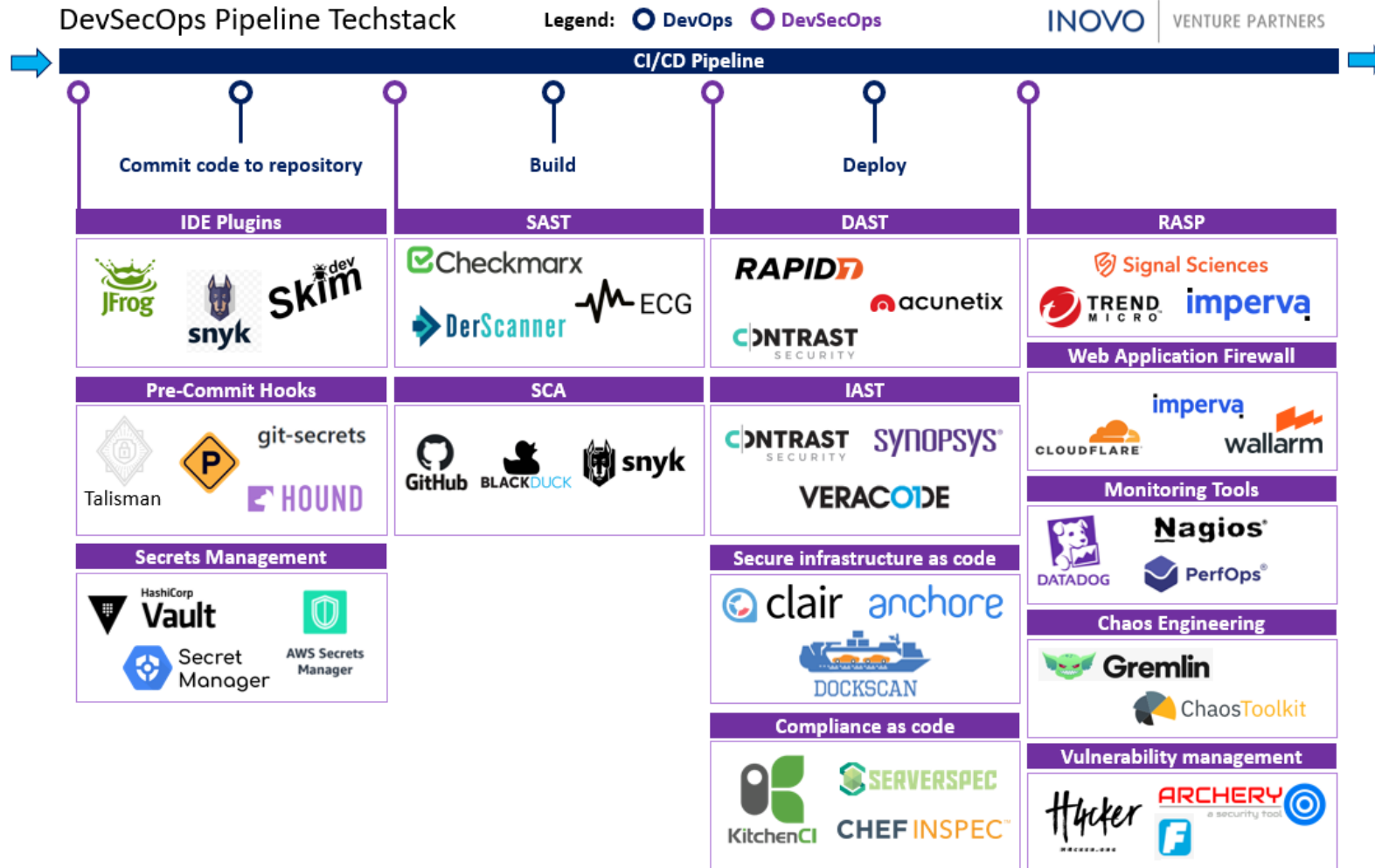
**FIG 3.2:** The integration of DevOps and security has already started to take hold in most organizations that have already adopted DevOps practices, according to a 2018 DevOps survey of over 1,000 IT pros by Logz.io.



# CICD DevSecOps integration



# Example Pipeline tools





# Elements of a CI/CD pipeline

- **Build** - The stage where the application is compiled.
- **Test** - The stage where code is tested. Automation here can save both time and effort.
- **Release** - The stage where the application is delivered to the repository.
- **Deploy** - In this stage code is deployed to production.
- **Validation and compliance** - The steps to validate a build are determined by the needs of your organization. Image security scanning tools, like [Clair](#), can ensure the quality of images by comparing them to known [vulnerabilities \(CVEs\)](#).

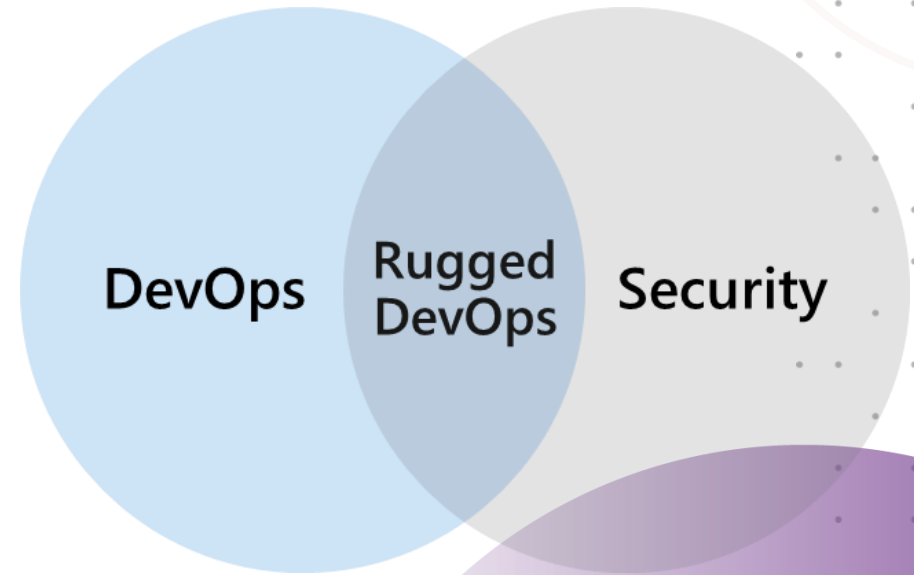




# Security in the pipeline

# What is DevSecOps ?

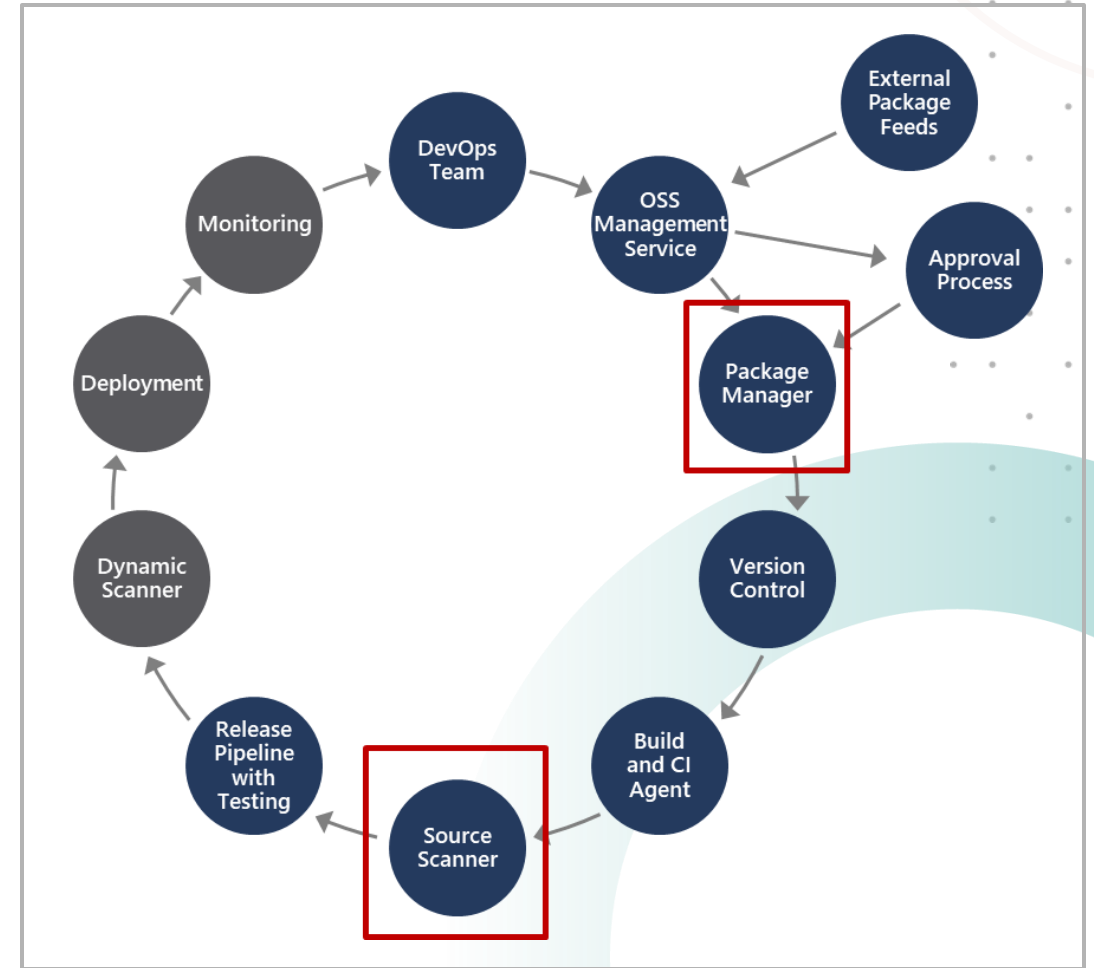
- DevSecOps is a set of practices designed to integrate DevOps and security
- The goal is to enable development teams to work fast without introducing unwanted vulnerabilities
- Security strategy includes access control, environment hardening, perimeter protection, and more



*DevSecOps is another term for Rugged DevOps*

# DevSecOps pipeline

- Package management, and the approval process associated with it, accounts for how software packages are added to the pipeline, and the approval process they need to go through
- Source scanner is for performing a security scan to verify certain security vulnerabilities are not present in our application source code



# Software composition analysis (SCA)

## Package Management:

- Unique source of binary components
- Create a local cache of approved components
- Use Azure Artifacts to share and organize your packages
- Package types: NuGet, npm, Maven, Gradle, Universal, and Python

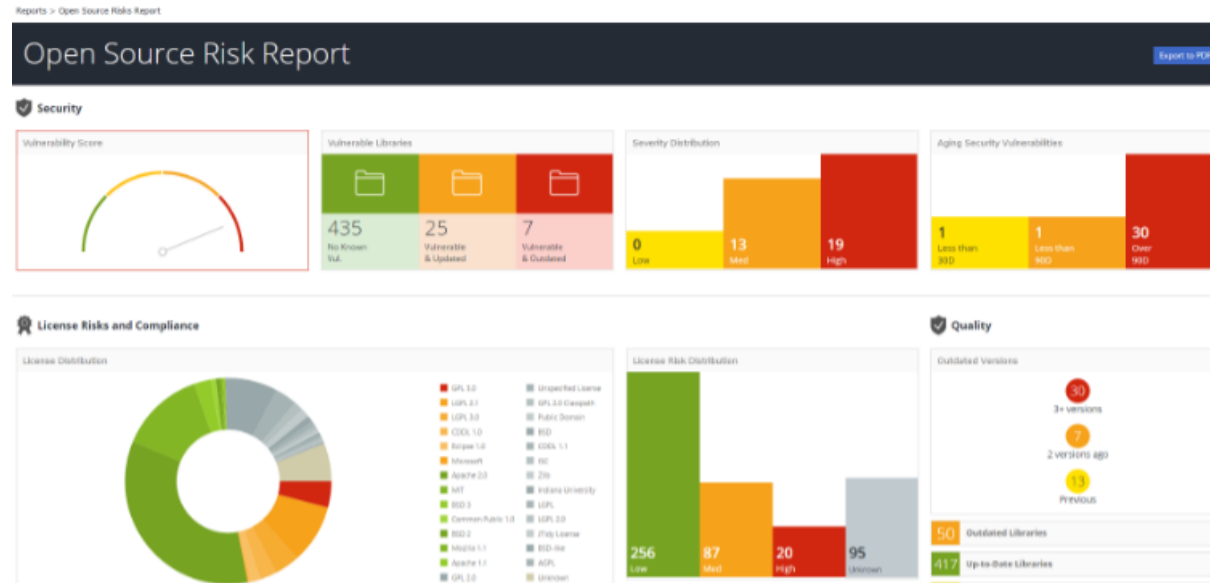
## Open-Source Software (OSS) Components:

- Reused dependencies can have security vulnerabilities
- Ensure you have the latest version
- Check for the correct binaries
- Promptly address vulnerabilities
- Analyzing the software to determine its composition can help mitigate potential vulnerabilities
- Scanning tools discussed in the next module

# WhiteSource integration with Azure DevSecOps pipeline

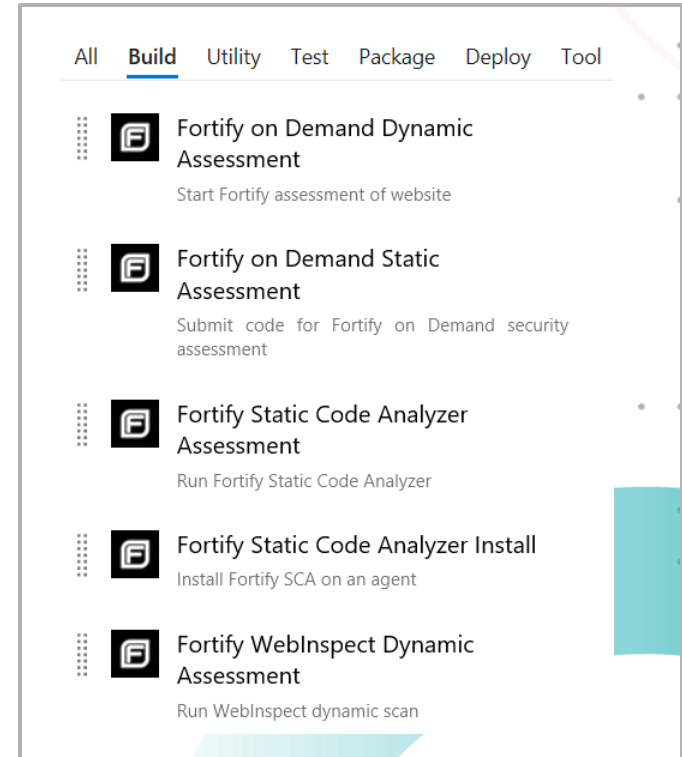
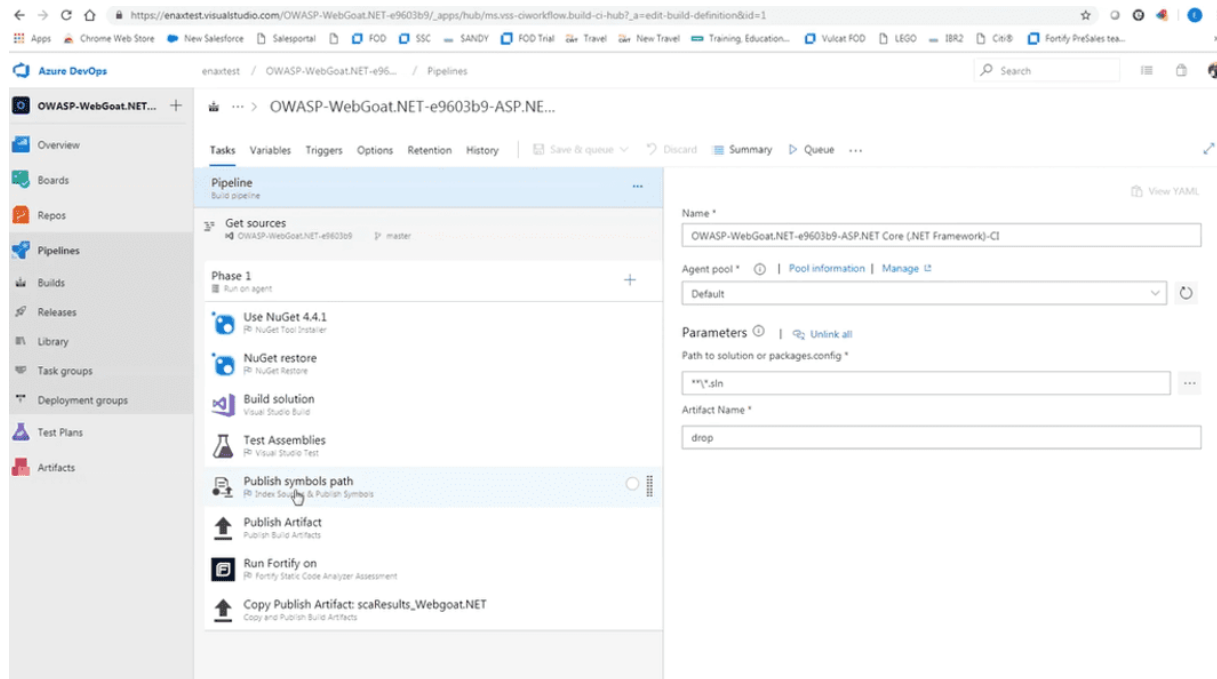
*WhiteSource* is one such example of an extension available on the Azure DevOps Marketplace

- If consuming external packages, the *WhiteSource* extension specifically addresses the questions of open-source security, quality, and license compliance.
- Continuously detect all open-source components in your software
- Receive alerts on open-source security vulnerabilities
- Automatically enforce open-source security and license compliance



# Micro Focus Fortify integration with Azure Pipelines

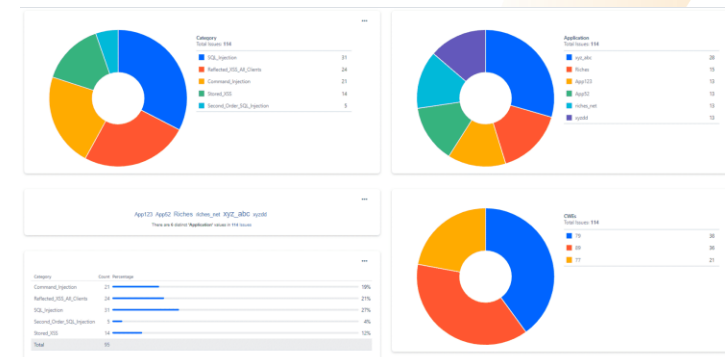
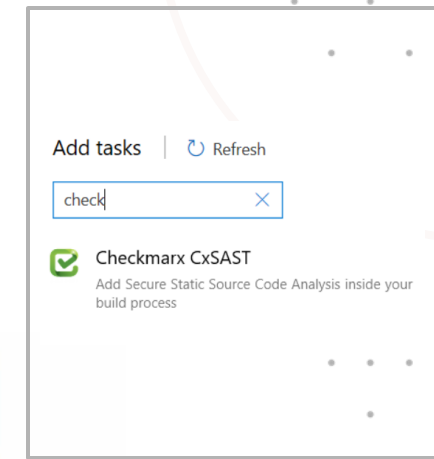
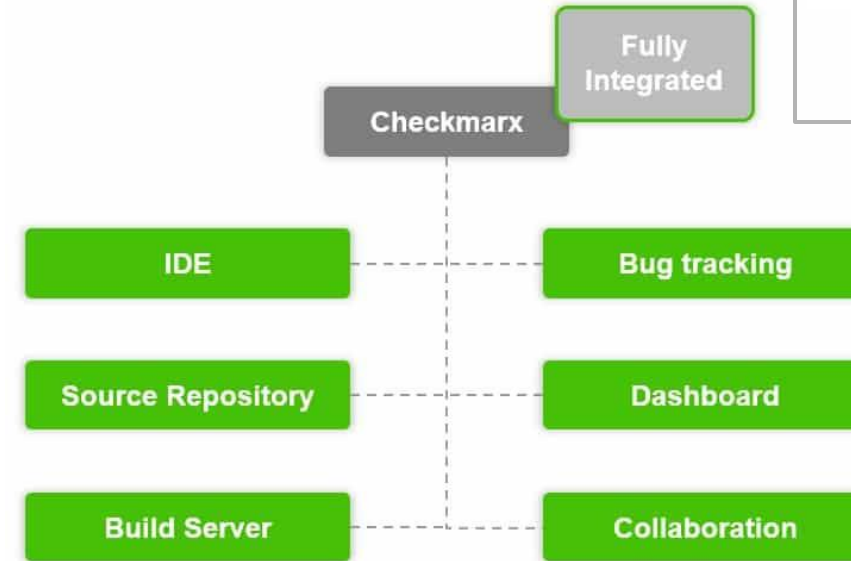
- Can add build tasks to CI/CD pipeline to help identify vulnerabilities in your source code
- Provides a comprehensive set of software security analyzers
- **Fortify Static Code Analyser:** Identifies root causes of software security vulnerabilities
- **Fortify on Demand:** Automatically submits static and dynamic scan requests



# Checkmarx integration with Azure DevOps

Identifies, tracks, and fixes technical and logical security flaws:

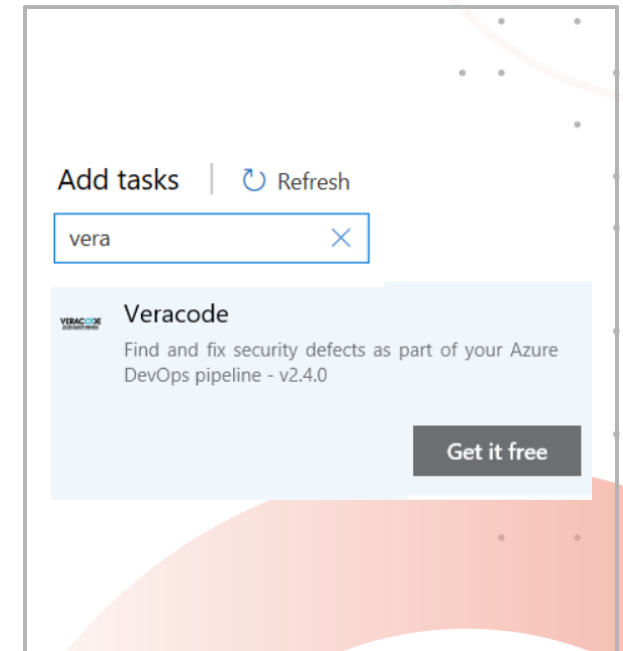
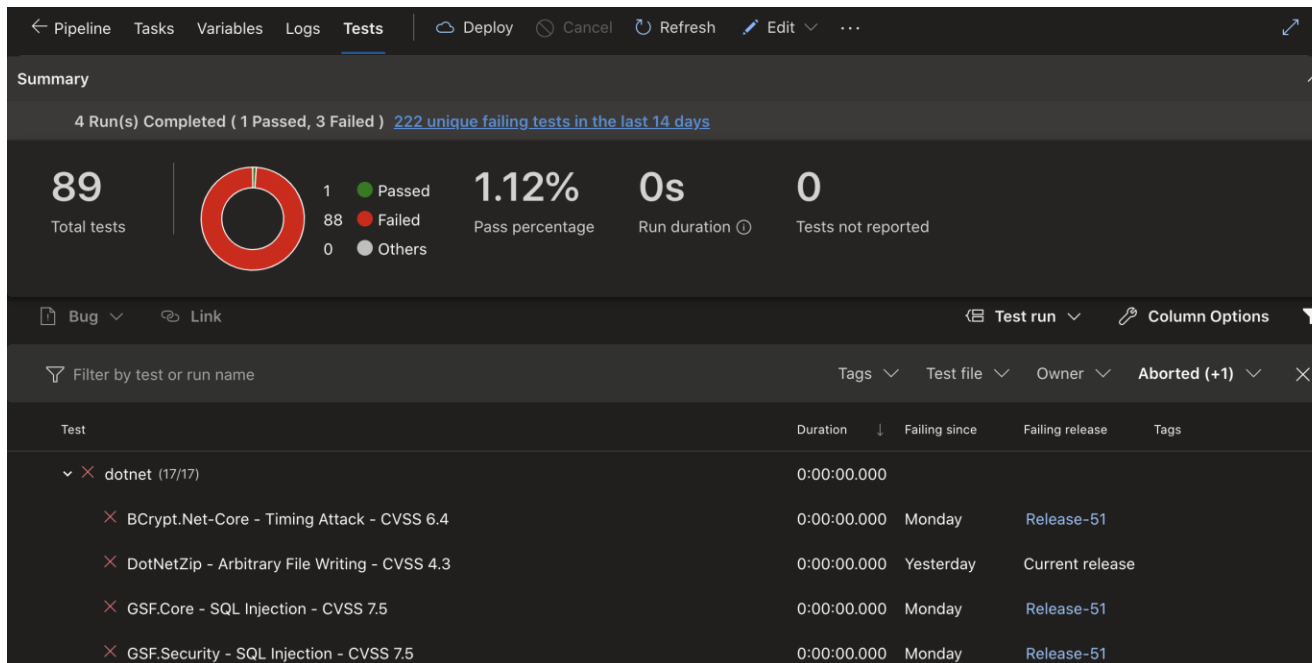
- Best fix location
- Quick and accurate scanning
- Incremental scanning
- Seamless integration
- Code portfolio protection
- Easy to initiate Open-Source Analysis with *Checkmarx Open-Source Analysis (CxOSA)* and *Checkmarx Static Application Security Testing (CxSAST)*



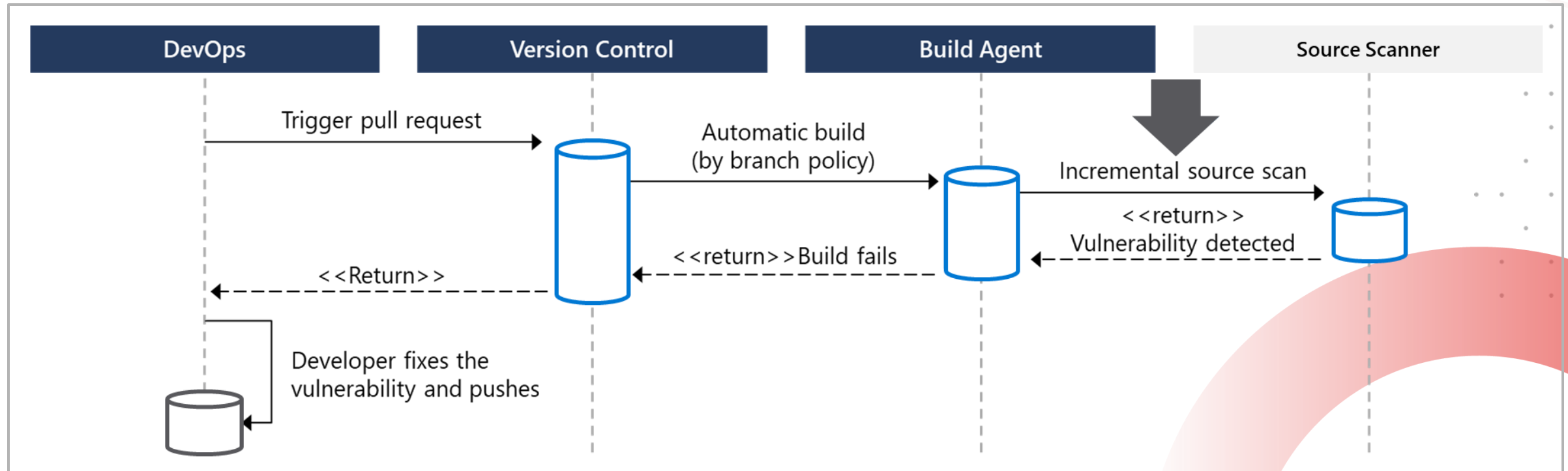


# Veracode integration with Azure DevOps

- Integrate application security into your development tools
- Don't stop for false alarms
- Align your application security practices with your development practices
- Don't just find vulnerabilities, fix them
- Onboarding process allows for scanning on day one



# How to integrate software composition analysis checks into pipelines



*Pull requests* are the way DevOps teams submit changes

WhiteSource, Checkmarx, Veracode, and Black Duck by Synopsis can facilitate incremental scans

Integrate scanning into a team's workflow at multiple points along the path



# Implementing pipeline security

Use traditional operational methods for protecting identities and assets

- **Authentication and Authorisation:** Use Multi-Factor Authentication and Just enough Admin (JEA)
- **Use the CI/CD release pipeline:** Use immutable infrastructure and only using your pipeline
- **Manage Permissions:** Secure the pipeline using RBAC
- **Dynamic Scanning:** Test running apps, such as penetration testing
- **Monitoring Production:** Look for anomalies related to intrusion

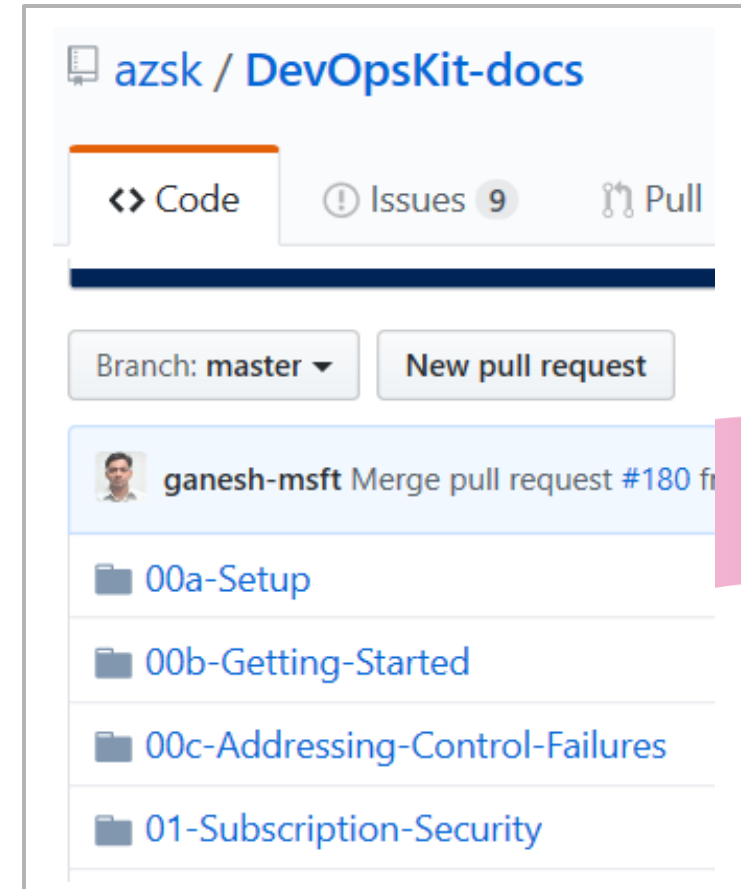


# Secure DevOps kit for Azure (AzSK)

AzSK is a collection of scripts, tools, extensions, automations

- Helps secure the subscription
- Enables secure development
- Integrates security into CI/CD
- Provides continuous assurance
- Assists with alerts & monitoring
- Governing cloud risks

Available at: <https://github.com/azsk/DevOpsKit-docs>

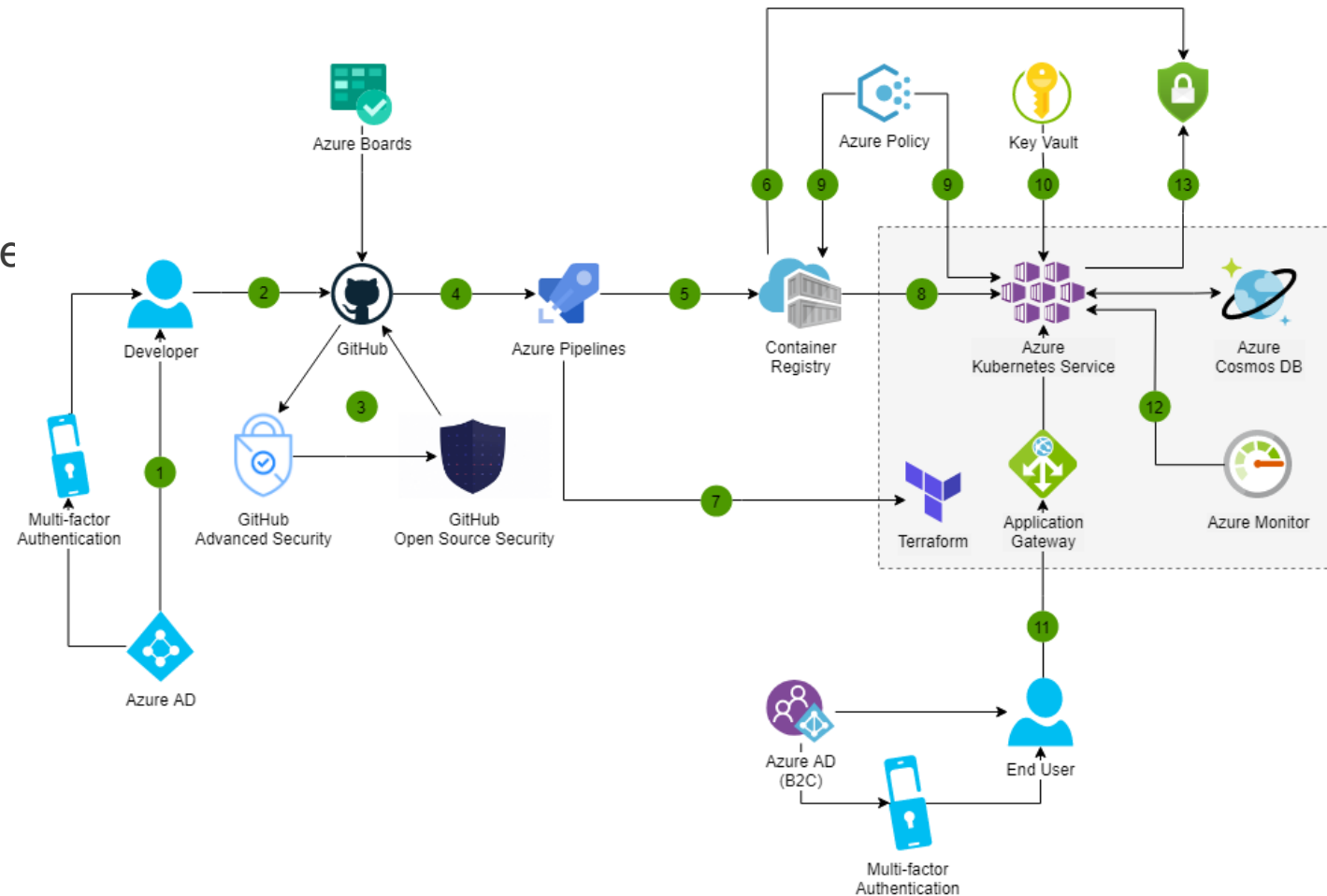




# Azure Security Center

# Azure Security Center

- Provide security recommendations
- Monitor all your services
- Analyze and identify potential attacks
- Supports Windows and Linux operating systems
- Monitor security settings
- Use Azure Machine Learning
- Provide just-in-time (JIT) access control
- Available in two pricing tiers



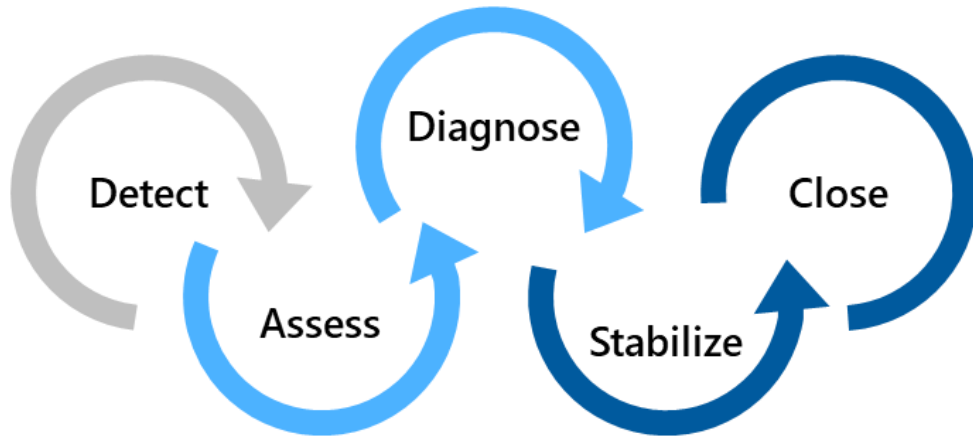
# Azure Security Center usage scenarios

## Scenario 1

Detect – Verify a high security alert was raised

Access – Obtain information about the alert

Diagnose – Follow the remediation steps



## Scenario 2

Security Policy

Recommendations

- Configure a security policy
- Implement the recommendations for the security policy

# Azure Policy

- Is a service in Azure that you use to create, assign, and manage policies
- Provides enforcement by using policies and initiatives
- Runs evaluations on your resources and scans for those not compliant
- Comes with several built-in policy and initiative definitions
- Integrates with Azure DevOps by applying any continuous integration (CI) and continuous deployment (CD) pipeline policies

An example of an Azure policy that you can integrate with your DevOps pipeline is the Check Gate task

Home > Policy - Assignments

## Policy - Assignments

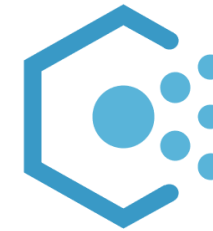
Search (Ctrl+/,)

Assign policy Assign initiative Refresh

Scope: Contoso Definition type: All definition types

Total Assignments 2 Initiative Assignments 1 Policy Assignments 1

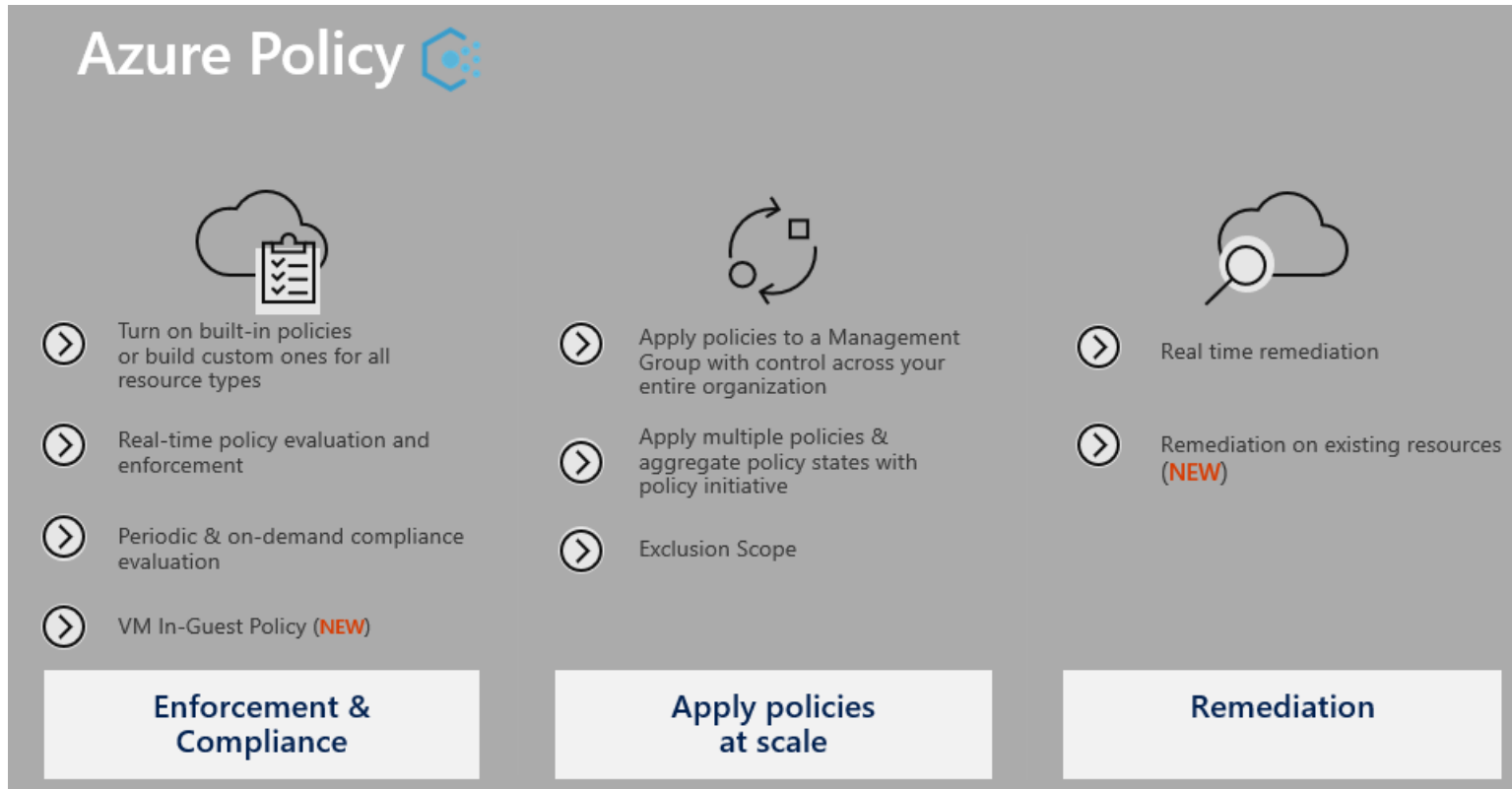
NAME	SCOPE
[Preview]: Enable Monitoring in Azure Security Center	Contoso/PolicyTarget
Approved VM images Assignment	Contoso/PolicyTarget





# Policies

- A *policy definition* expresses what to evaluate and what action to take
- Policies are defined in JSON
- Assign policies using Azure Portal, Azure CLI, or Azure PowerShell
- Use remediation for non-compliant resources
- The screenshot is a section of a policy file defining allowed locations



The image shows the Azure Policy console interface. At the top left is the 'Azure Policy' logo. Below it, there are three main sections: 'Enforcement & Compliance', 'Apply policies at scale', and 'Remediation'. Each section contains a list of actions with icons and descriptions.

Enforcement & Compliance	Apply policies at scale	Remediation
<ul style="list-style-type: none"><li>Turn on built-in policies or build custom ones for all resource types</li><li>Real-time policy evaluation and enforcement</li><li>Periodic &amp; on-demand compliance evaluation</li><li>VM In-Guest Policy (NEW)</li></ul>	<ul style="list-style-type: none"><li>Apply policies to a Management Group with control across your entire organization</li><li>Apply multiple policies &amp; aggregate policy states with policy initiative</li><li>Exclusion Scope</li></ul>	<ul style="list-style-type: none"><li>Real time remediation</li><li>Remediation on existing resources (NEW)</li></ul>

```
"displayName": "Allowed locations",
"description": "This policy enables you to
restrict the locations your organization can specify
when deploying resources.",
"policyRule": {
  "if": {
    "not": {
      "field": "location",
      "in": "[parameters('allowedLocations')]"
    }
  },
  "then": {
    "effect": "deny"
  }
}
```

# Initiatives

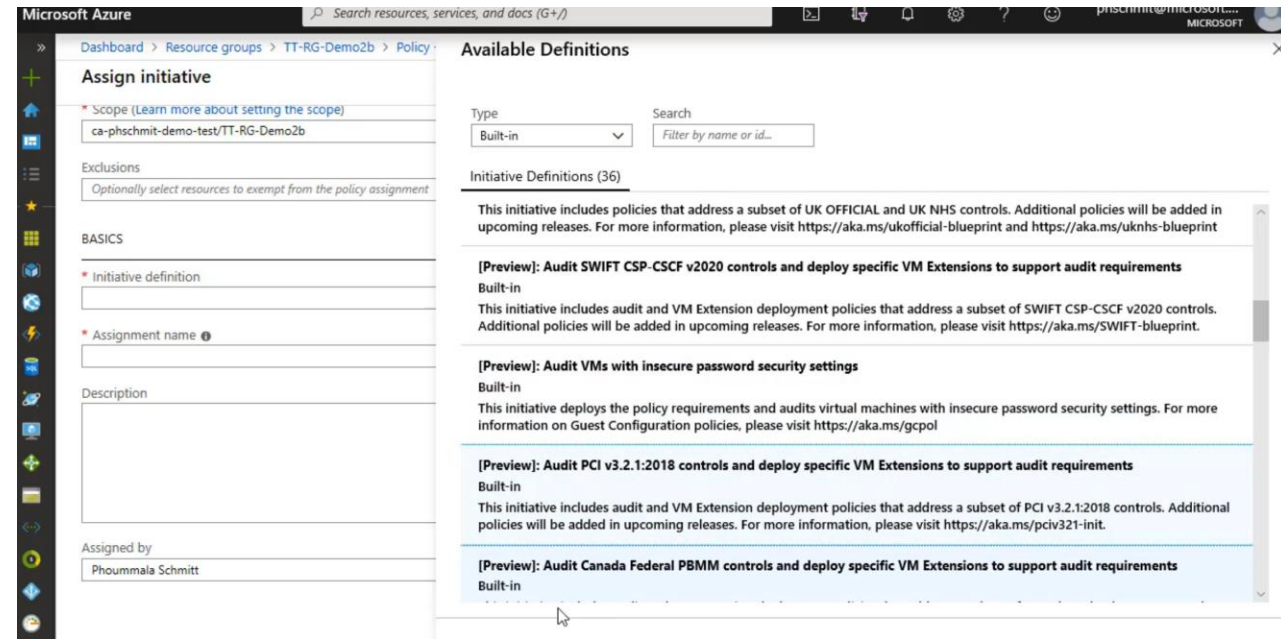
- An *initiative definition* is a set of policy definitions
- Helps track your compliance state for a larger goal
- Assigned to a specific scope
- Reduces the need for individual scope assignments

Even if you have a single policy, we recommend using initiatives if you anticipate increasing the number of policies over time.

## Initiative Example:

Create an Initiative named “*Enable Monitoring In Azure Security Center.*” This would provide the following policies:

- Monitor unencrypted SQL Database policy definition
- Monitor OS vulnerabilities policy definition
- Monitor missing Endpoint Protection policy definition

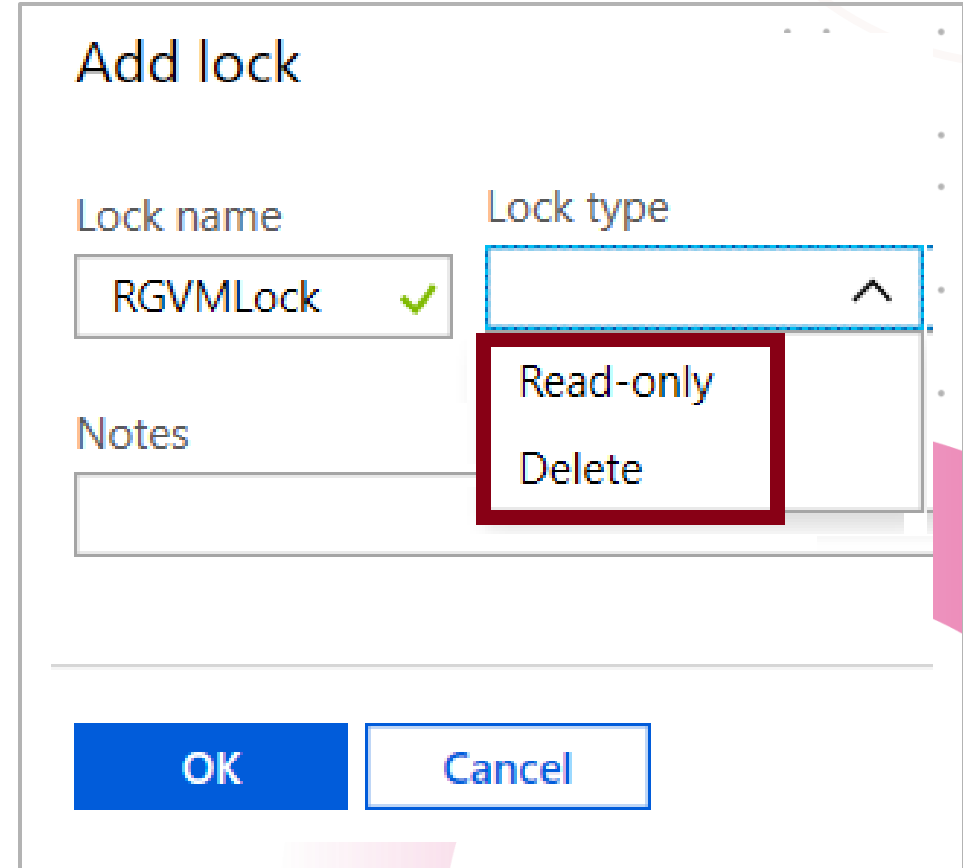


# Resource locks

Prevent accidental deletion or modification of your Azure resources:

- **CanNotDelete** – means authorised users can still read and modify a resource, but they can't *delete* the resource
- **ReadOnly** – means authorised users can read a resource, but they can't delete or update it. Applying this lock is like restricting all authorized users to the permissions granted by the Reader role

In the Azure portal, the locks are called *Delete* and *Read-only* respectively



**Add lock**

Lock name: RGVMLock ✓

Lock type: Read-only  
Delete

Notes:

OK Cancel

# Azure Blueprints

Allows for the definition of a repeatable set of Azure resources that implement and adhere to an organization's standards, patterns, and requirements

- Can ensure a deployment meets an organisation's standards, patterns, and requirements
- Is a declarative way to orchestrate deployment for various resource templates and other artifacts
- Provides for traceability and auditing as well as adherence



The blueprint definition (what should be deployed) and the blueprint assignment (what is deployed) supports improved deployment tracking and auditing

# Azure Advanced Threat Protection (ATP)

Identifies, detects, and helps you investigate advanced threats, compromised identities, and malicious insider actions:

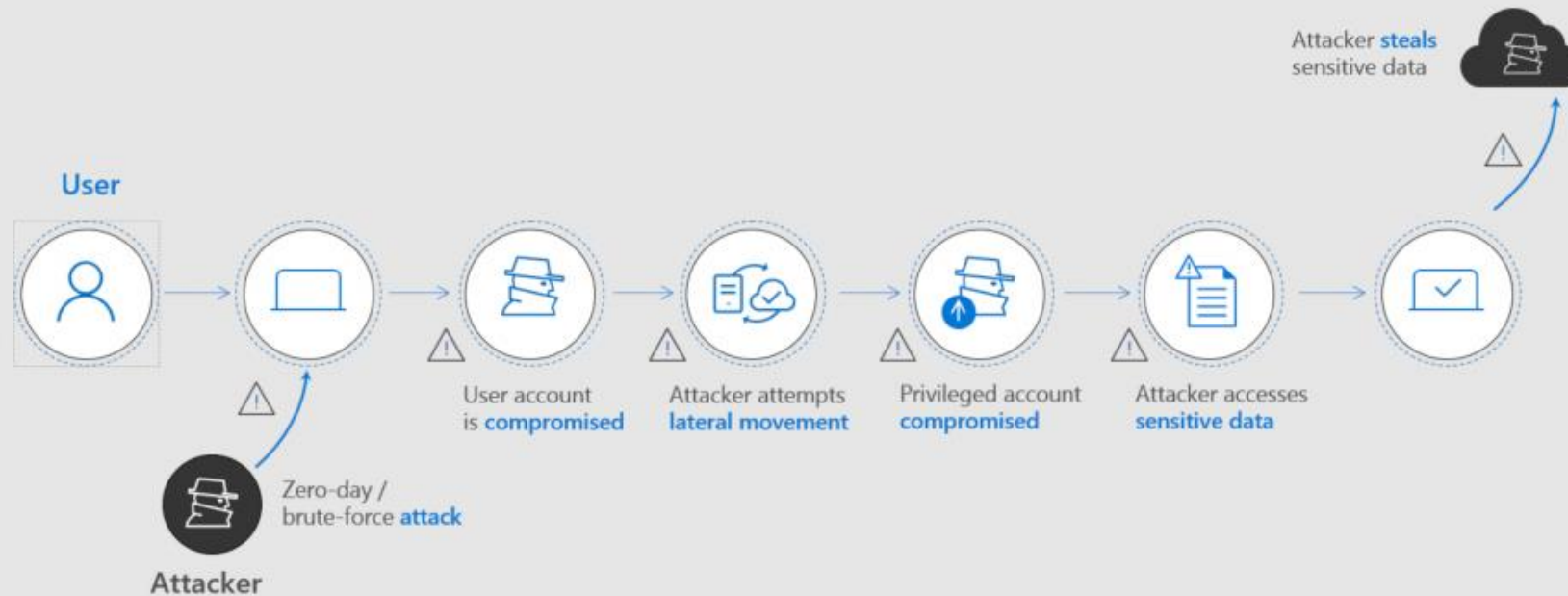
- Azure ATP **portal** monitors and responds to suspicious activity
- Azure ATP **sensor** monitors domain controller traffic
- Azure ATP **cloud service** connects to Microsoft Intelligent Security Graph

<https://techcommunity.microsoft.com/t5/security-compliance-and-identity/introducing-azure-advanced-threat-protection/ba-p/250332>



# Azure Advanced Threat Protection (ATP)

## The anatomy of an attack



Anomalous user behavior  
Unfamiliar sign-in location

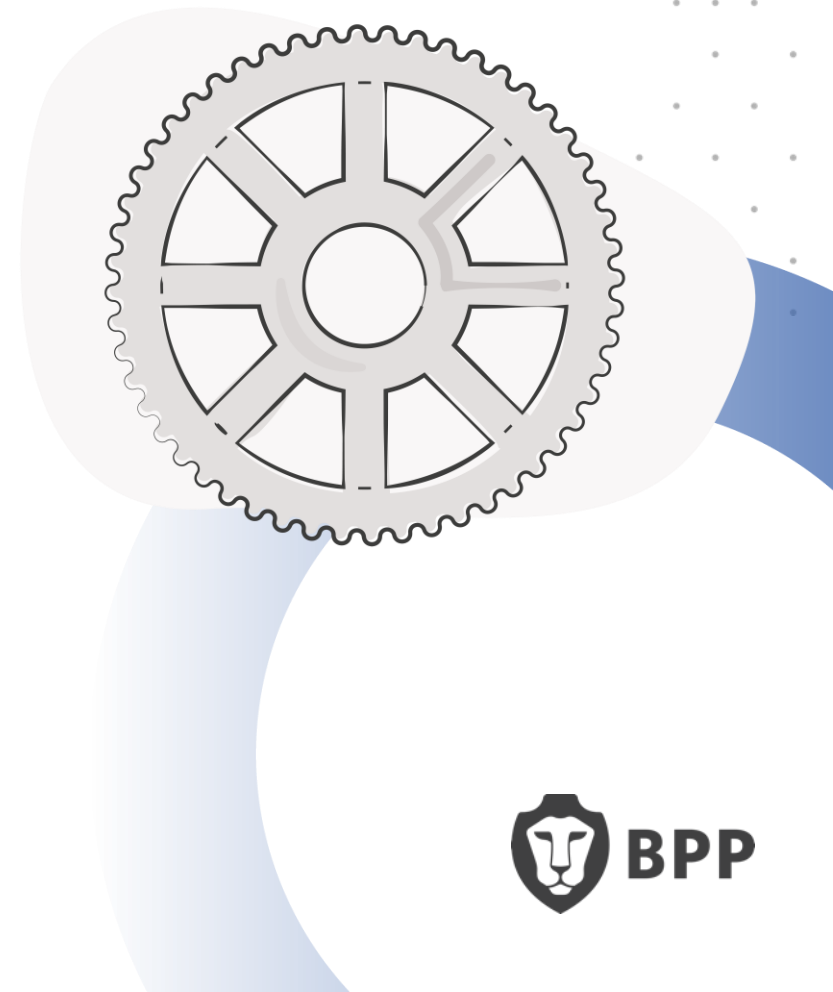


Lateral movement attacks  
Escalation of privileges  
Account impersonation

# Implement security and compliance in Azure DevOps Pipelines

## Objectives:

- Create a Build pipeline
- Install WhiteSource Bolt from the Azure DevOps marketplace and activate it
- Add WhiteSource Bolt as a build task in a build pipeline
- Run build pipeline and view WhiteSource security and compliance report



# Next Steps

Read about **performance engineering** and then reflect on its role in achieving security

<https://docs.aws.amazon.com/prescriptive-guidance/latest/performance-engineering-aws/introduction.html>







**Thank you**

**Do you have any questions,  
comments, or feedback?**

