



# Secure Your Code :

## Implement DevSecOps in Azure

21 Feb 2021



@srkbngl

# Who am I?



Serkan Bingöl

Cloud & DevOps Consultant [@kloia](#)  
MCSD / MCT / AAI / AWS Certified

Blog : [medium.com/@serkanbingoll](https://medium.com/@serkanbingoll)  
Tweet : [@srkbnegl](https://twitter.com/srkbnegl)  
LinkedIn : [/in/sbingol](https://www.linkedin.com/in/sbingol)  
GitHub : [github.com/serkanbingol](https://github.com/serkanbingol)

# Agenda

- What is Dev{Sec}Ops ?
- DevSecOps in GitHub & Demos
- DevSecOps in Azure
- 3<sup>rd</sup> Party DevSecOps Tools
- DEMO : Implement Security in Azure DevOps CI/CD
- Q & A

# What is Dev{Sec}Ops ?

**DevOps** is the union of people, process, and technology to enable continuous delivery of value to your end users.



image from <https://stackify.com/devops-engineer-starter-guide/>

## Automation + Security

**DevSecOps** is a cultural movement that furthers the movements of Agile and DevOps into Security



image from <https://www.recordedfuture.com/>

# What is Dev{Sec}Ops ?

**DEV : OPS :SEC Ratio - 100:10:1**

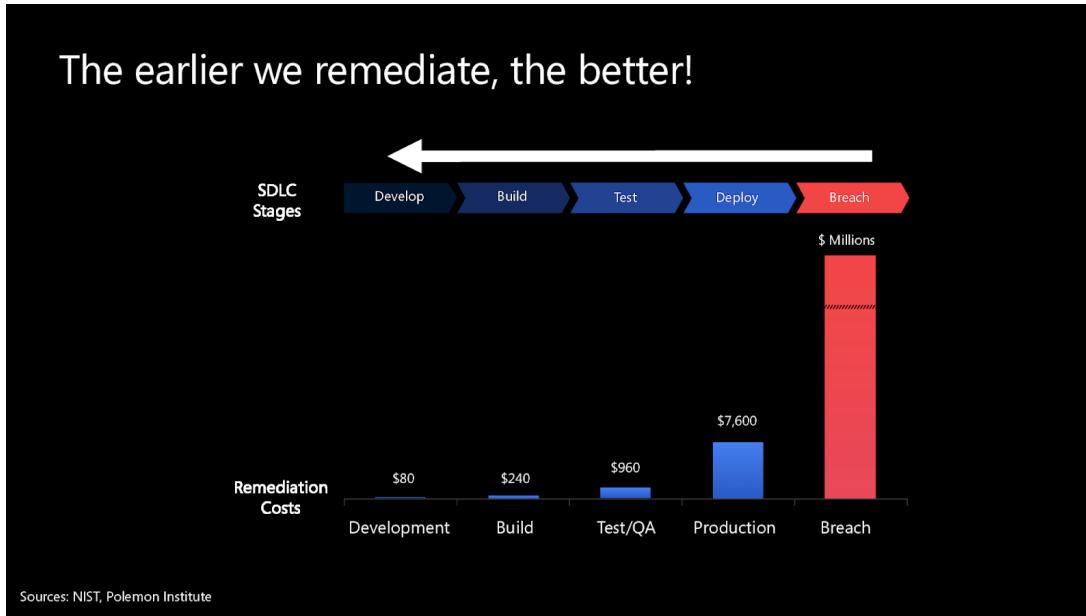
“The ratio of engineers in Development, Operations, and Infosec in a typical technology organization is 100:10:1. When Infosec is that outnumbered, without automation and integrating information security into the daily work of Dev and Ops, Infosec can only do compliance checking, which is the opposite of security engineering—and besides, it also makes everyone hate us.”



image from <https://twitter.com/joshcorman/status/644153295464960000>

# What is Dev{Sec}Ops ?

Discover vulnerability during the development.



\* Slide 9 : <https://speaking.sasharosenbaum.com/EI8lo/slides>

# What is Dev{Sec}Ops ?

**More we code , more we need security.**

Insecure code causes breaches Source: 2019 Data Breach Investigations Report, Verizon 53% of breaches are caused by weaknesses in applications.

**53%**

of breaches are caused by weaknesses in applications

Report link :

<https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf>

# What is Dev{Sec}Ops ?

## Security Patterns : Old Path Vs. New Path

- Embrace Secrecy
- Just Pass Audit!
- Enforce Stability
- Build a Wall
- Slow Validation
- Certainly Testing
- Test When DONE !
- Process Driven



image from <https://wellbeingeconomy.org/>

- Create Feedback Loops
- Compliance adds Value
- Create Chaos
- Zero Trust Networks
- Fast and Non-Blocking
- Adversity Testing
- Shift Left
- The Paved Road

# What is Dev{Sec}Ops ?

## Azure Patterns: Infrastructure Planning

- ✗ Subscription **per** environment
- ✗ Apply **policies** to control transparently and proactively
- ✗ Security Center ON from day **1**
- ✗ Infrastructure as code. **Period**
- ✗ Production **ONLY** for CI/CD pipelines
- ✗ CI/CD pipeline is a **heart** of security



image from <https://bridgecrew.io/blog/infrastructure-as-code-security-101/>

# What is Dev{Sec}Ops ?

## Azure Patterns: Watch Every Step

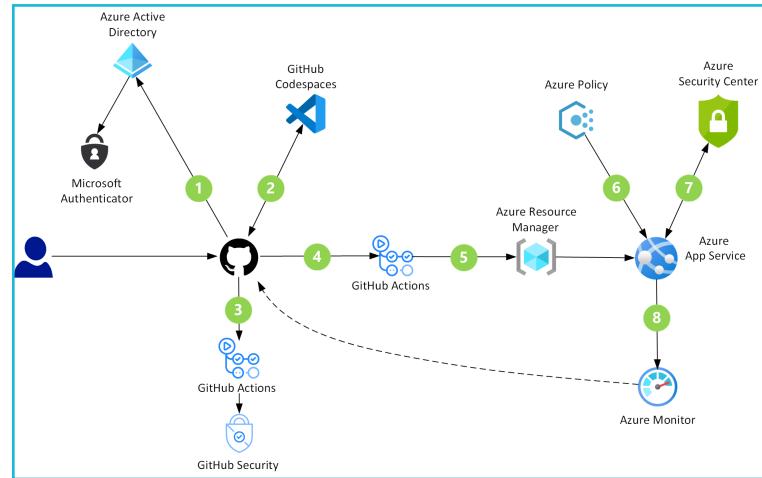


- Threat modeling
- IDE Security plugins
- Pre-commit hooks
- Peer review
- Static code analysis
- Security unit tests
- Container Security
- Dependency management
- IaC
- Security scanning
- Cloud configuration
- Security acceptance tests
- Security smoke tests
- Security Configuration
- Secret Management
- Server Hardening
- Continuous monitoring
- Threat intelligence
- Penetration Testing
- Blameless postmortems

# DevSecOps in GitHub

- Browser-based IDEs with built-in security extensions.
- Agents that continuously monitor security advisories and replace vulnerable and out-of-date dependencies.
- Search capabilities that scan source code for vulnerabilities.
- Action-based workflows that automate every step of development, testing, and deployment.
- Spaces that provide a way to privately discuss and resolve security threats and then publish the information.

\* Combined with the monitoring and evaluation power of Azure, these features provide a superb service for building secure cloud solutions.



\* <https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/devsecops-in-github>

# DevSecOps in GitHub

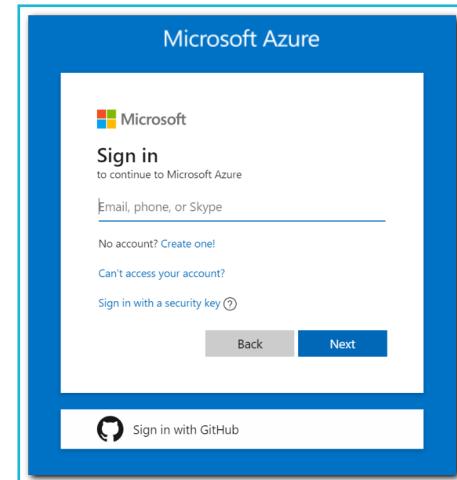
## Foundation of the Enterprise in Cloud : Azure AD

- Azure AD auth to GitHub
- GitHub auth to Azure portal
  - SAML SSO
  - LDAP
  - RBAC
  - Required 2FA
  - GitHub Connect
  - Audit log

4 Set up Internal GitHub Enterprise Server  
You'll need to configure the application to link with Azure AD.

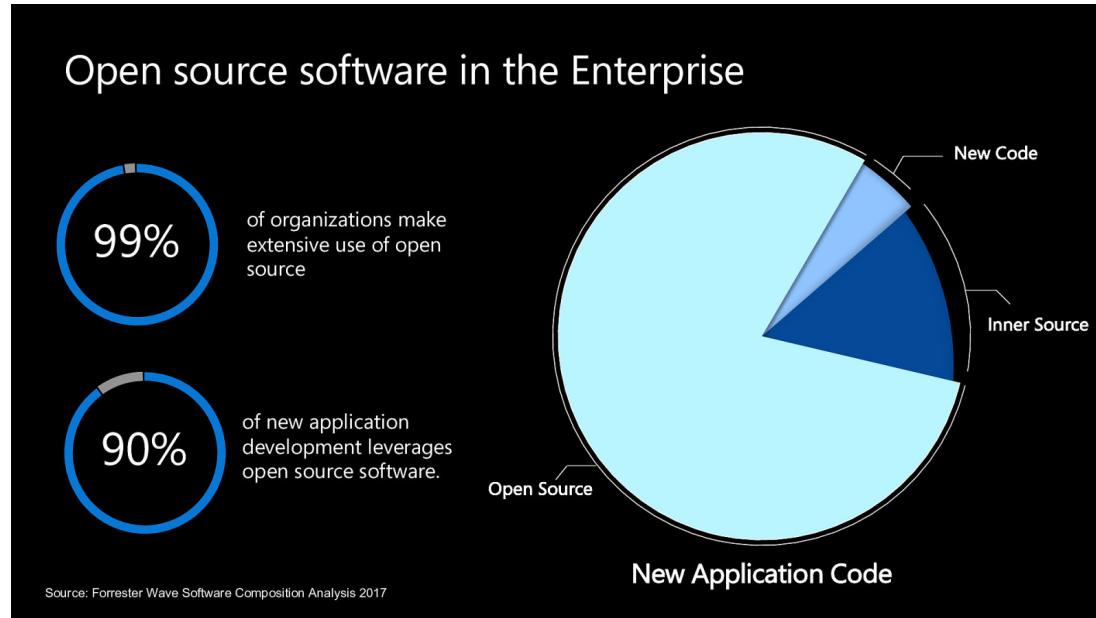
Login URL	<input type="text" value="https://login.microsoftonline.com/"/>	
Azure AD Identifier	<input type="text" value="https://sts.windows.net/"/>	
Logout URL	<input type="text" value="https://login.microsoftonline.com/common/wsf..."/>	

[View step-by-step instructions](#)



# DevSecOps in GitHub

## Secure your supply chain

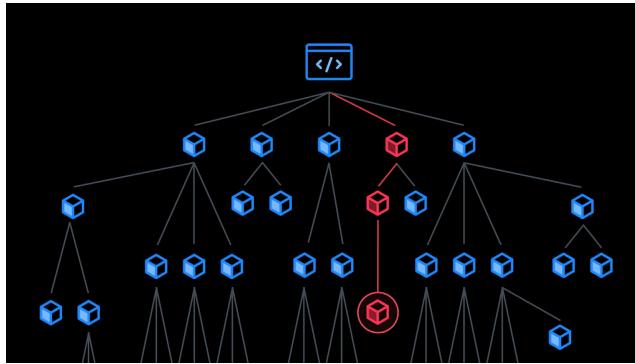
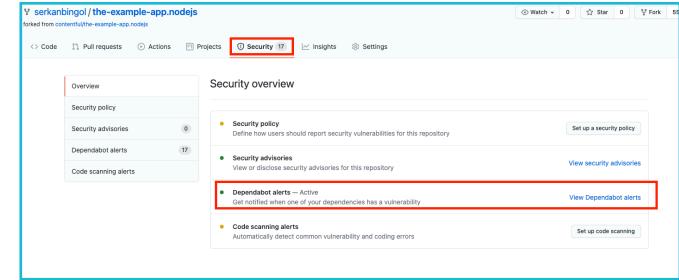


\* Slide 22 : <https://speaking.sasharosenbaum.com/EI8loj/slides>

# DevSecOps in GitHub

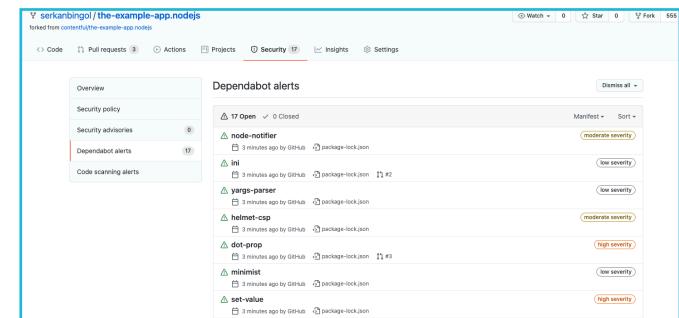
## Secure your supply chain: GitHub Dependabot Alerts

- Automate dependency updates
- Identify vulnerable dependencies

**Dependabot alerts — Active**  
Get notified when one of your dependencies has a vulnerability

**Code scanning alerts**  
Automatically detect common vulnerability and coding errors



Alert Type	Package	Last Update
node-notifier	node-notifier	3 minutes ago by GitHub
ini	ini	3 minutes ago by GitHub
yaml-parser	yaml-parser	3 minutes ago by GitHub
helmet-csp	helmet-csp	3 minutes ago by GitHub
dot-prop	dot-prop	3 minutes ago by GitHub
minimist	minimist	3 minutes ago by GitHub
set-value	set-value	3 minutes ago by GitHub

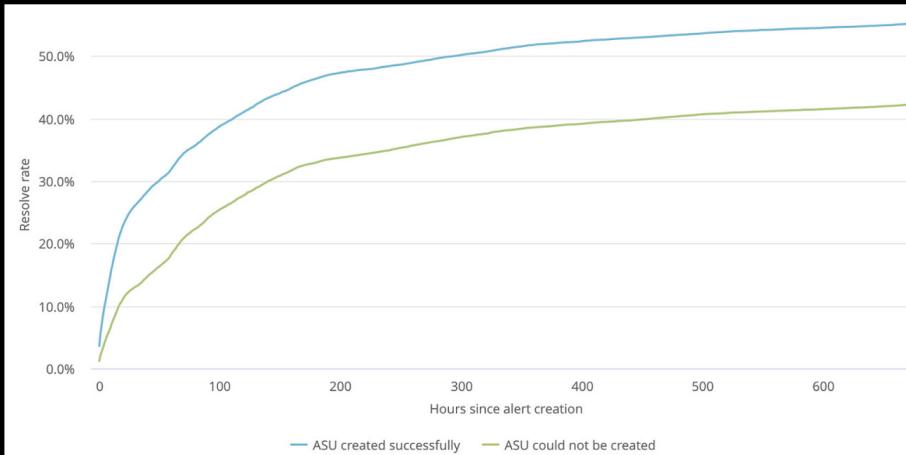
Managing Vulnerabilities in your project Dependencies :

<https://docs.github.com/en/github/managing-security-vulnerabilities/managing-vulnerabilities-in-your-projects-dependencies>

# DevSecOps in GitHub

## Secure your supply chain: GitHub Dependabot Alerts

Dependabot increases the resolve rate and speed



\* Slide 30 : <https://speaking.sasharosenbaum.com/EI8loj/slides>

# DevSecOps in GitHub

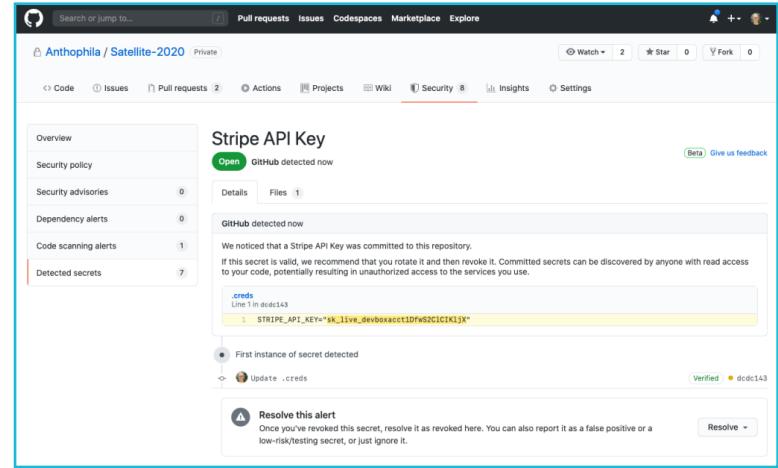
## Secure your repository: GitHub Secret Scanning

- Scan and detect secrets in your repository
- Lots of service providers
  - Atlassian
  - Azure
  - AWS
  - Alibaba Cloud
  - Google Cloud
  - Stripe
  - .....

\* Secret scanning is available in public repositories, and in private repositories owned by organizations with an Advanced Security license.

Configuring secret scanning for your repositories :

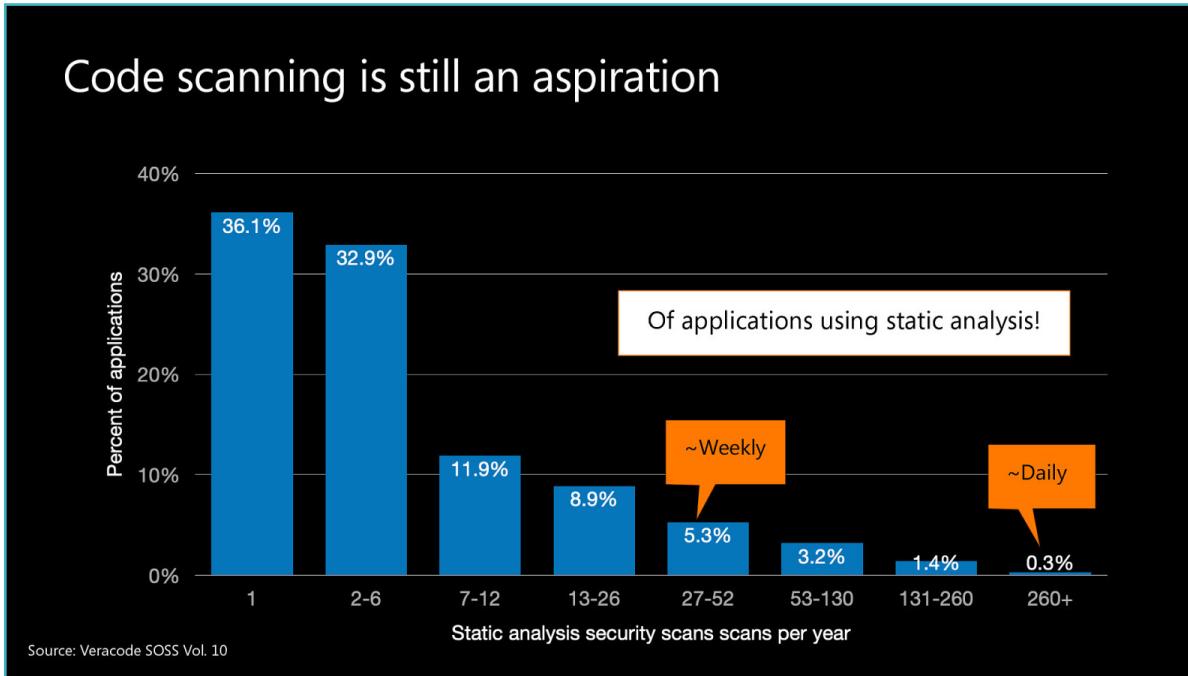
<https://docs.github.com/en/github/administering-a-repository/configuring-secret-scanning-for-your-repositories>



The screenshot shows the GitHub repository interface for 'Anthophila / Satellite-2020'. The 'Security' tab is selected. A prominent alert for a 'Stripe API Key' is displayed, stating 'GitHub detected now'. Below it, a note says: 'We noticed that a Stripe API Key was committed to this repository. If this secret is valid, we recommend that you rotate it and then revoke it. Committed secrets can be discovered by anyone with read access to your code, potentially resulting in unauthorized access to the services you use.' A code snippet from a '.creds' file shows the secret: 'STRIPE\_API\_KEY="sk\_live\_devboxacctIDFwSzC1C1kjXK"'. There are buttons to 'Resolve this alert' or 'Update .creds'. Other tabs like 'Overview', 'Security policy', 'Dependency alerts', and 'Code scanning alerts' are visible.

# DevSecOps in GitHub

## Analyse your code



# DevSecOps in GitHub

## Analyse your code: GitHub Code Scanning

- Automate your code review.
- CodeQL has leading security teams and individuals.
- CodeQL has more CVEs than any other static application security testing vendor team.
- GitHub is now a CNA.

The screenshot shows a GitHub repository page for 'serkanbingol/the-example-app.nodejs'. The 'Security' tab is selected, showing various alerts. A prominent alert is 'Incomplete string escaping or encoding' (Rule ID: JS/incomplete-sanitization), which is a warning. The alert details a specific line of code: 'public/scripts/index.js' containing 'i = function(e){function t(r){if(in(r))return n[r].exports;var i=n[r]=(i;r,i;i,exports:{});return e(r).call(i.exports,i,i.exports);}' and notes that it 'This replaces only the first occurrence of ```'.. Below the alert, there's a 'Tool' section for CodeQL, a 'Rule ID' section for JS/incomplete-sanitization, and a 'Query' section with a 'View source' link. At the bottom, it shows the commit where the alert was first appeared: 'First appeared in commit 5f8c245 2 minutes ago' and a link to 'Create codeql-analysis.yml'.

\* CVE : Common Vulnerabilities and Exposures

\*\* CNA : CVE Numbering Authority

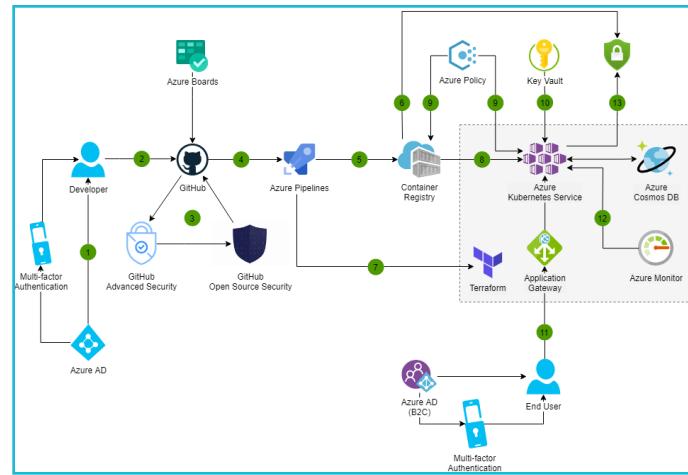
\*\*\* Zero-Day : Software vulnerability that is unknown to, or unaddressed by, those who are interested in mitigating it.

About code scanning :

<https://docs.github.com/en/github/finding-security-vulnerabilities-and-errors-in-your-code/about-code-scanning>

# DevSecOps in Azure

- **Azure AD** can be configured as the identity provider for GitHub.
- **GitHub Enterprise** can integrate automatic security and dependency scanning.
- **Azure Pipelines** generates a Docker container image that is stored to **Azure Container Registry**, which is to be used at release time by **Azure Kubernetes Service**.
- A release on Azure Pipelines integrates the **Terraform** tool, managing both the cloud infrastructure as code, provisioning resources such as Azure Kubernetes Service, **Application Gateway**, and **Azure Cosmos DB**.
- **Azure Security Center** will be able to do active threat monitoring on the Azure Kubernetes Service, on both Node level (VM threats) and internals.



\* <https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/devsecops-in-azure>

# DevSecOps in Azure

## Infrastructure as Code : Terraform

- Deliver Infrastructure as Code
  - Write : Write infrastructure as code using declarative configuration files.
  - Plan : Check whether the execution plan for a configuration matches your expectations before provisioning or changing infrastructure.
  - Apply : Apply changes to hundreds of cloud providers to reach the desired state of the configuration.

```

ado_project.tf ×
AzureDevOps > ado_project.tf
1 resource "azureddevops_project" "adoproj" {
2   project_name        = "TF Real-World Examples Project"
3   description         = "Example ADO project with real-world scenarios."
4   visibility          = "public" # Options: private, public
5   version_control     = "Git" # Options: Git, Tfvc
6   work_item_template = "Agile" # Options: Agile, Basic, CMMI, Scrum
7   features = {
8     "boards" = "enabled"
9     "repositories" = "enabled"
10    "pipelines" = "enabled"
11    "testplans" = "enabled"
12    "artifacts" = "enabled"
13  }
14}
15
16 # Either method (ie. inline or as a separate resource, will work)
17 resource "azureddevops_project_features" "my-project-features" {
18   project_id = azureddevops_project.adoproj.id
19   features = {
20     "boards" = "enabled"
21     "repositories" = "enabled"
22     "pipelines" = "enabled"
23     "testplans" = "enabled"
24     "artifacts" = "enabled"
25   }
26 }

```

image from <https://adinermie.com/>

Terraform is commercial templating tool that can provision cloud-native applications across all the major cloud players: Azure, Google Cloud Platform, AWS, and AliCloud. Instead of using JSON as the template definition language, it uses the slightly more terse YAML.

Infrastructure as code with Microsoft:

<https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/infrastructure-as-code>

# DevSecOps in Azure

## Policy-as-code : Azure Policy

- Understand evaluation outcomes
- Control the response to an evaluation
- Remediate non-compliant resources
- Single source of truth

```
HashiCorp Configuration Language | ⚙ Copy

provider "azurerm" {
    version = "~>2.0"
    features {}
}

resource "azurerm_policy_assignment" "auditvms" {
    name = "audit-vm-manageddisks"
    scope = var.cust_scope
    policy_definition_id = "/providers/Microsoft.Authorization/policyDefinitions/06a78e..."
    description = "Shows all virtual machines not using managed disks"
    display_name = "Audit VMs without managed disks Assignment"
}
```

image from <https://docs.microsoft.com/en-us/azure/governance/policy/assign-policy-terraform>

Azure Policy is a service that offers both built-in and user-defined policies across categories mapping the various Azure services such as Compute, Storage or even AKS. These policies can be defined on the Azure Portal and assigned to one or more subscriptions/resource groups.

Azure Policy documentation :  
<https://docs.microsoft.com/en-us/azure/governance/policy/>

# DevSecOps in Azure

## Maintain your keys : Azure Key Vault

- Centralize application secrets
- Securely store secrets and keys
- Simplified administration of application secrets
- Integrate with other Azure services

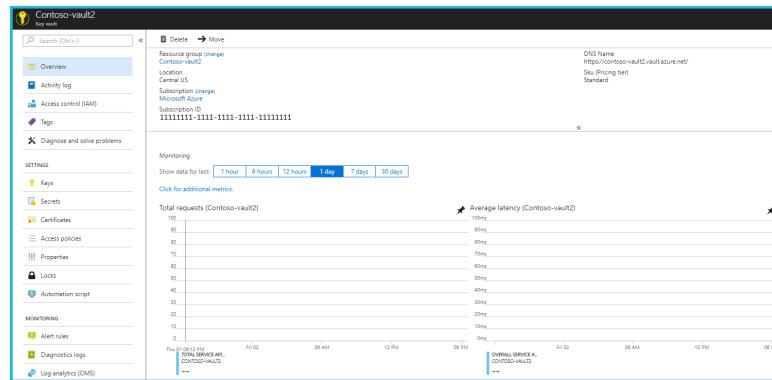


image from <https://docs.microsoft.com/en-us/azure/key-vault/general/quick-create-portal>

Azure Key Vault helps solve “Secrets Management” , “Key Management” and “Certificate Management” problems.

Azure Key Vault documentation :  
<https://docs.microsoft.com/en-us/azure/key-vault/general/overview>

# DevSecOps in Azure

## Keep your resources safe: Azure Security Center

- Strengthen security posture
- Protect against threats
- Get secure faster
- Prioritize your security work

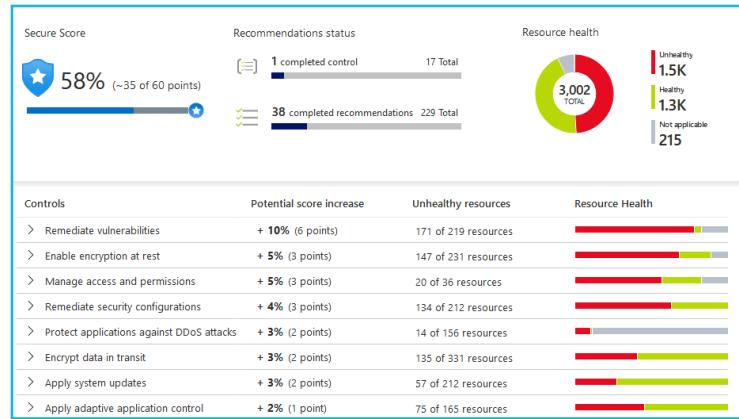


image from <https://docs.microsoft.com/en-us/azure/security-center/security-center-introduction>

Azure Security Center is a unified infrastructure security management system that strengthens the security posture of your data centers, and provides advanced threat protection across your hybrid workloads in the cloud - whether they're in Azure or not - as well as on premises.

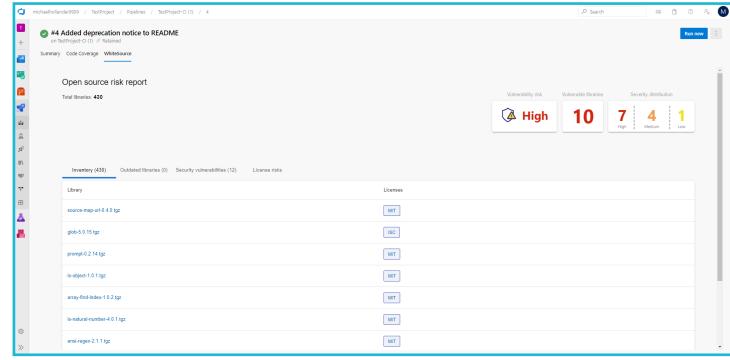
Azure Security Center documentation :

<https://docs.microsoft.com/en-us/azure/security-center/>

# 3rd Party DevSecOps Tools

## Check open source components : Whitesource Bolt

- WhiteSource Secures & Manages Your Open Source Usage
- Get Real-Time Alerts on Security Vulnerabilities
- Ensure license compliance
- Automated Up-to-Date Inventory Reports



The screenshot shows the Whitesource Bolt interface. At the top, there's a navigation bar with 'micahelkohler009' and tabs for 'Techniques', 'Reports', and 'Technique Details'. Below the navigation is a summary section with a green icon for '44 Added deprecation notice to README on Technique 0 (1) - Notified', a 'Summary' section with 'Code Coverage' and 'Whitelists', and a 'Open source risk report' section showing 'Total Sources: 450'. To the right of these are three boxes: 'Unresolved risk' (High: 10), 'Unresolved licenses' (7), and 'Severity distribution' (4 High, 1 Medium, 1 Low). The main area is titled 'Inventory (450)' and shows a list of dependencies with their licenses:

Dependency	License
source-map-api@4.4.1.tgz	MIT
glob@0.3.15.tgz	MIT
preact@2.2.14.tgz	MIT
is-equal-1.0.1.tgz	MIT
array-find-index@1.0.2.tgz	MIT
is-natural-number@0.0.1.tgz	MIT
ansi-regex@2.1.1.tgz	MIT

image from <https://marketplace.visualstudio.com/items?itemName=whitesource.whiteSource-bolt-v2>

WhiteSource Bolt scans all your projects and detects open source components, their license and known vulnerabilities. Not to mention, it also provide fixes.

Whitesource Bolt Ver. 1.0 documentation :

<https://marketplace.visualstudio.com/items?itemName=whitesource.ws-bolt>

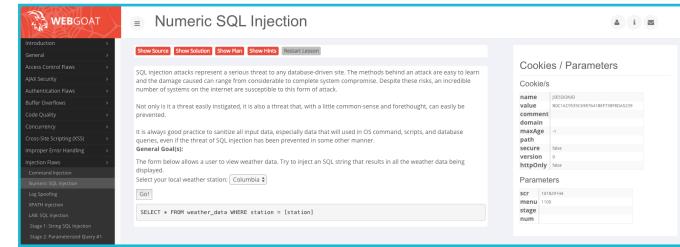
Whitesource Bolt Ver. 2.0 documentation :

<https://marketplace.visualstudio.com/items?itemName=whitesource.whiteSource-bolt-v2>

# 3rd Party DevSecOps Tools

## Demonstrate security and penetration testing: WebGoat

- Learn the hack - Stop the attack
- Test vulnerabilities
- Create a de-facto interactive teaching environment



The screenshot shows a web browser displaying the 'WEBGOAT' application. The main page is titled 'Numeric SQL Injection'. It contains a sidebar with a navigation menu and a central content area. The content area includes a descriptive text about SQL injection attacks, a goal statement, and a form for entering an SQL query. On the right side, there are sections for 'Cookies / Parameters' and 'Parameters' with their respective values.

Numeric SQL Injection

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

**General Goal(s):**

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

Select your local weather station:

Cookies / Parameters

name: SESSION  
value: BAC7AECB6E9A8A99B7D8A8A8A8A8A8A8  
comment:  
domain:  
path:  
secure:  
version:  
httpOnly:  
Parameters

scr: 1000x100  
menu: 100  
stage: num

image from <https://www.zupzup.org/websecurity-with-webgoat>

WebGoat is a deliberately insecure web application maintained by OWASP designed to teach web application security lessons.

WebGoat documentation :  
<https://github.com/WebGoat/WebGoat>

# 3rd Party DevSecOps Tools

## Create security report : Microsoft Security Code Analysis

- Simple configuration and execution
- Keep builds clean
- Auto-Update
- Break the build
- Scan Credentials

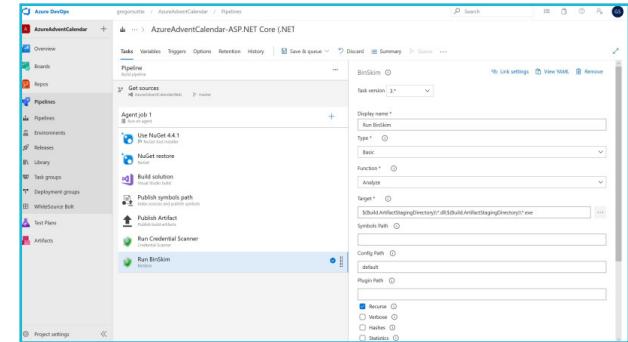


image from <https://techdailychronicle.com/>

The Microsoft Security Code Analysis extension empowers you to do so, easily integrating the running of static analysis tools in your Azure DevOps pipelines..

Microsoft Security Code Analysis Extension documentation :  
<https://secdevtools.azurewebsites.net/>

# 3rd Party DevSecOps Tools

## Scan containers for the security and compliance: Anchore

Configure Anchore to authenticate with Azure Container Registry (ACR) and analyze an image.

The Anchore Engine is an open-source project that provides a centralized service for inspection, analysis, and certification of container images. The Anchore Engine is provided as a Docker container image that can be run standalone or within an orchestration platform such as Kubernetes, Docker Swarm, Rancher, Amazon ECS, and other container orchestration platforms.

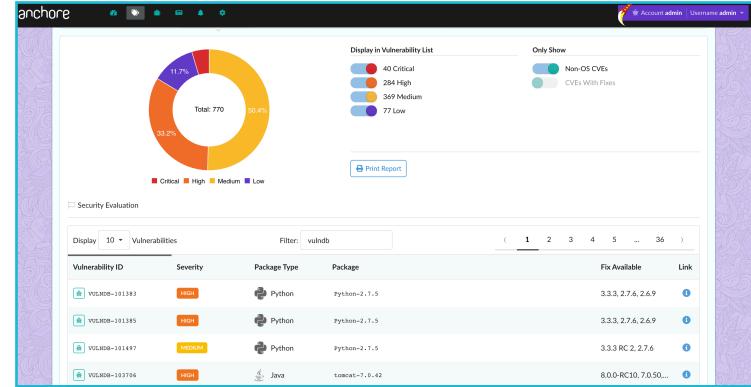


image from <https://anchore.com/blog/enhanced-vulnerability-data/>

Anchore documentation :

<https://anchore.comopensource/>

# DEMO: Implement Security in Azure DevOps CI/CD

## Prerequisites

- An Azure subscription
- Azure DevOps account
- GitHub Account
- Helm CLI
- Anchore CLI
- Kubectl

## Steps

- Create **Azure DevOps** project and prepare WebGoat source code
- Create container registry with **ACR** and **Azure Key Vault**
- Configure CI/CD pipeline for **Webgoat**
- Check open source components with **Whitesource Bolt**
- Scan containers with **Anchore** for the security with **AKS**



# DEMO: Implement Security in Azure DevOps CI/CD

## Install Required CLI's

Azure CLI :

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>

Helm CLI :

<https://helm.sh/docs/intro/install/>

Kubectl :

<https://kubernetes.io/docs/tasks/tools/install-kubectl/>

Anchore CLI :

<https://github.com/anchore/anchore-cli>

Anchore CLI for Mac M1 :

<https://www.dynamsoft.com/codepool/python-barcode-apple-m1-mac.html>



# DEMO: Implement Security in Azure DevOps CI/CD

## Azure DevOps & Azure Subscription Connections

Screenshot of the Azure DevOps Organization Settings page under 'Azure Active Directory'.

The page shows that the organization is connected to the **SRKNBNGL** directory. It displays the tenant ID: f7054334-3dab-4acd-9730-2bb803fc1caf and provides a link to other frequently asked questions. A 'Disconnect directory' button is available if needed.

Below this, there is a section for downloading connected organizations, with a 'Download' button.

The left sidebar lists various settings categories:

- General
- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage
- Extensions
- Azure Active Directory

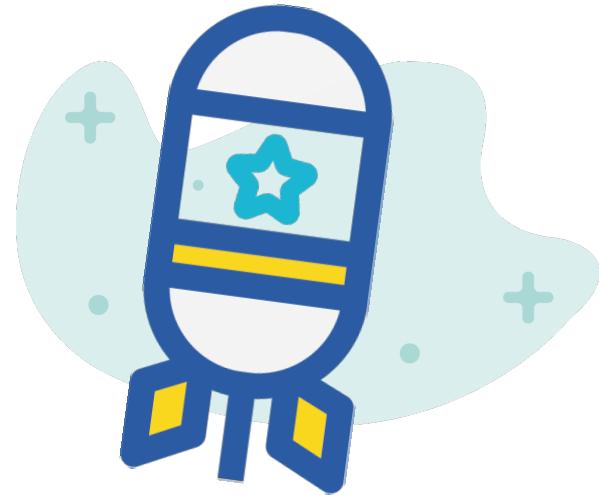
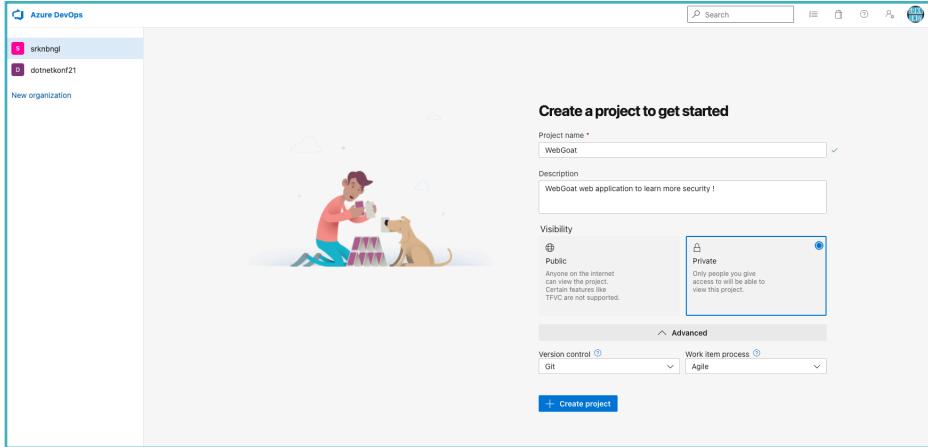


Connect your organization to Azure Active Directory :

<https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/connect-organization-to-azure-ad?view=azure-devops>

# DEMO: Implement Security in Azure DevOps CI/CD

## Create Azure DevOps Project

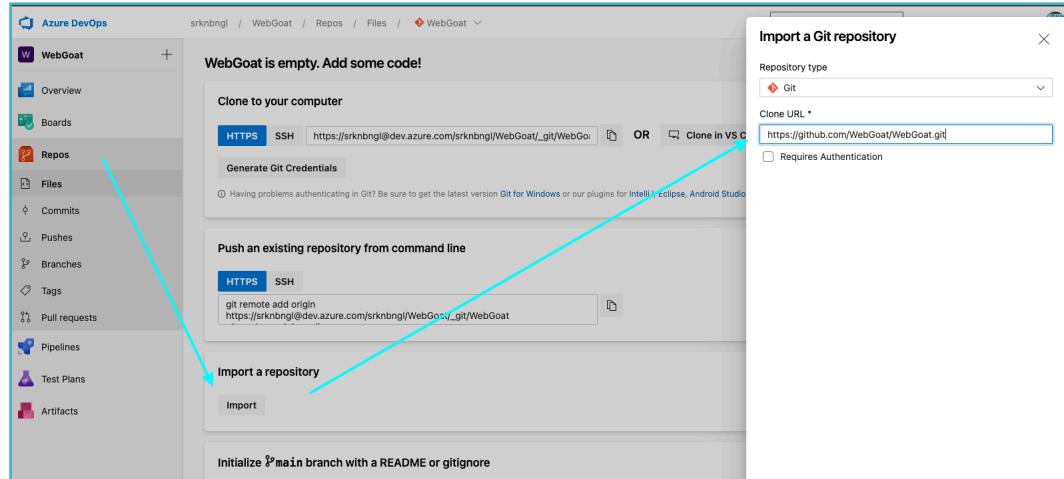


Create a project in Azure DevOps :

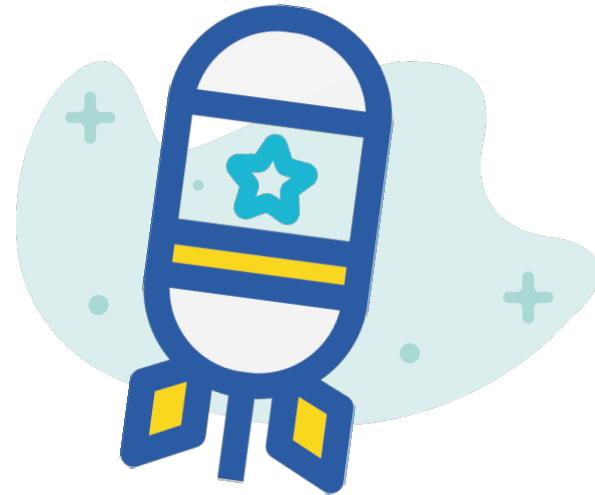
<https://docs.microsoft.com/en-us/azure/devops/organizations/projects/create-project?view=azure-devops&tabs=preview-page>

# DEMO: Implement Security in Azure DevOps CI/CD

## Create a new Git repo in your project



The screenshot shows the Azure DevOps interface for managing repositories. On the left, the 'Repos' section is selected. In the center, there's a modal window titled 'Import a Git repository'. The 'Repository type' dropdown is set to 'Git'. The 'Clone URL' field contains 'https://github.com/WebGoat/WebGoat.git'. There's also a checkbox for 'Requires Authentication' which is unchecked. A cyan arrow points from the 'Import' button in the main 'Repos' area to the 'Clone in VS Code' button in the modal.



Create a new Git repo in your project :

<https://docs.microsoft.com/en-us/azure/devops/repos/git/create-new-repo?view=azure-devops>

WebGoat GitHub project :

<https://github.com/WebGoat/WebGoat>

# DEMO: Implement Security in Azure DevOps CI/CD

## Update WebGoat dockerfile version

srkhngl / WebGoat / Repos / Files / WebGoat

**WebGoat**

- .mvn
- config
- docker
- docs
- platformQuickStarts
- scripts
- webgoat-container
- webgoat-images
- webgoat-integration-tests
- webgoat-lessons
- webgoat-server**

  - src
    - Dockerfile

pom.xml

Committed 76a0ca82: Updated Dockerfile

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3   <artifactId>webgoat-server</artifactId>
4   <packaging>jar</packaging>
5   <parent>
6     <groupId>org.owasp.webgoat</groupId>
7     <artifactId>webgoat-parent</artifactId>
8     <version>0.1.0</version>
9   </parent>
10  <properties>
11    <start-class>org.owasp.webgoat.StartWebGoat</start-
12    </properties>
13    <dependencies>
14      <dependency>
15        <groupId>org.owasp.webgoat</groupId>
16        <artifactId>webgoat-container</artifactId>
17        <version>${project.version}</version>
18      </dependency>
19      <dependency>
20        <groupId>org.owasp.webgoat.lesson</groupId>
21        <artifactId>challenge</artifactId>
22        <version>${project.version}</version>
23      </dependency>
24    </dependencies>
25  </project>

```

master / webgoat-server / Dockerfile

**Dockerfile**

Contents Highlight changes

```

1 FROM openjdk:11.0.1-jre-slim-stretch
2
3 ARG webgoat_versionv0.1.0
4
5 RUN \
6   apt-get update && apt-get install -y \
7   useradd -m -d /home/webgoat --create-home -U webgoat
8
9 USER webgoat
10 RUN cd /home/webgoat/; mkdir -p .webgoat/.version
11cp target/webgoat-server-${webgoat_version}.jar /home/we
12
13 EXPOSE 8080
14
15 WORKDIR /home/webgoat
16 ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/urandom", "-server.port=8080", "-server.address=0.0.0.0"]
17 CMD ["-server.port=8080", "-server.address=0.0.0.0"]
18

```



# DEMO: Implement Security in Azure DevOps CI/CD

## Azure CLI : Login & Set Subscription

```
srknbing1@800 ➜ az login
srknbing1@800 ➜ 
The default web browser has been opened at https://login.microsoftonline.com/common/oauth2/authorize. Please continue the login in the
ils to open, use device code flow with 'az login --use-device-code'.
You have logged in. Now let us find all the subscriptions to which you have access...
[{"cloudName": "AzureCloud", "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "managedByTenants": [], "name": "VSE_Production", "state": "Enabled", "tenantId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "user": {"name": "serkanbingol@outlook.com", "type": "user"}}, {"cloudName": "AzureCloud", "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "managedByTenants": [], "name": "VSE_Test", "state": "Enabled", "tenantId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "user": {"name": "serkanbingol@outlook.com", "type": "user"}}
]
srknbing1@800 ➜ az account show --output table
EnvironmentName  HomeTenantId  IsDefault  Name  State  TenantId
AzureCloud  XXXXX  True  VSE_Test  Enabled  XXXXX
srknbing1@800 ➜
```



Sign in with Azure CLI :

<https://docs.microsoft.com/en-us/cli/azure/authenticate-azure-cli>

# DEMO: Implement Security in Azure DevOps CI/CD

## Azure CLI : Create Resource Group

```
srknbnlg1@t-800 ~ az group create --name webGoatKonfRG --location westeurope

{
  "id": "/subscriptions/[REDACTED]/resourceGroups/webGoatKonfRG",
  "location": "westeurope",
  "managedBy": null,
  "name": "webGoatKonfRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```



### azure-cli command

```
az group create --name webGoatKonfRG --location westeurope
```

Create Azure Resource Group :

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-cli>

# DEMO: Implement Security in Azure DevOps CI/CD

## Azure CLI : Create Azure Container Registry

```
srknbg1@t-800 ~ % az acr create --resource-group webGoatKonfRG --name webGoatKonfACR --sku Basic --admin-enabled true --location westeurope
{
  "id": "/subscriptions/.../resourceGroups/webGoatKonfRG/providers/Microsoft.ContainerRegistry/registries/webGoatKonfACR",
  "location": "westeurope",
  "name": "webGoatKonfACR",
  "resourceGroup": "webGoatKonfRG",
  "status": "Enabled"
}
```



### azure-cli command

```
az acr create --resource-group webGoatKonfRG --name webGoatKonfACR --sku Basic --admin-enabled true --location westeurope
```

Create Azure Container Registry :

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-cli>

# DEMO: Implement Security in Azure DevOps CI/CD

## Azure CLI : Azure Key Vault

```
srknbnlg1@t-800 ~ % az keyvault create --name "webGoatKonfKV" --resource-group "webGoatKonfRG" --location westeurope
{- Finished ..
{"id": "/subscriptions/.../resourceGroups/webGoatKonfRG/providers/Microsoft.KeyVault/vaults/webGoatKonfKV",
"location": "westeurope",
"name": "webGoatKonfKV",
"properties": {
"accessPolicies": [
{
"applicationId": null,
"objectId": "...",
"permissions": {
"certificates": [
"get",
"get",
"list"
]
```



### azure-cli command

```
az keyvault create --name "webGoatKonfKV" --resource-group "webGoatKonfRG" --location westeurope
```

Create Azure Key Vault :

<https://docs.microsoft.com/bs-cyrl-ba/Azure/key-vault/general/quick-create-cli>

# DEMO: Implement Security in Azure DevOps CI/CD

## Check Azure Resources - Resource Group , ACR , Key Vault

Screenshot of the Azure portal showing the 'webGoatKonfRG' Resource Group details.

**Essentials:**

- Subscription (change) : VSE\_Test
- Subscription ID : 4... (redacted)
- Deployments : No deployments
- Location : West Europe

**Tags (change) :** Click here to add tags

**Filter for any field...** Type == all Location == all Add filter

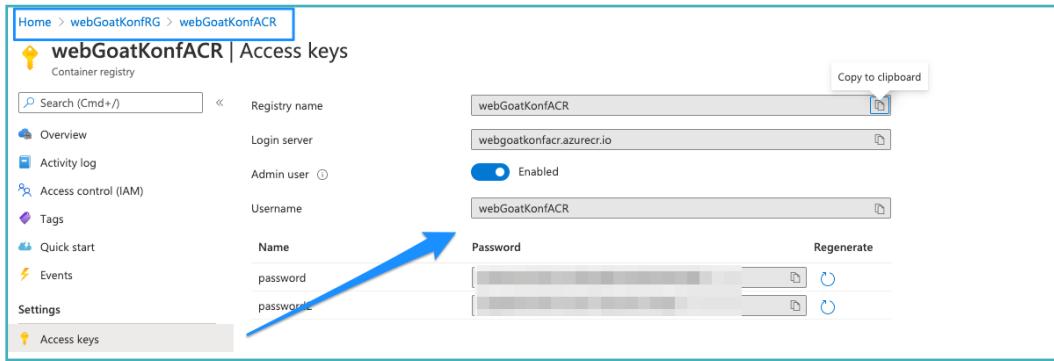
Showing 1 to 2 of 2 records. Show hidden types

Name	Type	Location
webGoatKonfACR	Container registry	West Europe
webGoatKonfKV	Key vault	West Europe



# DEMO: Implement Security in Azure DevOps CI/CD

## Get Azure CR Credentials



The screenshot shows the 'Access keys' section of the Azure Container Registry (ACR) interface. It displays the following information:

- Registry name: webGoatKonfACR
- Login server: webgoatkonfac.azurecr.io
- Admin user: Enabled (checkbox)
- Username: webGoatKonfACR
- Name: password
- Password: password
- Regenerate: (button)

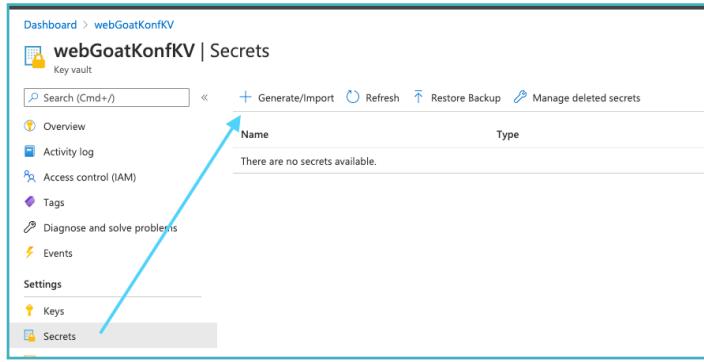
A blue arrow points from the text "Copy to clipboard" at the top right towards the 'password' field.



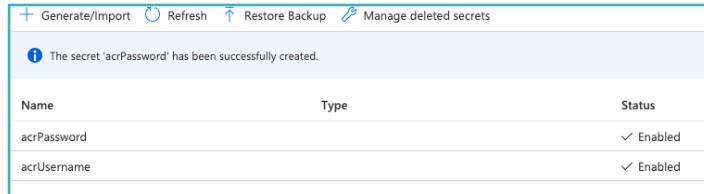
Azure Container Registry Documentation :  
<https://docs.microsoft.com/en-us/azure/container-registry/>

# DEMO: Implement Security in Azure DevOps CI/CD

## Create secrets Azure Key Vault



The screenshot shows the 'Secrets' blade in the Azure Key Vault interface. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Keys, and Secrets. The main area displays a table with columns for Name and Type, stating 'There are no secrets available.'



The screenshot shows the 'Secrets' blade after creating a new secret. A message at the top says 'The secret 'acrPassword' has been successfully created.' Below is a table with two rows:

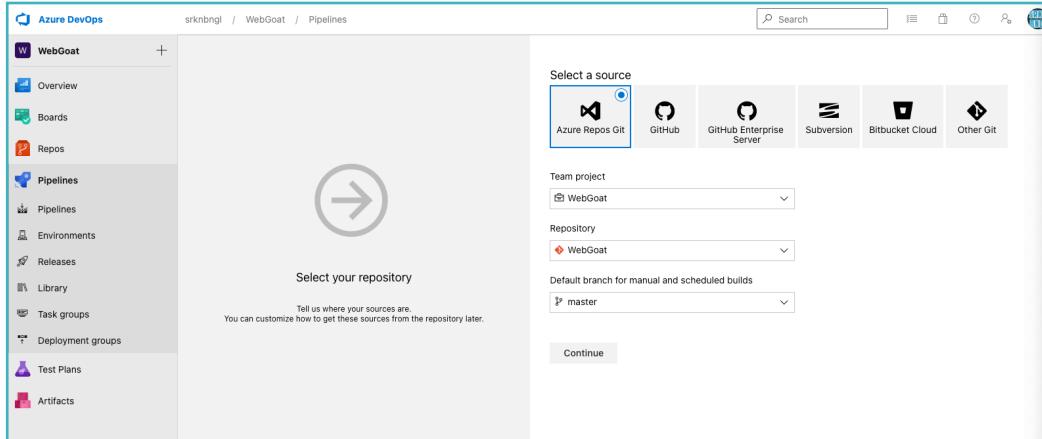
Name	Type	Status
acrPassword		✓ Enabled
acrUsername		✓ Enabled



Use **acrUsername** and **acrPassword** names and set ACR credentials as value.

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Build Pipeline



The screenshot shows the 'Select a source' step in the Azure DevOps Pipelines interface. On the left, a sidebar lists 'WebGoat' under 'Pipelines'. The main area has a large circular arrow icon. Below it, the text 'Select your repository' and 'Tell us where your sources are. You can customize how to get these sources from the repository later.' A 'Continue' button is at the bottom. To the right, a 'Select a source' section shows 'Azure Repos Git' selected, with other options like GitHub, Bitbucket Cloud, and Subversion available.



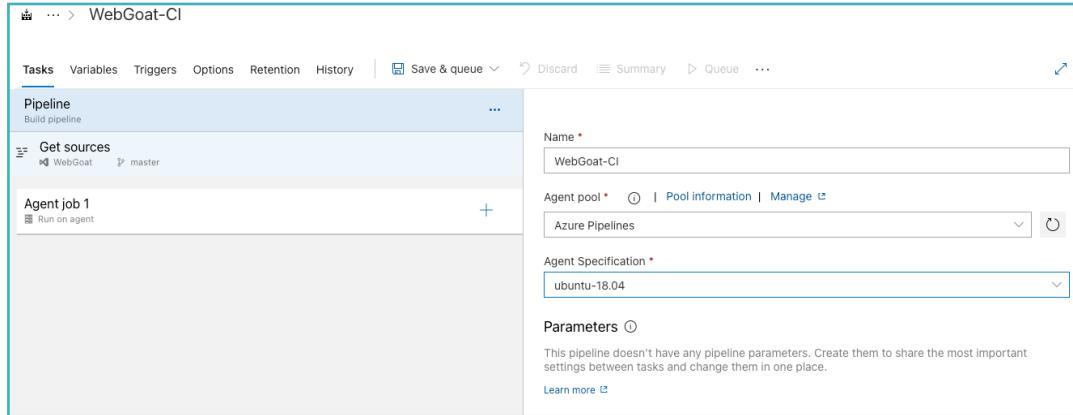
Use the **classic editor** to create a pipeline without YAML.

Create your first pipeline :

<https://docs.microsoft.com/en-us/azure/devops/pipelines/create-first-pipeline?view=azure-devops&tabs=java%2Ctfs-2018-2%2Cbrowser>

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Build Pipeline



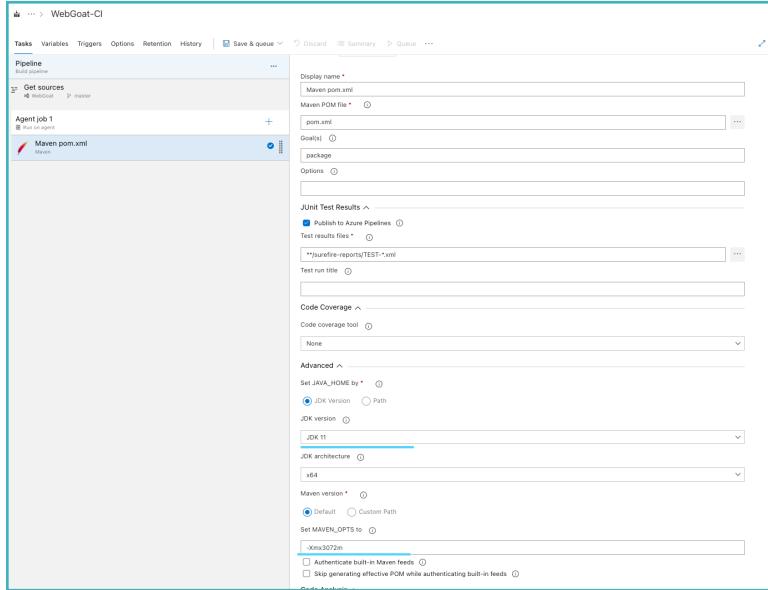
The screenshot shows the 'Pipeline' configuration page for a build pipeline named 'WebGoat-Cl'. The pipeline consists of a single 'Agent job 1' which runs on an 'Empty job'. The job has a task named 'Get sources' which checks out the 'WebGoat' repository from the 'master' branch. The pipeline is set to run on the 'Azure Pipelines' agent pool and uses the 'ubuntu-18.04' agent specification.



Use **Empty job** to start. Select ubuntu-18.04 agent for agent job.

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Build Pipeline

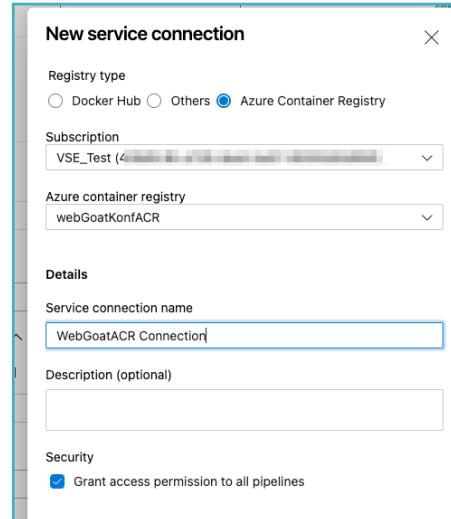
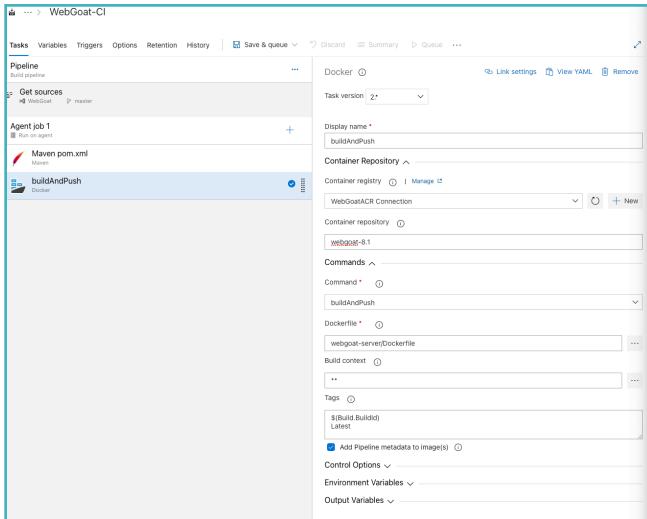


Add **Maven** task to then “Advanced” step set MAVEN\_OPT to **-Xmx3072m** .



# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Build Pipeline



Add **Docker** task. Select **webgoat-server** folder dockerfile to build. Add **Latest** to tags. Create new Container Registry connection with **New** button. Last step **Save & Queue** .

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Build Pipeline

#1 Updated Dockerfile on WebGoat-CI

Cancel ⋮

Summary

Manually run by Serkan Bingöl

Repository and version

WebGoat master ↗ 76a0ca8

Time started and elapsed

Just now 20s

Related

0 work items 1 consumed

Tests and coverage

Get started

View 250 changes

Jobs

Name	Status	Duration
Agent job 1	Running	18s

← Jobs in run #1 WebGoat-CI

Agent job 1

Pool: Azure Pipelines

Image: ubuntu-18.04

Agent: Hosted Agent

Started: Today at 2:48 PM

Duration: 5m 41s

Job preparation parameters

1 queue time variable used

99.5% tests passed

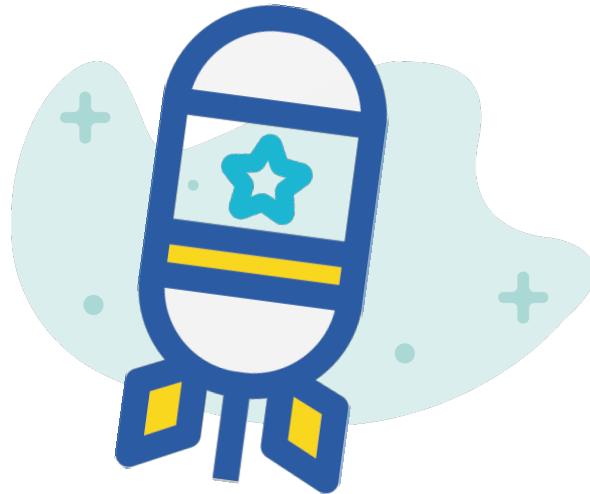
Job live console data:

Finishing: Agent job 1

View raw log

Jobs

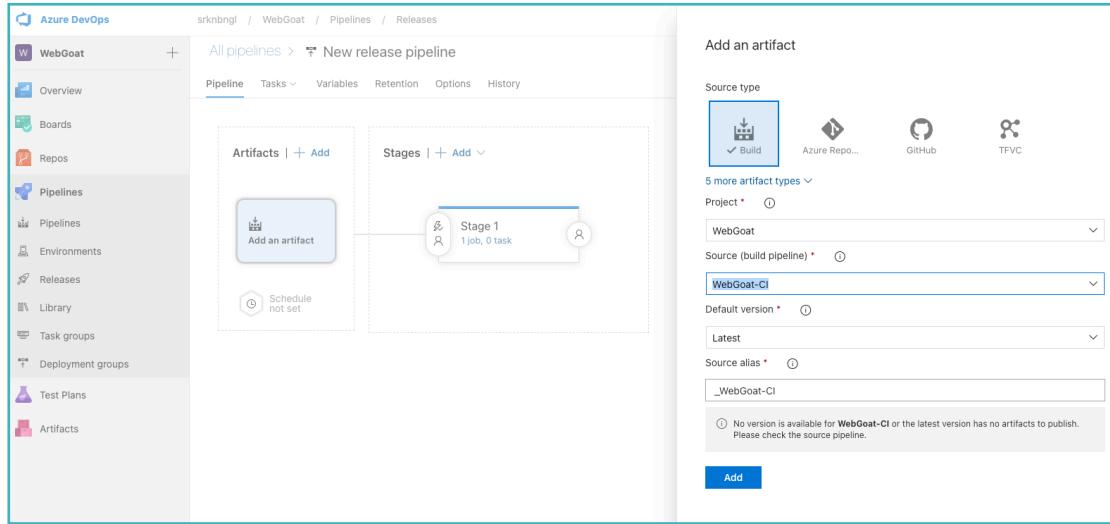
- Agent job 1 5m 41s
- Initialize job 3s
- Checkout WebGoat@... 7s
- Maven pom.xml 4m 33s
- buildAndPush 57s
- Post-job: Checkout ... <1s
- Finalize Job <1s
- Report build status <1s



Build will take approximately 5 mins then check build pipeline finished with successful.

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Release Pipeline : Add Artifact



The screenshot shows the Azure DevOps Pipelines interface for the 'WebGoat' project. On the left, the sidebar is open with 'Pipelines' selected. The main area shows a pipeline structure with 'Artifacts' and 'Stages'. A 'Stage 1' stage is visible with '1 job, 0 task'. A modal window titled 'Add an artifact' is open, showing the 'Source type' section. Under 'Source type', 'Build' is selected, indicated by a checked checkbox. Other options include 'Azure Repo...', 'GitHub', and 'TFVC'. Below this, there are fields for 'Project' (set to 'WebGoat'), 'Source (build pipeline)' (set to 'WebGoat-CI'), 'Default version' (set to 'Latest'), and 'Source alias' (set to '\_WebGoat-CI'). At the bottom of the modal is a blue 'Add' button.

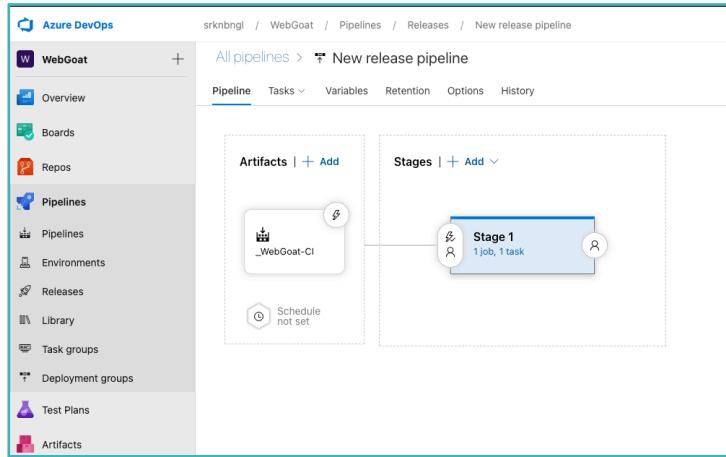


Define your multi-stage continuous deployment (CD) pipeline :

<https://docs.microsoft.com/en-us/azure/devops/pipelines/release/define-multistage-release-process?view=azure-devops>

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Release Pipeline : Add Stage



The screenshot shows the Azure DevOps Pipelines interface for creating a new release pipeline. On the left, the sidebar lists various options like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The 'Pipelines' option is selected. The main area shows a 'New release pipeline' configuration. It includes sections for 'Artifacts' (containing '\_WebGoat-CI') and 'Stages' (containing 'Stage 1' which has '1 job, 1 task'). A note says 'Schedule not set'.

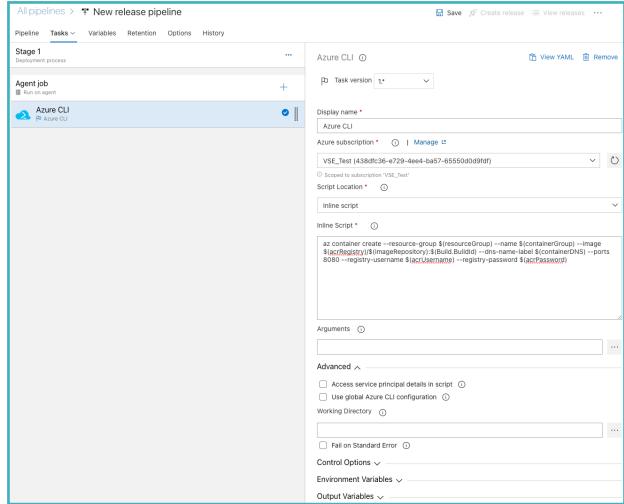


Define your multi-stage continuous deployment (CD) pipeline :

<https://docs.microsoft.com/en-us/azure/devops/pipelines/release/define-multistage-release-process?view=azure-devops>

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Release Pipeline : Add Stage Tasks



The screenshot shows the 'Tasks' tab of a release pipeline named 'New release pipeline'. It contains one stage, 'Stage 1', which has a single 'Deployment process' step. This step includes an 'Agent job' set to 'Run on agent' and an 'Azure CLI' task. The task is titled 'Azure CLI' and has the following configuration:

- Display name:** Azure CLI
- Azure subscription:** VSE\_Test (1380d3c6-e729-4ee4-ba57-65550d09ffdf)
- Script Location:** Inline script
- Inline Script:**

```
az container create --resource-group $(resourceGroup) --name $(containerGroup) --image $(acrRegistry)$(imageRepository):$(Build.BuildId) --dns-name-label $(containerDNS) --ports 8080 --registry-username $(acrUsername) --registry-password $(acrPassword)
```
- Arguments:** (empty)
- Advanced:**
  - Access service principal details in script
  - Use global Azure CLI configuration
- Working Directory:** (empty)
- Control Options:** (empty)
- Environment Variables:** (empty)
- Output Variables:** (empty)

### azure-cli command

```
az container create --resource-group $(resourceGroup) --name $(containerGroup) --image $(acrRegistry)$(imageRepository):$(Build.BuildId) --dns-name-label $(containerDNS) --ports 8080 --registry-username $(acrUsername) --registry-password $(acrPassword)
```

Add **Azure CLI** task. Select **Azure Subscription** created before. Select **Inline Script**. Add cli command given above to **Inline Script** area.

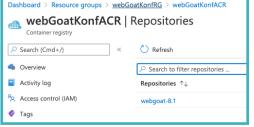


# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Release Pipeline : Add Stage Variables - Non-sensitive



Resource group: webGoatKonfRG  
 Location: West Europe  
 SKU: Basic  
 Provisioning state: Succeeded



Repositories: webgoat-8.1

Pipeline variables

Name	Value	Scope
resourceGroup	webGoatKonfRG	Release
containerGroup	webgoatkonfcg	Release
acrRegistry	webgoatkonfacr.azurecr.io	Release
containerDNS	webgoatkonfdns	Release
imageRepository	webgoat-8.1	Release

[+ Add](#)

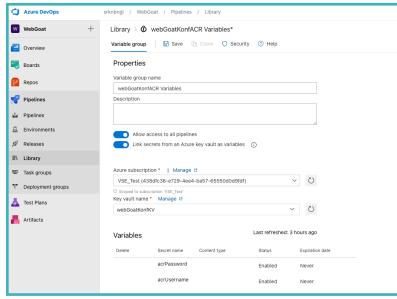
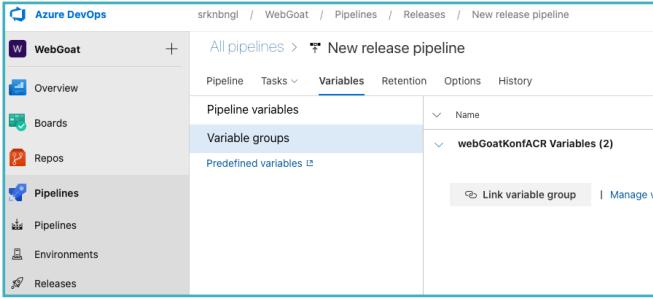


Get values from **webGoatKonfACR** named container registry to use in release pipeline variables.

In variables section for non-sensitive variables add **resourceGroup** , **containerGroup** , **acrRegistry** , **containerDNS** , **imageRepository** names with values of **webGoatKonfRG** , **webgoatkonfcg** , **webgoatkonfacr.azurecr.io** , **webgoatkonfdns** , **webgoat-8.1** in a row under Pipeline variables area.

# DEMO: Implement Security in Azure DevOps CI/CD

## Configure Release Pipeline : Add Stage Variables - Sensitive

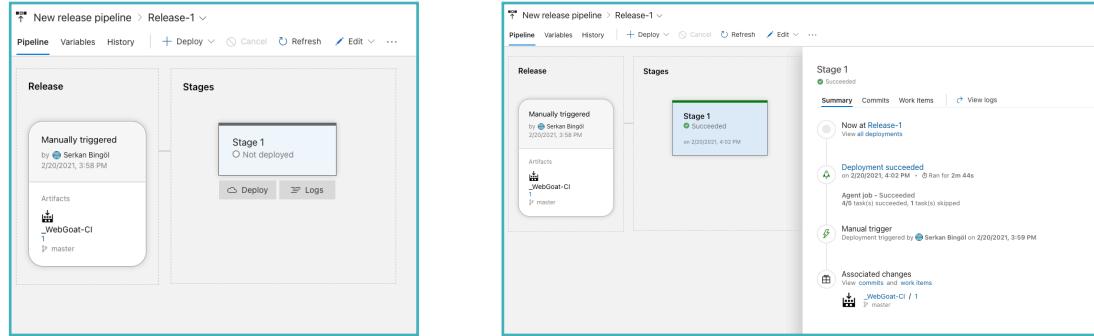




Create Variable Group from Azure Key Vault under library menu from Pipelines section.

For sensitive variables select variable groups under release pipeline variables title and link it with created from Azure Key Vault service.

# DEMO: Implement Security in Azure DevOps CI/CD

## Create Release



The screenshot shows the Azure DevOps Release Pipeline interface. On the left, under 'Release', there is a 'Manually triggered' card for 'Release-1' by 'Serkan Bingöl' at 2/20/2021, 3:58 PM. Below it, an artifact named '\_WebGoat-CI' is listed. On the right, under 'Stages', a 'Stage 1' card is shown with the status 'Not deployed'. A 'Deploy' button is visible next to it. The overall interface is clean and modern, typical of Microsoft's product design.

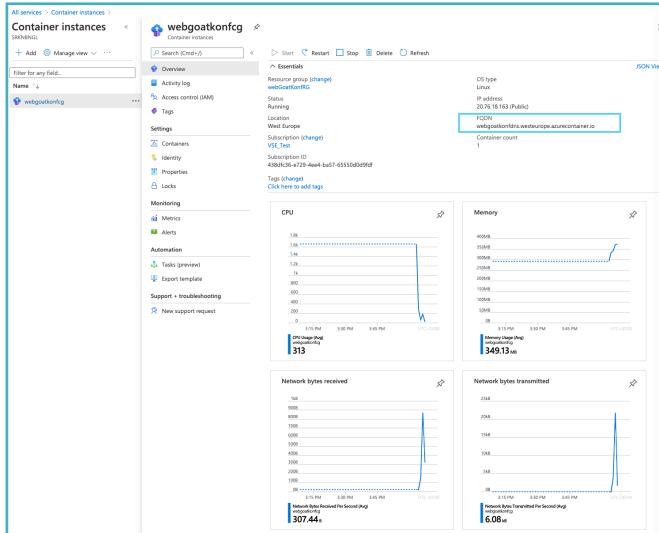


Create a new release under release pipeline by clicking **Create New Release** button.

You need a manually approve to deploy created artifact by pushing deploy button in Stage 1

# DEMO: Implement Security in Azure DevOps CI/CD

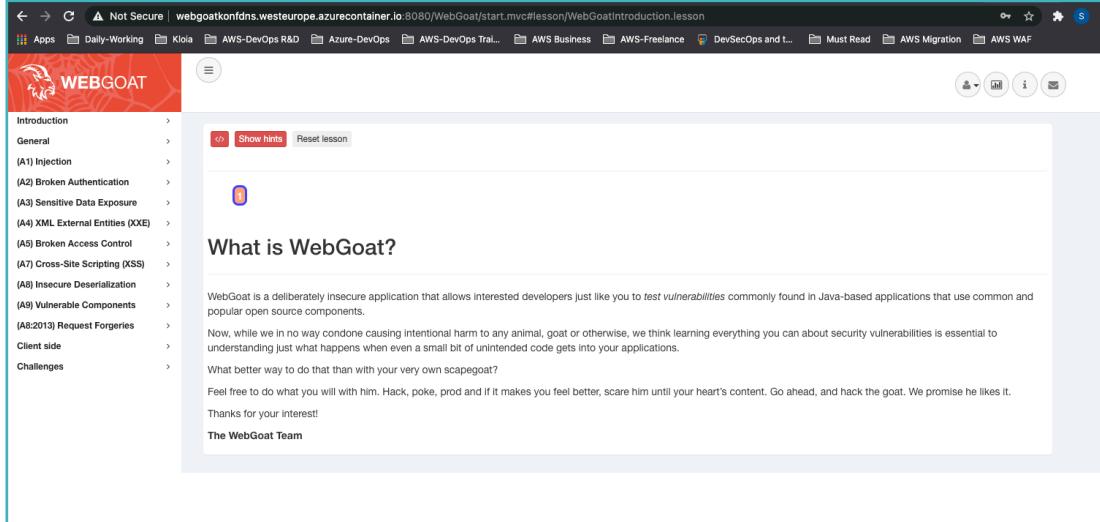
## Check Azure Container Instances



After create a successful deployment go to Azure Container Instance service to get our container published web link.

# DEMO: Implement Security in Azure DevOps CI/CD

## Browse WebGoat and Learn Security !



The screenshot shows a web browser window with the URL <http://webgoatkondns.westeurope.azurecontainer.io:8080/WebGoat/start.mvc#lesson/WebGoatIntroduction.lesson>. The page title is "Not Secure | webgoatkondns.westeurope.azurecontainer.io:8080/WebGoat/start.mvc#lesson/WebGoatIntroduction.lesson". The left sidebar menu includes categories like Introduction, General, and various types of injection attacks (A1 to A8). The main content area displays the "What is WebGoat?" lesson, which describes it as a deliberately insecure application for testing vulnerabilities. It features a small icon of a goat and a "Show hints" button.



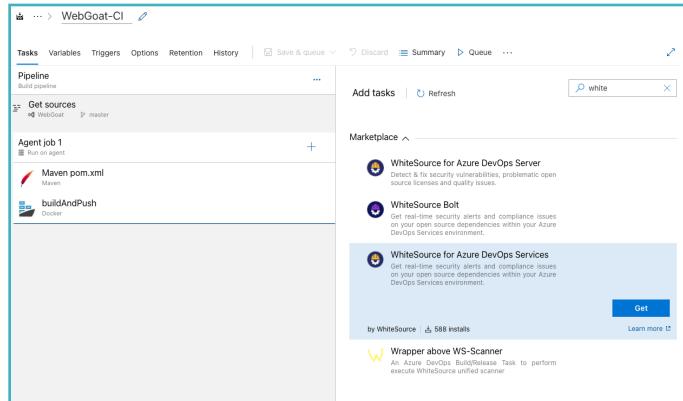
WebGoat Container Browse Link : [http://ACI\\_FQDN:8080/WebGoat](http://ACI_FQDN:8080/WebGoat)

WebGoat Official Page : <https://owasp.org/www-project-webgoat/>

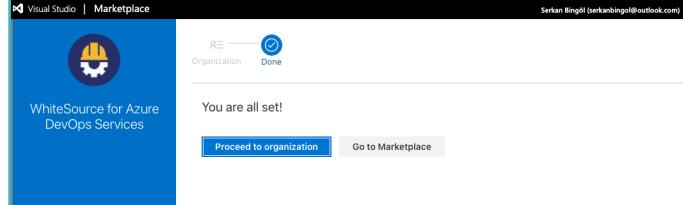
\*ACI\_FQDN is our container instance FQDN

# DEMO: Implement Security in Azure DevOps CI/CD

## Whitesource Bolt: check open source components



The screenshot shows the Azure DevOps Pipeline Editor for a pipeline named "WebGoat-CI". On the left, there's a "Get sources" step and an "Agent job 1" step containing "Maven pom.xml" and "buildAndPush". In the center, the "Add tasks" dialog is open with a search bar containing "white". Two results are shown: "WhiteSource for Azure DevOps Server" and "WhiteSource Bolt". The "WhiteSource Bolt" result is selected, showing its details: "Get real-time security alerts and compliance issues on your open source dependencies within your Azure DevOps Services environment.", "by WhiteSource", "588 installs", and a "Get" button. Below it, a note says "Wrapper above WS-Scanner An Azure DevOps Build/Release Task to perform execute Whitesource unified scanner".

The screenshot shows the Visual Studio Marketplace page for "WhiteSource for Azure DevOps Services". It displays the product icon, name, developer information ("Serkan Bingöl (serkanbingol@outlook.com)"), and a message saying "You are all set!". At the bottom are two buttons: "Proceed to organization" and "Go to Marketplace".



Add Whitesource Bolt extension to build pipeline from Visual Studio Marketplace

# DEMO: Implement Security in Azure DevOps CI/CD

## Whitesource Bolt: check open source components

Screenshot of the Azure DevOps Pipeline editor showing a pipeline configuration:

- Pipeline**: Build pipeline
- Get sources**: WebGoat, master
- Agent job 1**: Run on agent
- WhiteSource**: WhiteSource (selected)
- Maven pom.xml**: Maven
- buildAndPush**: Docker

The WhiteSource task configuration pane is open:

- Task version: 21.\*
- Display name: WhiteSource
- Root working directory: \$(System.DefaultWorkingDirectory)
- Project name: (empty)
- WhiteSource Configuration: (empty)
- Control Options: (empty)
- Output Variables: (empty)

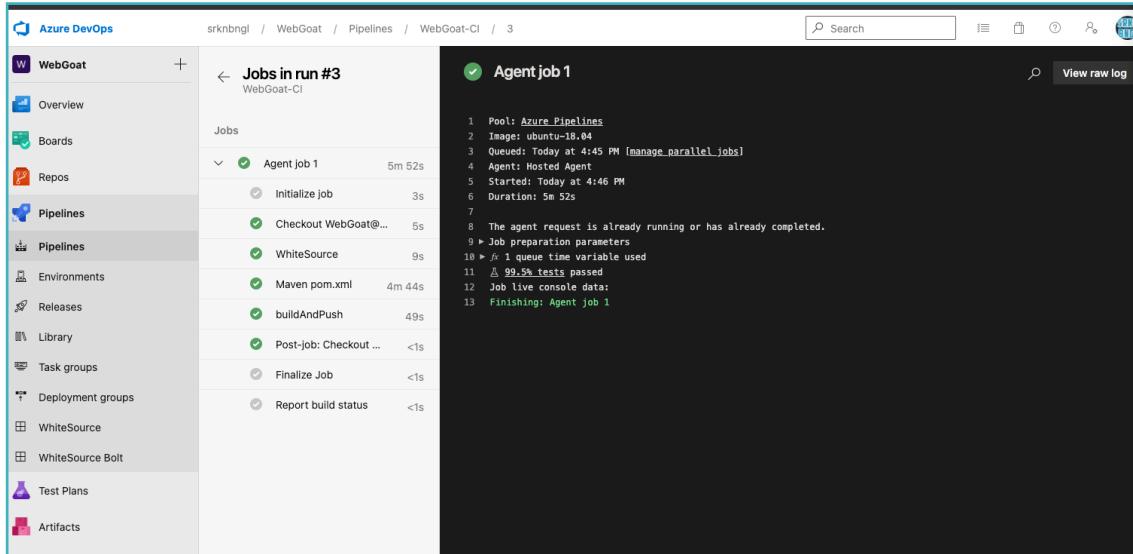


WhiteSource Bolt Documentation :

<https://marketplace.visualstudio.com/items?itemName=whitesource.ws-bolt>

# DEMO: Implement Security in Azure DevOps CI/CD

## Whitesource Bolt: check open source components



The screenshot shows the Azure DevOps interface for a project named 'WebGoat'. The left sidebar is filled with various navigation items like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, WhiteSource, WhiteSource Bolt, Test Plans, and Artifacts. The 'Pipelines' item is currently selected. In the center, a pipeline named 'WebGoat-CI' is displayed under the heading 'Jobs in run #3'. This pipeline contains one job named 'Agent job 1', which has a duration of 5m 52s. The steps within this job are listed as follows:

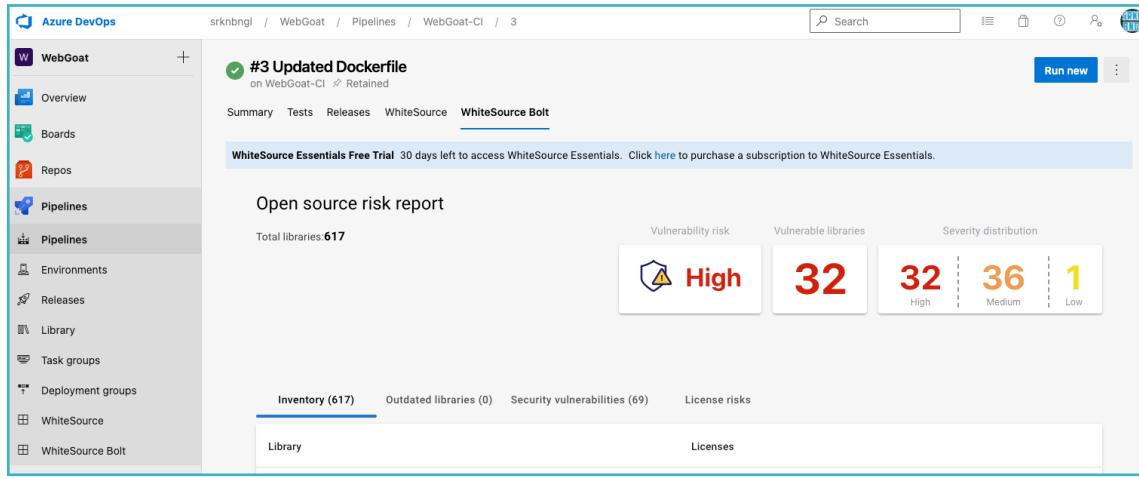
```
1 Pool: Azure Pipelines
2 Image: ubuntu-18.04
3 Queued: Today at 4:45 PM [manage_parallel_jobs]
4 Agent: Hosted Agent
5 Started: Today at 4:46 PM
6 Duration: 5m 52s
7
8 The agent request is already running or has already completed.
9 ▶ Job preparation parameters
10 ▶ 1 queue time variable used
11 ▶ 99.5% tests passed
12 Job live console data:
13 Finishing: Agent job 1
```



After adding Whitesource Bolt extension rebuild pipeline to get reports.

# DEMO: Implement Security in Azure DevOps CI/CD

## Whitesource Bolt: Check Vulnerability Reports



The screenshot shows the Azure DevOps Pipelines interface for a project named "WebGoat". The pipeline step "WebGoat" has completed successfully, indicated by a green checkmark and the message "#3 Updated Dockerfile on WebGoat-CI Retained". The "WhiteSource Bolt" tab is selected, displaying a "WhiteSource Essentials Free Trial" banner with 30 days left. Below it is an "Open source risk report" section. Key statistics shown include "Total libraries: 617", "Vulnerability risk: High: 32", "Vulnerable libraries: 32 (High), 36 (Medium), 1 (Low)", and "Severity distribution: High (32), Medium (36), Low (1)". Navigation tabs at the bottom include "Inventory (617)" (selected), "Outdated libraries (0)", "Security vulnerabilities (69)", and "License risks". Buttons for "Library" and "Licenses" are also present.



After build pipeline Whitesource Bolt extension creates vulnerability reports.

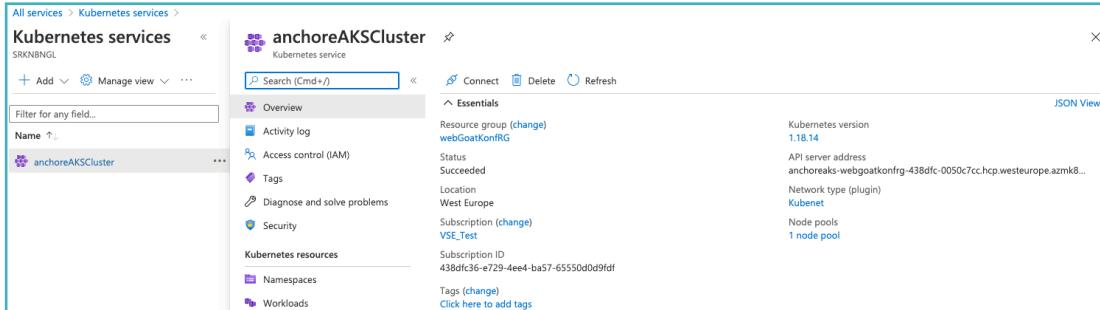
# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Create AKS Cluster on WebGout Resource Group

```
srknbg1@t-800 ~ % az aks create --resource-group webGoatKonfRG --name anchoreAKScluster --node-count 3 --enable-addons monitoring --generate-ssh-keys
AAD role propagation done[########################################] 100.0000%
{
  "aadProfile": null,
  "addonProfiles": {
    "KubeDashboard": {
      "config": null,
      "enabled": false,
      "identity": null
    }
  }
}
```

### azure-cli command

az aks create --resource-group webGoatKonfRG --name anchoreAKScluster --node-count 3 --enable-addons monitoring --generate-ssh-keys



The screenshot shows the Azure portal interface for managing Kubernetes services. On the left, there's a navigation bar with 'All services > Kubernetes services'. Below it, a search bar and a 'Manage view' dropdown are visible. A filter bar allows filtering by name or tags. In the main area, a list of Kubernetes services is shown, with 'anchoreAKScluster' selected. The right-hand pane displays detailed information about the selected service:

- Overview** tab is selected.
- Resource group**: webGoatKonfRG
- Kubernetes version**: 1.18.14
- Status**: Succeeded
- Location**: West Europe
- Subscription**: VSE\_Test
- Node pools**: 1 node pool
- Tags**: Click here to add tags
- Kubernetes resources** section lists Namespaces and Workloads.



Install Anchore server on AKS with Helm :

<https://anchore.com/blog/azure-anchore-kubernetes-service-cluster-with-helm/>

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Get credentials for AKS cluster and check credentials

```
srknbnlg1@800 ~ $ az aks get-credentials --resource-group webGoatKonfRG --name anchoreAKSCluster
Merged "anchoreAKSCluster" as current context in /Users/srknbnlg1/.kube/config
srknbnlg1@800 ~ $ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
aks-nodepool1-80324486-vmss000000  Ready    agent   12m   v1.18.14
aks-nodepool1-80324486-vmss000001  Ready    agent   12m   v1.18.14
aks-nodepool1-80324486-vmss000002  Ready    agent   12m   v1.18.14
srknbnlg1@800 ~ $ az aks browse --resource-group webGoatKonfRG --name anchoreAKSCluster
Kubernetes resources view on https://portal.azure.com/#resource/subscriptions/438dfc36-e729-4ee4-ba57-65550d
ter/workloads
srknbnlg1@800 ~ $
```



### cli commands

**Get credentials** : az aks get-credentials --resource-group webGoatKonfRG --name anchoreAKSCluster

**Get nodes** : kubectl get nodes

**Browse aks cluster** : az aks browse --resource-group webGoatKonfRG --name anchoreAKSCluster

Name	Namespace	Ready	Up-to-date	Available	Age
coredns-autoscaler	kube-system	1/1	1	1	14 minutes
coredns	kube-system	2/2	2	2	14 minutes
metrics-server	kube-system	1/1	1	1	14 minutes
omsagent-rs	kube-system	1/1	1	1	14 minutes
tunnelfront	kube-system	1/1	1	1	14 minutes

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : HELM Configuration

```
srknbnlg1@t-800 ~$ sudo vim helm-rbac.yaml
Password:
srknbnlg1@t-800 ~$ kubectl apply -f helm-rbac.yaml
serviceaccount/tiller created
clusterrole.rbac.authorization.k8s.io/kube-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/tiller created
clusterrolebinding.rbac.authorization.k8s.io/rook-operator created
srknbnlg1@t-800 ~$
```

- create a file name helm-rbac.yaml
- execute **sudo vim helm-rbac.yaml**
- add configuration same as given code block
- execute **kubectl apply - f helm-brac.yaml** command

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kube-dashboard
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: rook-operator
  namespace: rook-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kube-dashboard
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system
```



# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Install Anchore

```

srknbg10t-800 ➔ helm repo add anchore https://charts.anchore.io
"anchore" already exists with the same configuration, skipping
srknbg10t-800 ➔ helm install anchore-demo anchore/anchore-engine
NAME: anchore-demo
LAST DEPLOYED: Sat Feb 20 17:39:42 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
To use Anchore Engine you need the URL, username, and password to access the API.

Anchore Engine can be accessed via port 8228 on the following DNS name from within the cluster:
anchore-demo-anchore-engine-api.default.svc.cluster.local

Here are the steps to configure the anchore-cli ('pip install anchorecli'). Use these same values for direct API access as well.

To configure your anchore-clc run:
  ANCHORE_CLI_USER=admin
  ANCHORE_CLI_PASS=$(kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath='{.data.ANCHORE_ADMIN_PASSWORD}' | base64 --decode; echo)

using the service endpoint from within the cluster you can use:
  ANCHORE_CLI_URL=http://anchore-demo-anchore-engine-api.default.svc.cluster.local:8228/v1/

```



### cli commands

**Add anchore helm repo** : helm repo add anchore https://charts.anchore.io

**Install anchore helm chart** : helm install anchore-demo anchore/anchore-engine

```

srknbg10t-800 ➔ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
anchore-demo-anchore-analyzer   1/1     1           1           4m50s
anchore-demo-anchore-engine-api 1/1     1           1           4m50s
anchore-demo-anchore-engine-catalog 1/1     1           1           4m50s
anchore-demo-anchore-engine-policy 1/1     1           1           4m50s
anchore-demo-anchore-engine-simplequeue 1/1     1           1           4m50s
anchore-demo-postgresql         1/1     1           1           4m50s
srknbg10t-800 ➔ kubectl expose deployment anchore-demo-anchore-engine-api --type=LoadBalancer --name=anchore-engine --port=8228
service/anchore-engine exposed
srknbg10t-800 ➔ kubectl get service anchore-engine
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
anchore-engine LoadBalancer 10.0.180.42  20.73.32.172  8228:30477/TCP  46s
srknbg10t-800 ➔

```

### cli commands

**Get deployments** : kubectl get deployments

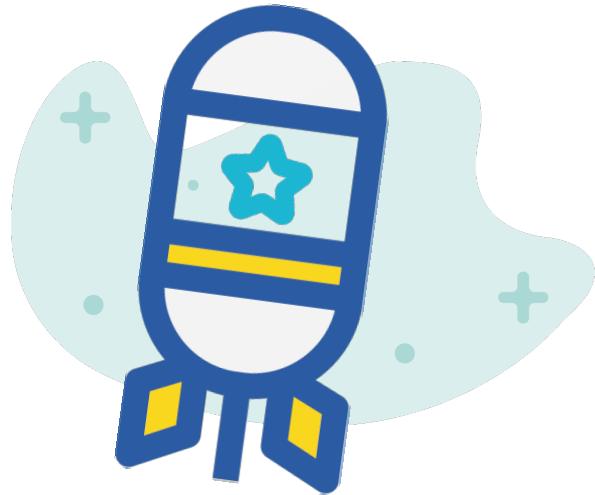
**Expose API port externally** : kubectl expose deployment anchore-demo-anchore-engine-api --type=LoadBalancer --name=anchore-engine --port=8228

**View service and external IP** : kubectl get service anchore-engine

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Check Anchore Status

```
srknbg10t-800 ➜ /usr/local/bin/anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p $(kubectl get secret default anchore-demo-anchore-engine -o jsonpath='{.data.ANCHORE_ADMIN_PASSWORD}' | base64 --decode; echo) system status
de: ) system status
Service anchore-demo-anchore-engine-catalog-569b48c58d-edps, http://anchore-demo-anchore-engine-catalog:8082): up
Service analyzer (anchore-demo-anchore-engine-analyzer-7fd5544998-75ph7, http://anchore-demo-anchore-engine-analyzer:8084): up
Service policy_engine (anchore-demo-anchore-engine-policy-56dc5d4448-fmkxn, http://anchore-demo-anchore-engine-policy:8087): up
Service simplequeue (anchore-demo-anchore-engine-simplequeue-66cbcbd965-lmf9w, http://anchore-demo-anchore-engine-simplequeue:8083): up
Service apiext (anchore-demo-anchore-engine-api-9b694647b-dpjz1, http://anchore-demo-anchore-engine-api:8228): up
Engine DB Version: 0.0.14
Engine Code Version: 0.9.1
srknbg10t-800 ➜ [ 468 18:12:51 ]
```



### cli commands

**Check status of Anchore :** anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p \$(kubectl get secret default anchore-demo-anchore-engine -o jsonpath='{.data.ANCHORE\_ADMIN\_PASSWORD}' | base64 --decode; echo) system status

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Add Registry and Image to Anchore

```
srknbnlg1@t-800 ➜ ~ anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p $(kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath=".data.ANCHORE_ADMIN_PASSWORD" | base64 --decode; echo) registry add --registry-type docker_v2 webgoatkonfacr.azurecr.io webGoatKonfACR LHZiFHz/YeuQL=wQhHgRGva5MDOMCWdE
Registry: webgoatkonfacr.azurecr.io
Name: webgoatkonfacr.azurecr.io
User: webGoatKonfACR
Type: docker_v2
Verify TLS: True
Created: 2021-02-20T15:27:29Z
Updated: 2021-02-20T15:27:29Z

srknbnlg1@t-800 ➜ 470 18:27:29
```

### cli commands

**Add Registry to Anchore :** anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p \$(kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath=".data.ANCHORE\_ADMIN\_PASSWORD" | base64 --decode; echo) registry add --registry-type docker\_v2 webgoatkonfacr.azurecr.io webGoatKonfACR LHZiFHz/YeuQL=wQhHgRGva5MDOMCWdE

```
srknbnlg1@t-800 ➜ ~ anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p $(kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath=".data.ANCHORE_ADMIN_PASSWORD" | base64 --decode; echo) image add webgoatkonfacr.azurecr.io/webgoat-8.1:Latest
Image Digest: sha256:e79e82deec6a74ef1fb895e52a765b88e0b63e6593638d5c4f7dec31911ac72
Parent Digest: sha256:e79e82deec6a74ef1fb895e52a765b88e0b6d63e503638d5c4f7dec31911ac72
Analysis Status: not_analyzed
Image Type: docker
Analyzed At: None
Image ID: 66f057b1e8282fbc0de37408fd457e59e217e417e4bf9ab97a2578acf891f4d
Dockerfile Mode: None
Distro: None
Distro Version: None
Size: None
Architecture: None
Layer Count: None

Full Tag: webgoatkonfacr.azurecr.io/webgoat-8.1:Latest
Tag Detected At: 2021-02-20T15:31:02Z

srknbnlg1@t-800 ➜ 472 18:31:02
```

### cli commands

**Add Image to Anchore :** anchore-cli --url http://20.73.32.172:8228/v1 --u admin --p \$(kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath=".data.ANCHORE\_ADMIN\_PASSWORD" | base64 --decode; echo) image add webgoatkonfacr.azurecr.io/webgoat-8.1:Latest



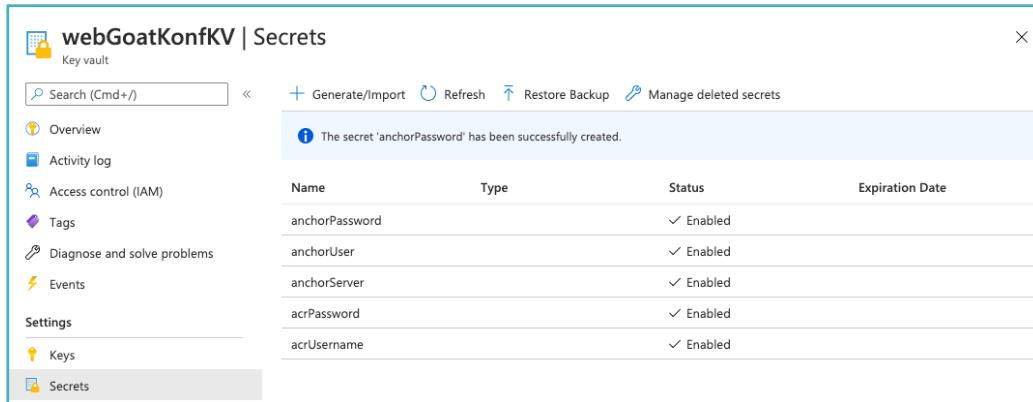
# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Add credentials to Azure Vault

```
srknbnlg1@t-800 ~ ➔ kubectl get secret --namespace default anchore-demo-anchore-engine -o jsonpath=".data.ANCHORE_ADMIN_PASSWORD" | base64 --decode; echo  
nyX2cKP4g1NemzY35AcUrgv8c81e5rZ  
srknbnlg1@t-800 ➔
```

### cli command for get anchore-cli password

```
kubectl get secret --namespace default anchore-demo-anchore-engine -o  
jsonpath=".data.ANCHORE_ADMIN_PASSWORD" | base64 --decode; echo
```



The screenshot shows the 'webGoatKonfKV | Secrets' blade in the Azure portal. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Keys, and Secrets. The 'Secrets' link is currently selected. The main area displays a message: 'The secret 'anchorPassword' has been successfully created.' Below this, a table lists six secrets:

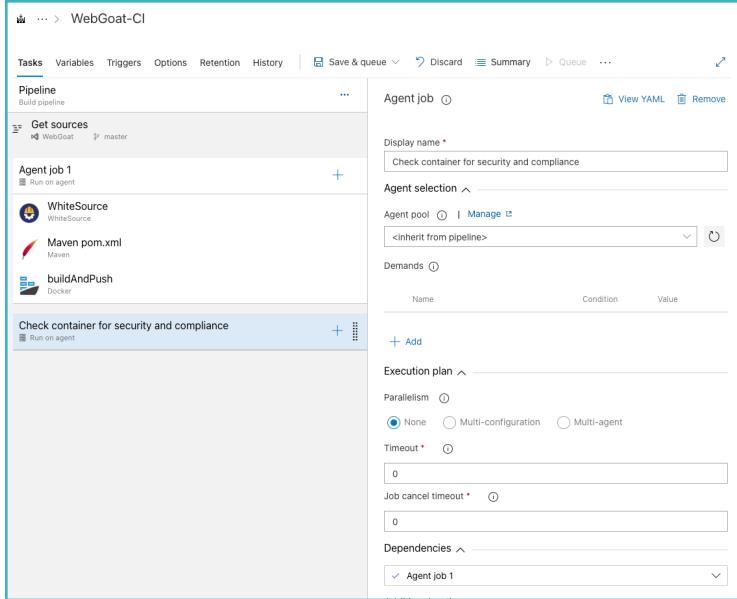
Name	Type	Status	Expiration Date
anchorPassword		✓ Enabled	
anchorUser		✓ Enabled	
anchorServer		✓ Enabled	
acrPassword		✓ Enabled	
acrUsername		✓ Enabled	



For sensitive variables add **anchorPassword** , **anchorUser** , **anchorServer** , azure key vault wallet to use in vulnerability scan task. Don't forget to link variable group to pipeline. And also don't forget to add non-sensitive variables to pipeline variables as **imageRepository** and **acrRegistry**.

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Add new job agent to build pipeline

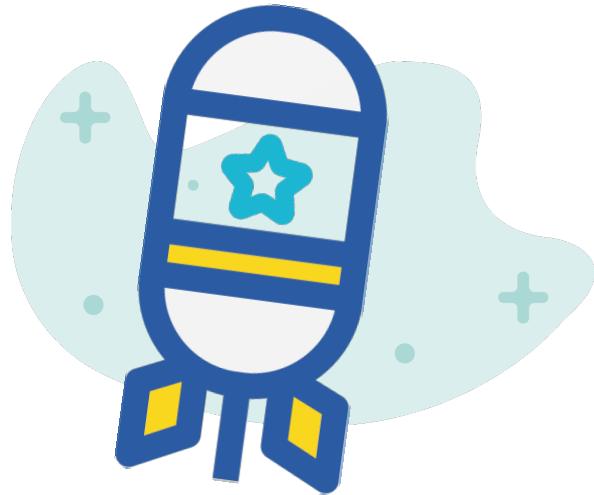


The screenshot shows the Azure DevOps Pipeline editor for a pipeline named 'WebGoat-Cl'. The pipeline consists of three steps:

- Get sources**: Sources from 'WebGoat' repository, master branch.
- Agent job 1**: Run on agent. This step is currently selected.
- Check container for security and compliance**: Run on agent.

The 'Agent job 1' step configuration includes:

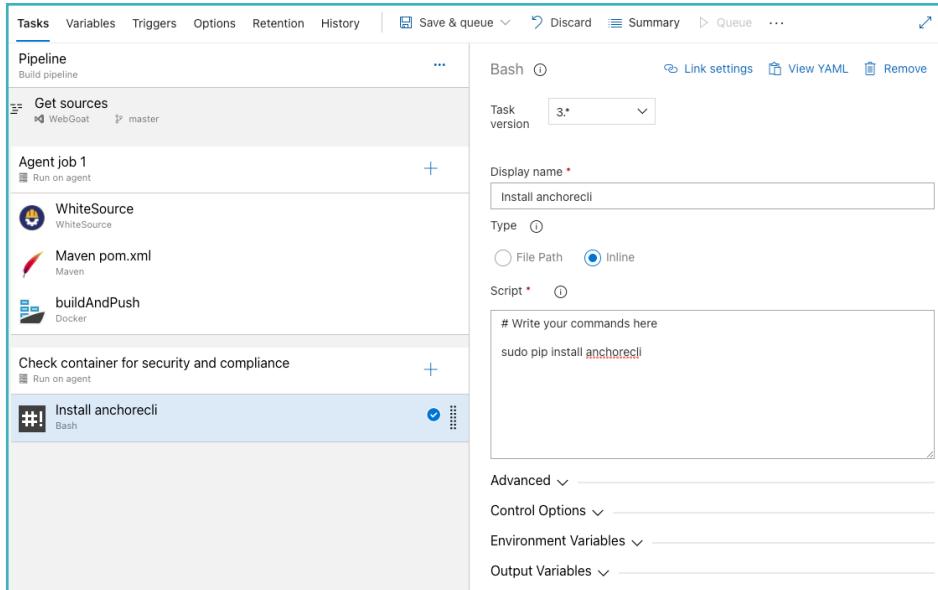
- Display name**: Check container for security and compliance.
- Agent selection**: Agent pool: <inherit from pipeline>.
- Demand**: None.
- Execution plan**: Parallelism: None.
- Timeout**: 0.
- Job cancel timeout**: 0.
- Dependencies**: Agent job 1.



In the Dependencies select agent job, which is responsible for compiling and pushing WebGoat container to ACR. It is important to have that job finished before scans gets triggered as we want to scan the latest image.

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : First Task - Install anchore cli



The screenshot shows the Azure DevOps Pipeline Editor. A pipeline named "Pipeline" is displayed. The tasks are:

- Get sources (WebGoat, master)
- Agent job 1 (Run on agent):
  - WhiteSource (WhiteSource)
  - Maven pom.xml (Maven)
  - buildAndPush (Docker)
- Check container for security and compliance (Run on agent)
- Install anchorecli (Bash):
  - Task version: 3.\*
  - Display name: Install anchorecli
  - Type: Inline
  - Script:

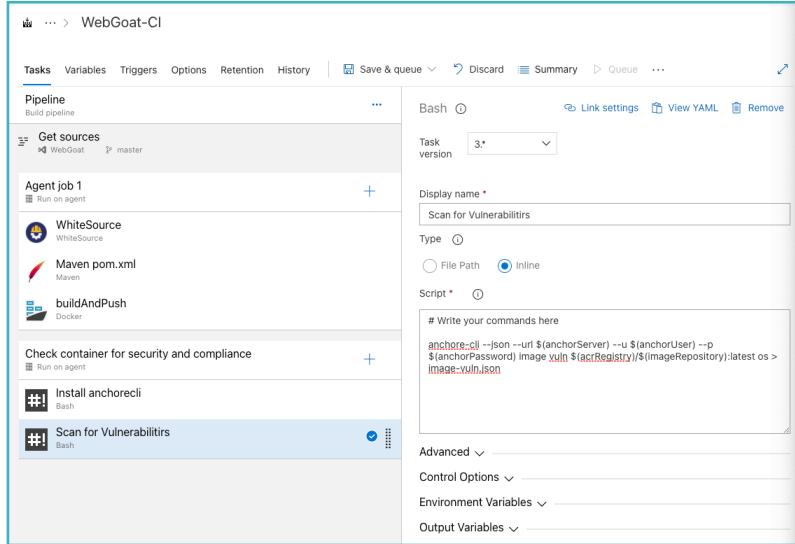
```
# Write your commands here
sudo pip install anchorecli
```



First **Bash** task is responsible for installing anchore-cli tool on agent.

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Second Task - Vulnerability scanner



The screenshot shows the Azure DevOps Pipeline interface for a build pipeline named "WebGoat-CI". The pipeline consists of several stages:

- Get sources**: Fetches code from a "WebGoat" repository.
- Agent job 1**: Contains steps for "WhiteSource" analysis, "Maven pom.xml" build, and "buildAndPush" Docker image creation.
- Check container for security and compliance**: Installs "anchorecli" and runs a "Scan for Vulnerabilitirs" task.
- Scan for Vulnerabilitirs**: The highlighted task is a "Bash" task with the following configuration:
  - Display name**: Scan for Vulnerabilitirs
  - Type**: Set to "Inline".
  - Script**: Contains the command:
 

```
# Write your commands here
anchore-cli --json --url ${anchorServer} --u ${anchorUser} --p ${anchorPassword} image vuln ${acrRegistry}/${imageRepository}:latest os > image-vuln.json
```



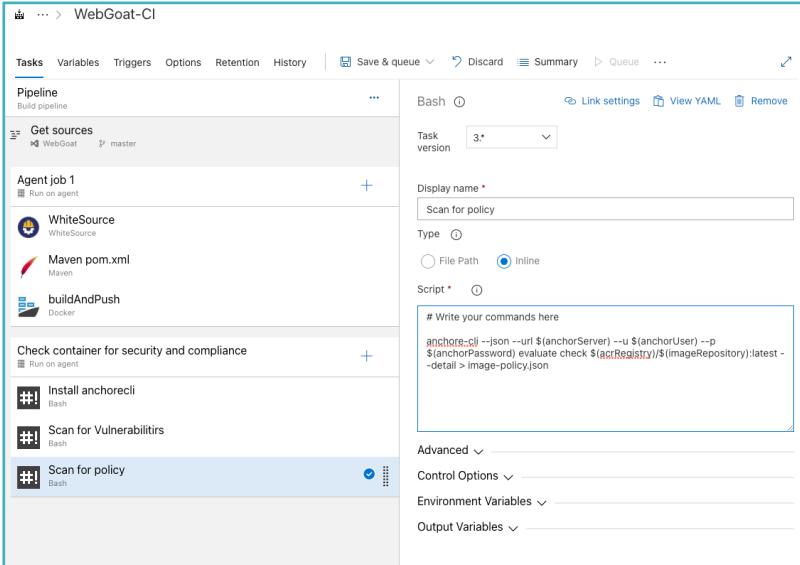
### Bash task command

```
anchore-cli --json --url ${anchorServer} --u ${anchorUser} --p ${anchorPassword} image vuln ${acrRegistry}/${imageRepository}:Latest os > image-vuln.json
```

Second **Bash** task is responsible for scan for vulnerabilities in the image.

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Third Task - Policy scanner



The screenshot shows the Azure DevOps Pipeline interface for a pipeline named "WebGoat-Cl". The pipeline consists of several tasks:

- Get sources**: Checks out "WebGoat" from "master".
- Agent job 1** (Run on agent):
  - WhiteSource**: WhiteSource task.
  - Maven pom.xml**: Maven task.
  - buildAndPush**: Docker task.
- Check container for security and compliance**: Run on agent task.
- Install anchorecli**: Bash task.
- Scan for Vulnerabilities**: Bash task.
- Scan for policy**: This task is currently selected. It is a Bash task with the following configuration:
  - Type**: Set to **Inline**.
  - Script**: Contains the command:
 

```
anchore-cli --json --url $(anchorServer) --u $(anchorUser) --p $(anchorPassword) evaluate check ${acrRegistry}/${imageRepository}:latest --detail > image-policy.json
```



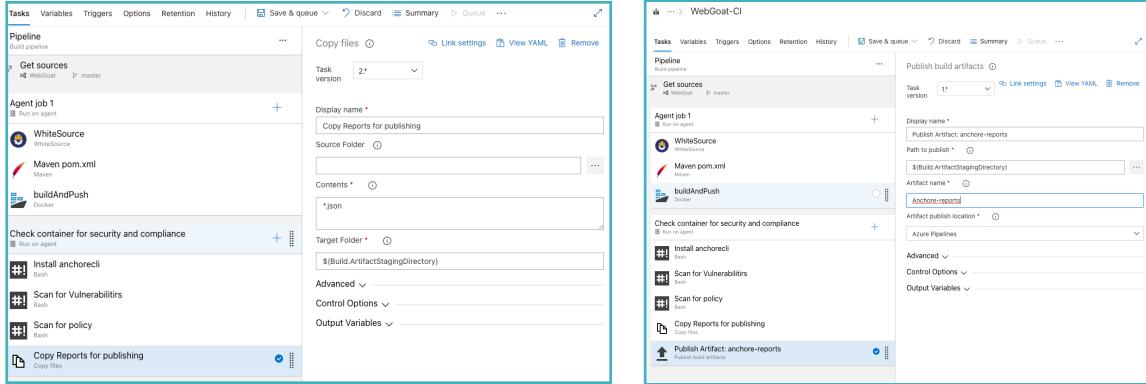
### Bash task command

```
anchore-cli --json --url $(anchorServer) --u $(anchorUser) --p $(anchorPassword) evaluate check ${acrRegistry}/${imageRepository}:Latest --detail > image-policy.json
```

Third **Bash** task is responsible for scan for policies in the image.

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Generate Anchore Report



The image shows two side-by-side Azure DevOps pipeline configurations:

- Pipeline 1 (Left):**
  - Tasks:** Get sources (Windows master), Agent job 1 (Run on agent) with WhiteSource, Maven pom.xml, buildAndPush (Docker).
  - Check container for security and compliance:**
    - Install anchorecli (Bash)
    - Scan for Vulnerabilities (Bash)
    - Scan for policy (Bash)
    - Copy Reports for publishing (Copy files)
- Pipeline 2 (Right):**
  - Tasks:** Get sources (Windows master), Agent job 1 (Run on agent) with WhiteSource, Maven pom.xml, buildAndPush (Docker).
  - Check container for security and compliance:**
    - Install anchorecli (Bash)
    - Scan for Vulnerabilities (Bash)
    - Scan for policy (Bash)
    - Copy Reports for publishing (Copy files)
  - Copy files Task:** Task version 2+, Display name "Copy Reports for publishing", Source Folder, Contents ".json", Target Folder "\$(Build.ArtifactStagingDirectory)".
  - Publish build artifacts Task:** Task version 1+, Display name "Publish Artifact: anchore-reports", Path to publish "\$(Build.ArtifactStagingDirectory)", Artifact name "Anchore-reports", Artifact publish location "Azure Pipelines".

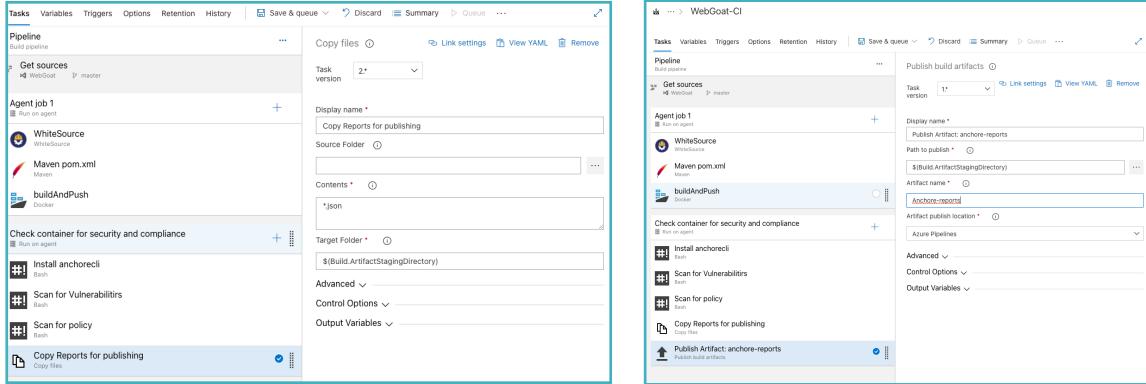


Both scan will produce json reports and we need to publish them as pipeline artifact, which can be downloaded after. To do this add two tasks:

- **Copy Files** : *contents => \*.json , Target Folder => \$(Build.ArtifactStagingDirectory)*
- **Publish build artifacts** : *Path to publish => \$(Build.ArtifactStagingDirectory) , Artifact name => Anchore-reports*

# DEMO: Implement Security in Azure DevOps CI/CD

## Anchore : Generate Anchore Report



The image shows two side-by-side Azure DevOps pipeline configurations:

- Pipeline 1 (Left):**
  - Tasks:** Get sources (Windows master), Agent job 1 (Run on agent) with WhiteSource, Maven pom.xml, buildAndPush (Docker).
  - Check container for security and compliance:**
    - Install anchorecli (Bash)
    - Scan for Vulnerabilities (Bash)
    - Scan for policy (Bash)
    - Copy Reports for publishing (Copy files)
- Pipeline 2 (Right):**
  - Tasks:** Get sources (Windows master), Agent job 1 (Run on agent) with WhiteSource, Maven pom.xml, buildAndPush (Docker).
  - Check container for security and compliance:**
    - Install anchorecli (Bash)
    - Scan for Vulnerabilities (Bash)
    - Scan for policy (Bash)
    - Copy Reports for publishing (Copy files)
  - Copy files Task:** Task version 2+, Display name "Copy Reports for publishing", Source Folder, Contents ".json", Target Folder "\$(Build.ArtifactStagingDirectory)".
  - Publish build artifacts Task:** Task version 1+, Display name "Publish Artifact: anchore-reports", Path to publish "\$(Build.ArtifactStagingDirectory)", Artifact name "Anchore-reports", Artifact publish location "Azure Pipelines".



Both scan will produce json reports and we need to publish them as pipeline artifact, which can be downloaded after. To do this add two tasks:

- **Copy Files** : *contents => \*.json , Target Folder => \$(Build.ArtifactStagingDirectory)*
- **Publish build artifacts** : *Path to publish => \$(Build.ArtifactStagingDirectory) , Artifact name => Anchore-reports*

# Come Join Us, The Kloians:

We are a team of experienced and happy high achievers,

Knowing that 'Great vision without great people is irrelevant.'

So:

- Talents determine the positions, we do not miss the best,
- We hire the best and providing them the best, pecuniary and non-pecuniary,
- We set high standards, so you better have those as well,
- Stunning colleagues means a great workplace, and the motive to be online.

If you have the Stunning 'Kloian' essence or potential by being:

- No Bullshit
- Self-motivated and Reliable
- Optimistic and Focused
- Happy and Fun
- Flexible and Trustworthy
- Willing and Creative
- Open minded and Kind (Majority of above, if not all.) 😊

You are more than welcome to apply: [career@kloia.com](mailto:career@kloia.com)



# Q & A

## Thank you for listening.



### RESOURCES MENTIONED IN THIS SESSION

<https://github.com/kloia/dotnetkonf21-Azure-DevSecOp>

