How to Contain Security in Your Containers

SecureContainers

Adrian Gonzalez
Principal Software Engineering Lead, Microsoft

Adrian Gonzalez

Principal Software Engineering Lead

Microsoft Commercial Software Engineering

I ensure Security is part of my teams engineering fundamentals

new cuisines

wine tastings

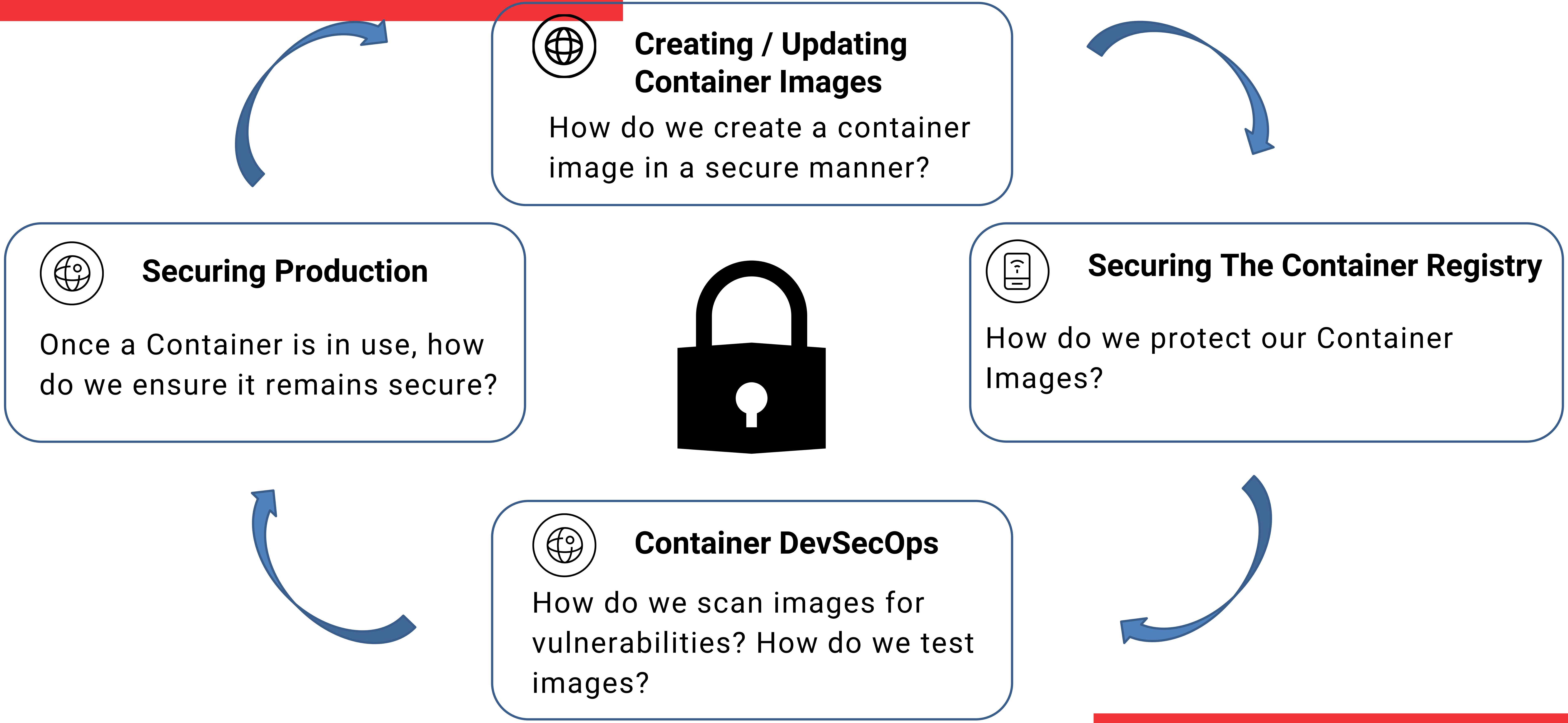
traveling

most outdoor activities

watching baseball

linkedin.com/in/**adrian-g-gonzalez/**

Where Does Security Live in Containers



Creating / Updating Container Images

How do we create a container image in a secure manner?

Securing The Container Registry

How do we protect our Container Images?

Container DevSecOps

How do we scan images for vulnerabilities? How do we test images?

Securing Production

Once a Container is in use, how do we ensure it remains secure?

Where Does Security Live in Containers

Creating / Updating Container Images

How do we create a container image in a secure manner?

Updated OS and containerization software

Using a non-root user

Using Trusted Registries (Public or Private)

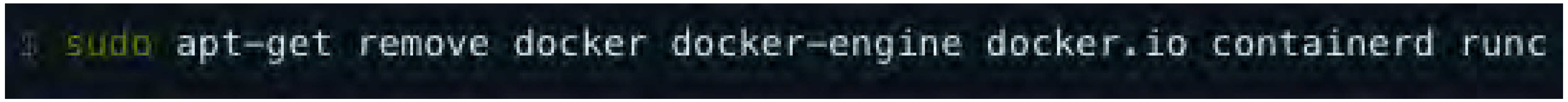
Have a lean image

Update Versions

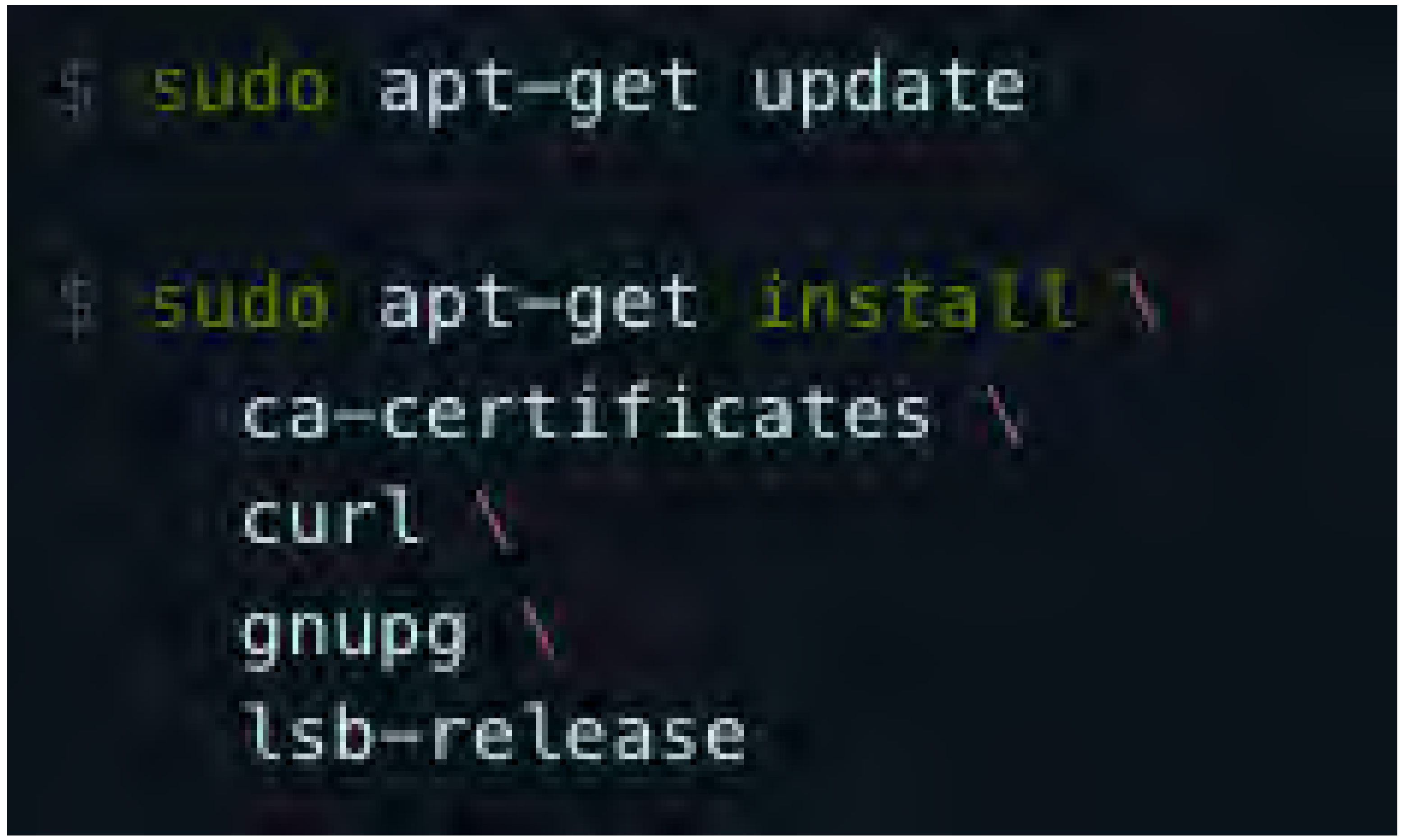
Defining a Container Image

docker docs

Uninstall old versions



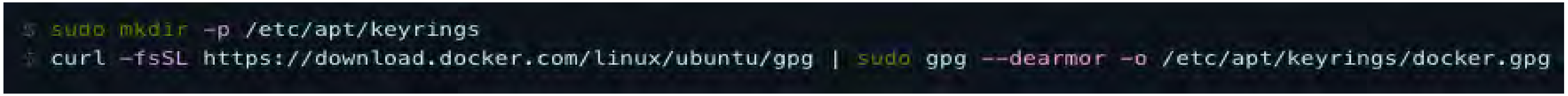
```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```



```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Update apt packages

Add Docker's official GPG key



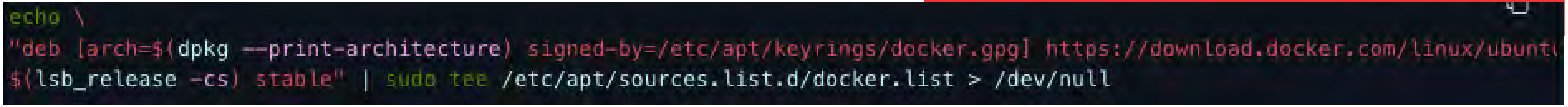
```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Update Versions

Defining a Container Image

docker docs

Set up the repository

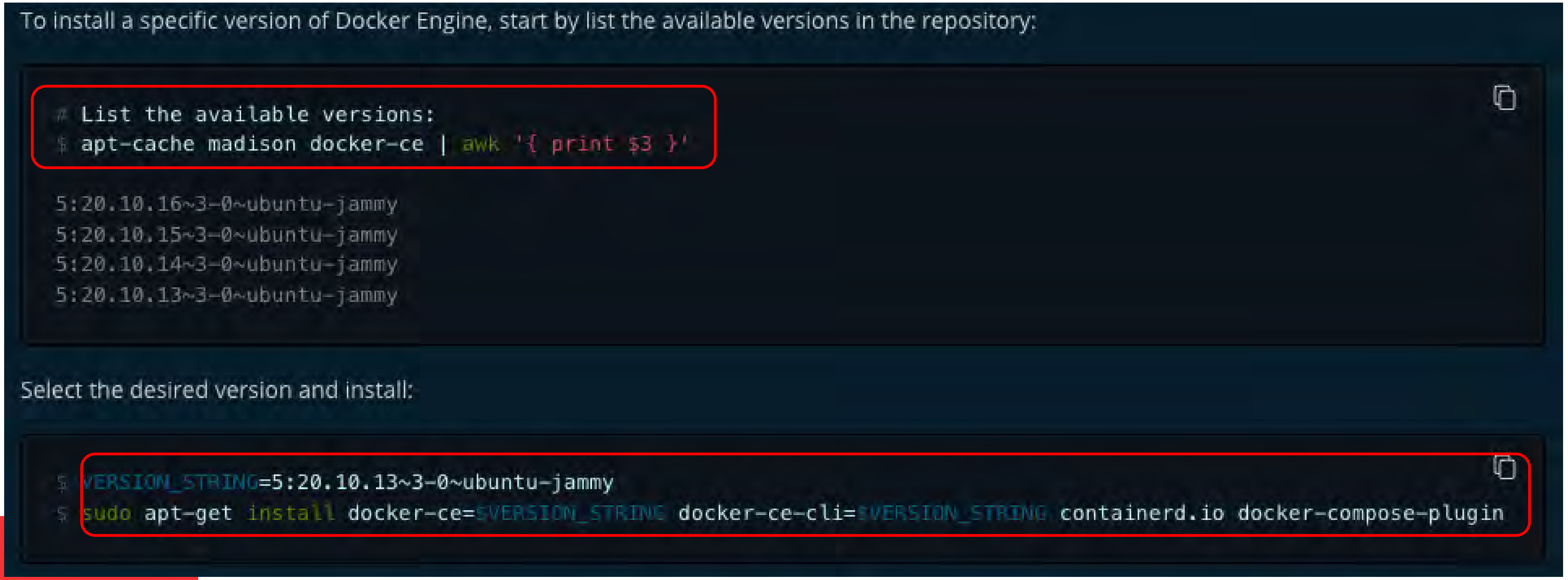


```
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubunt
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine,

container, and

Docker compose



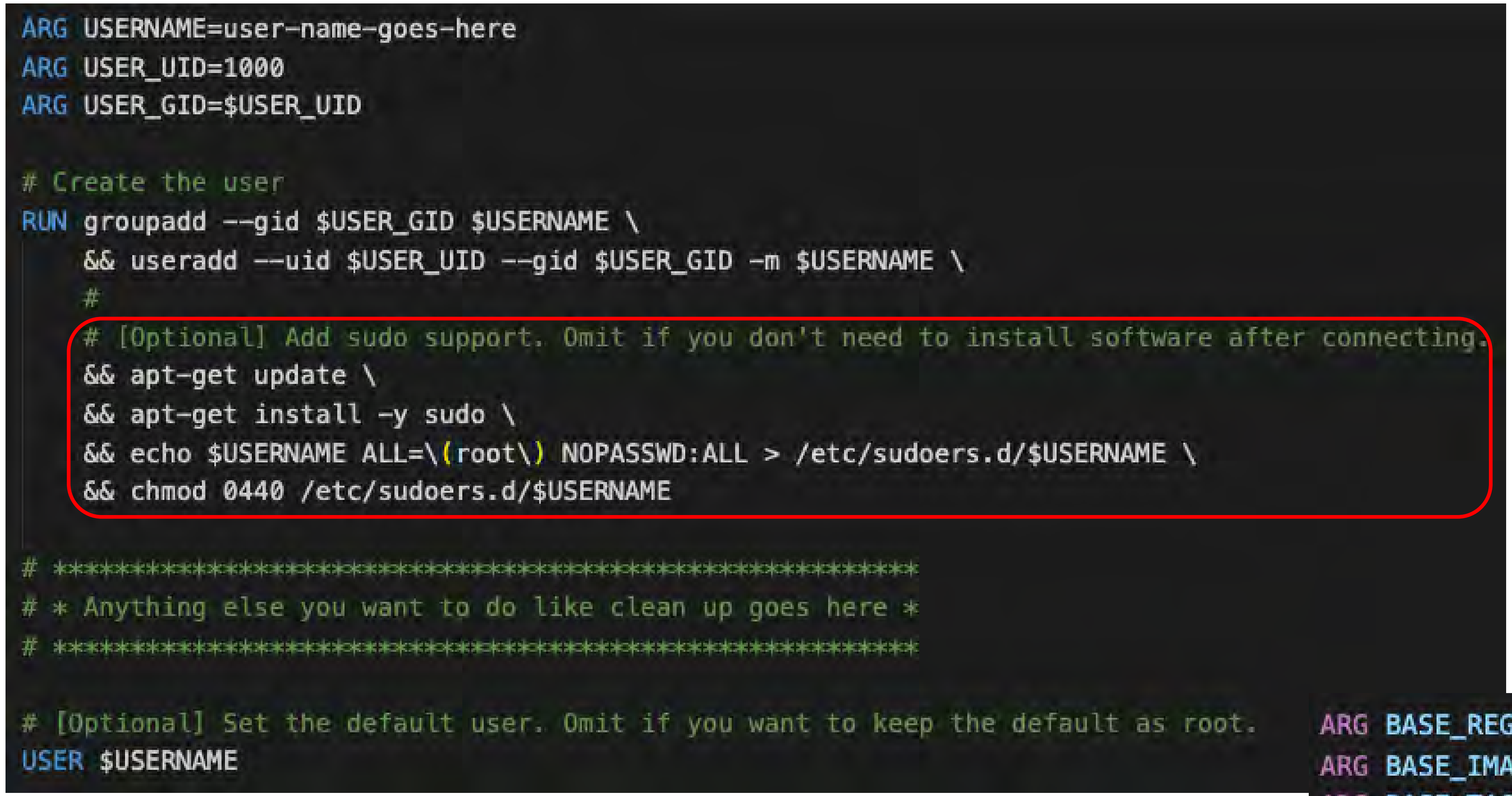
To install a specific version of Docker Engine, start by list the available versions in the repository:

```
# List the available versions:
$ apt-cache madison docker-ce | awk '{ print $3 }'

5:20.10.16~3-0~ubuntu-jammy
5:20.10.15~3-0~ubuntu-jammy
5:20.10.14~3-0~ubuntu-jammy
5:20.10.13~3-0~ubuntu-jammy
```

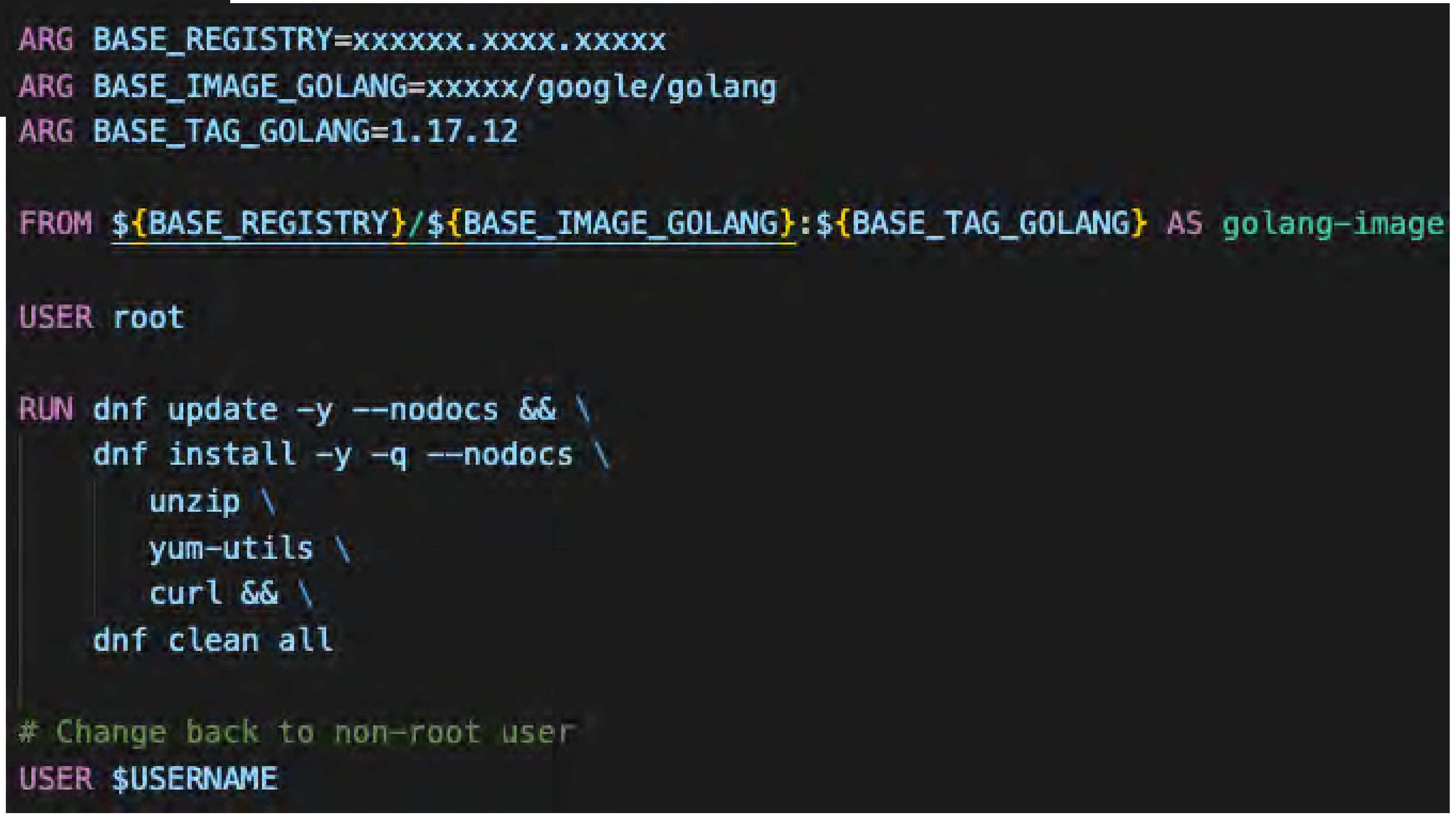
Select the desired version and install:

```
$ VERSION_STRING=5:20.10.13~3-0~ubuntu-jammy
$ sudo apt-get install docker-ce=$VERSION_STRING docker-ce-cli=$VERSION_STRING containerd.io docker-compose-plugin
```

Using a Non-Root User

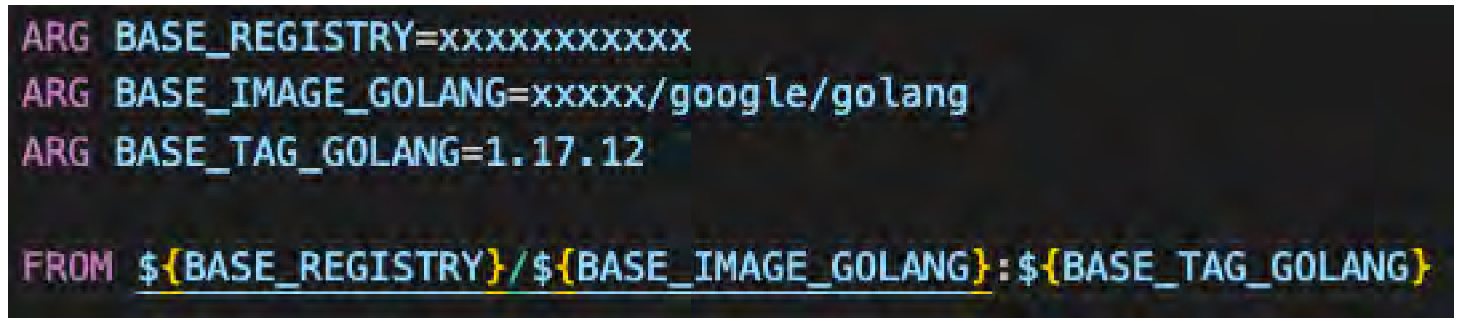


```
ARG USERNAME=user-name-goes-here
ARG USER_UID=1000
ARG USER_GID=$USER_UID

# Create the user
RUN groupadd --gid $USER_GID $USERNAME \
    && useradd --uid $USER_UID --gid $USER_GID -m $USERNAME \
    #
    # [Optional] Add sudo support. Omit if you don't need to install software after connecting.
    && apt-get update \
    && apt-get install -y sudo \
    && echo $USERNAME ALL=\(root\) NOPASSWD:ALL > /etc/sudoers.d/$USERNAME \
    && chmod 0440 /etc/sudoers.d/$USERNAME

# ********************************************************
# * Anything else you want to do like clean up goes here *
# ********************************************************

# [Optional] Set the default user. Omit if you want to keep the default as root.
USER $USERNAME
```

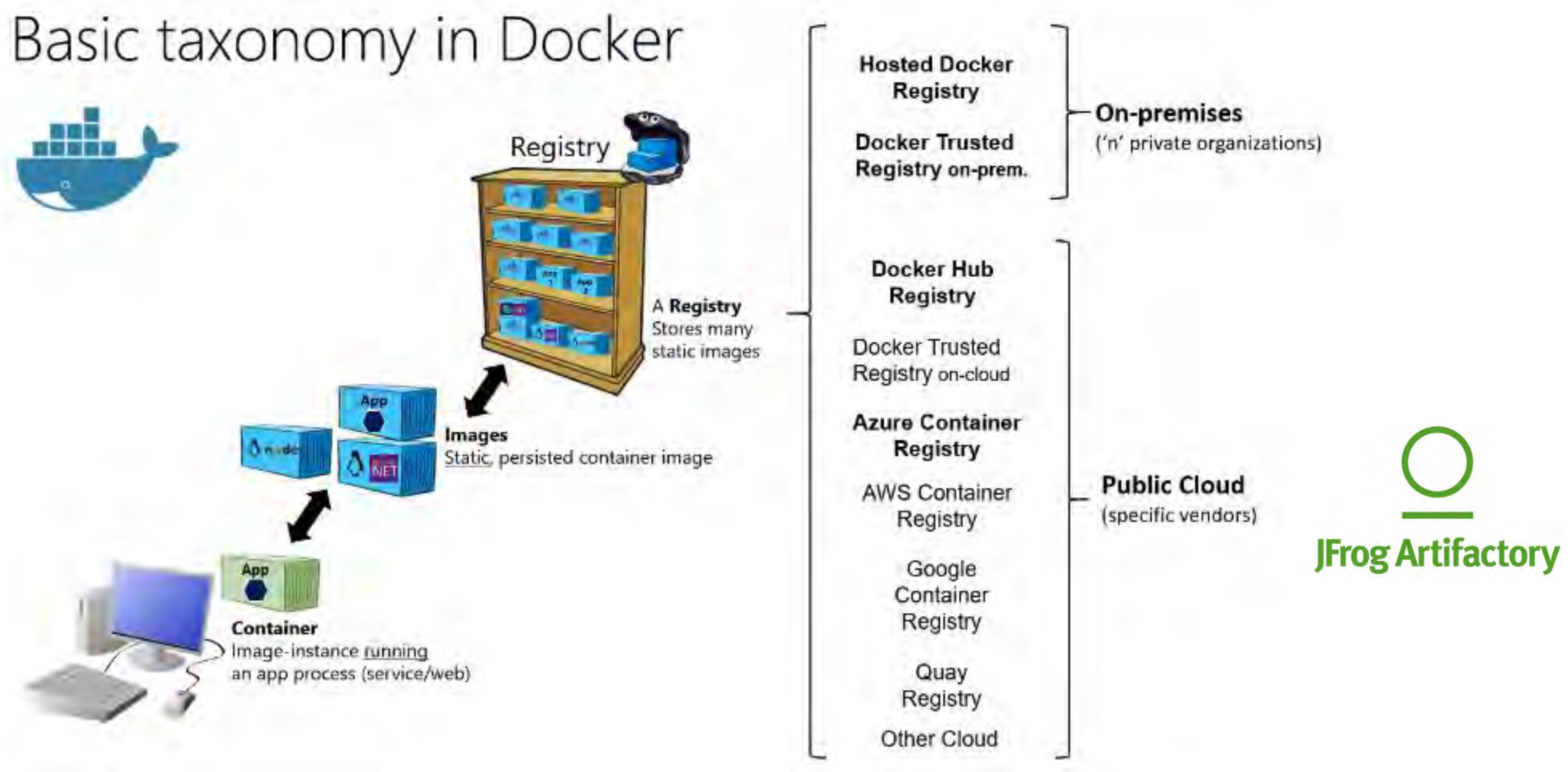


```
ARG BASE_REGISTRY=xxxxxx.xxxx.xxxxx
ARG BASE_IMAGE_GOLANG=xxxxx/google/golang
ARG BASE_TAG_GOLANG=1.17.12

FROM ${BASE_REGISTRY}/${BASE_IMAGE_GOLANG}:${BASE_TAG_GOLANG} AS golang-image

USER root

RUN dnf update -y --nodocs && \
    dnf install -y -q --nodocs \
        unzip \
        yum-utils \
        curl && \
    dnf clean all

# Change back to non-root user
USER $USERNAME
```

Using Trusted Sources/Registries

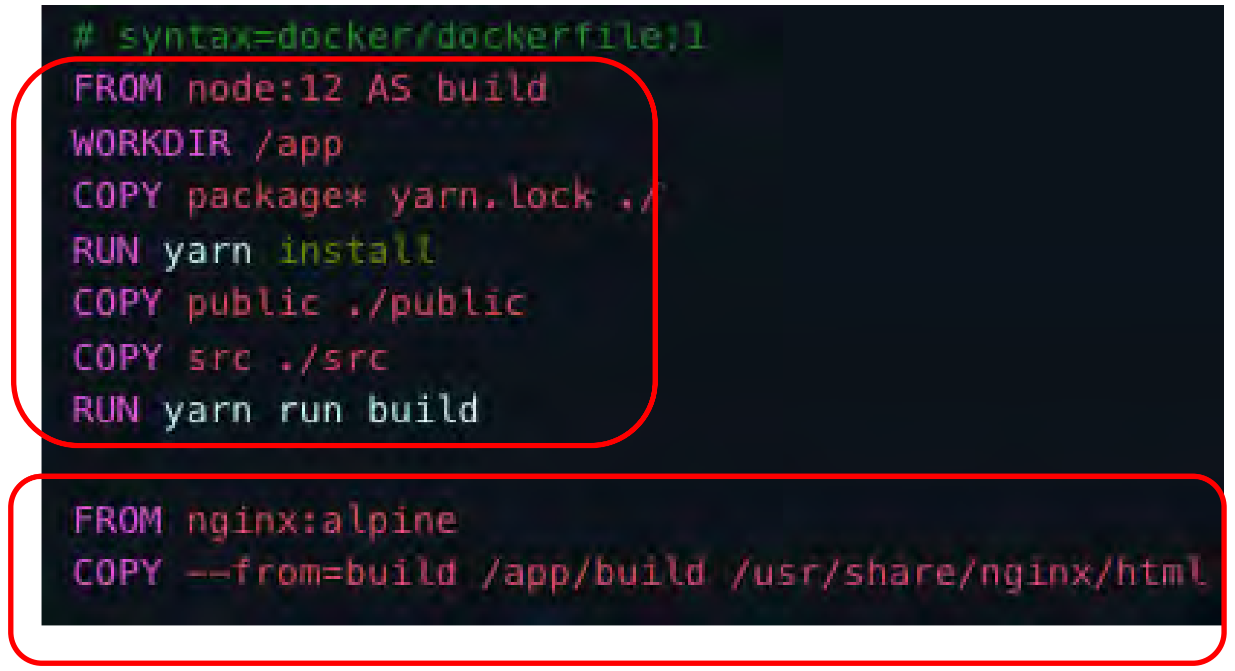


```
ARG BASE_REGISTRY=xxxxxxxxxxx
ARG BASE_IMAGE_GOLANG=xxxxx/google/golang
ARG BASE_TAG_GOLANG=1.17.12

FROM ${BASE_REGISTRY}/${BASE_IMAGE_GOLANG}:${BASE_TAG_GOLANG}
```



Lean Image with Required Packages

Multi-Stage Builds (separate build vs runtime dependencies)



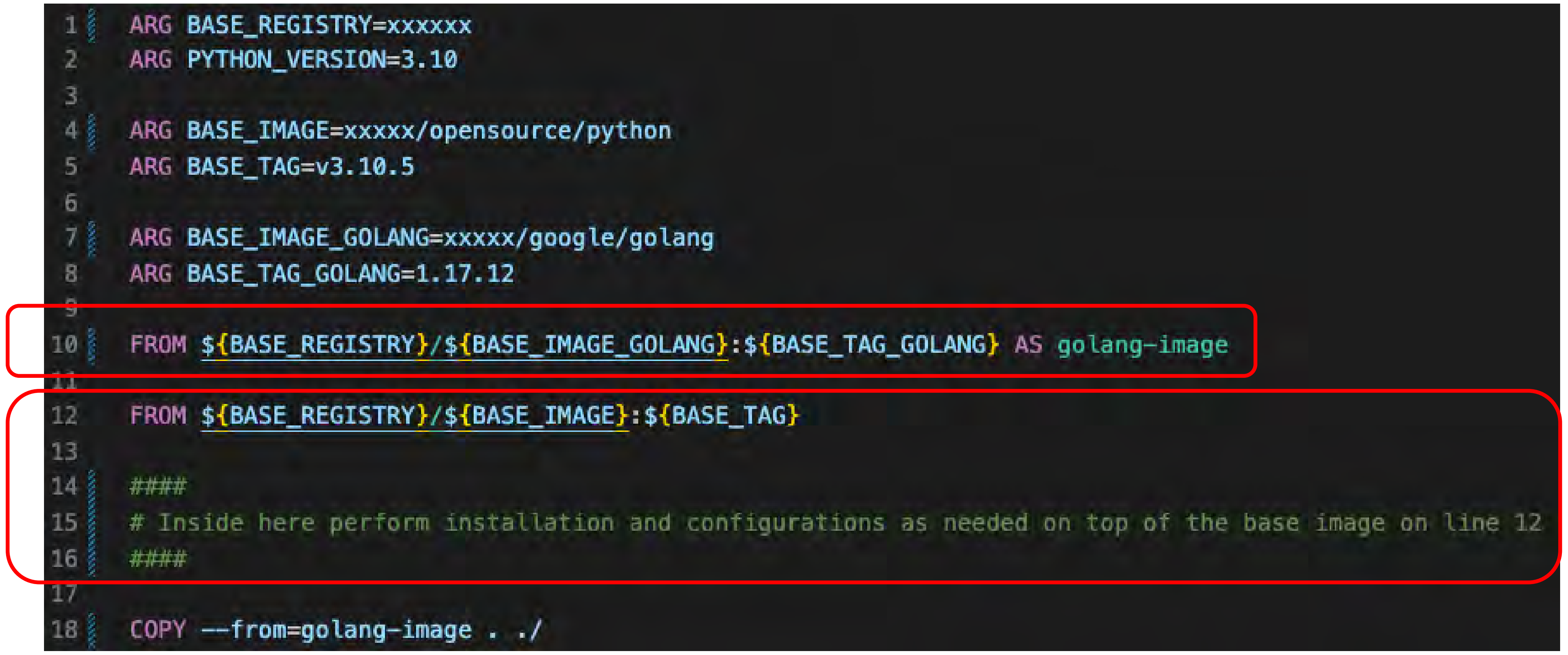
“build”

```
# syntax=docker/dockerfile:1
FROM node:12 AS build
WORKDIR /app
COPY package* yarn.lock ./
RUN yarn install
COPY public ./public
COPY src ./src
RUN yarn run build
```

“webapp”

```
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
```

Define Base Images for Shared Dependencies



```
1   ARG BASE_REGISTRY=xxxxxx
2   ARG PYTHON_VERSION=3.10
3
4   ARG BASE_IMAGE=xxxxx/opensource/python
5   ARG BASE_TAG=v3.10.5
6
7   ARG BASE_IMAGE_GOLANG=xxxxx/google/golang
8   ARG BASE_TAG_GOLANG=1.17.12
9
10  FROM ${BASE_REGISTRY}/${BASE_IMAGE_GOLANG}:${BASE_TAG_GOLANG} AS golang-image
11
12  FROM ${BASE_REGISTRY}/${BASE_IMAGE}:${BASE_TAG}
13
14  ####
15  # Inside here perform installation and configurations as needed on top of the base image on line 12
16  ####
17
18  COPY --from=golang-image . ./
```

Where Does Security Live in Containers

Creating / Updating Container Images

How do we create a container image in a secure manner?

Securing The Container Registry

How do we protect our Container Registry and manage authorization?

Private Registry - Connectivity

IDAM (Identity Access Management) and **RBAC** (Role Based Access Controls)

Digitally Signing Images

Private Registry Connectivity

Hosted Docker Registry

Docker Trusted Registry on-prem.

On-premises
('n' private organizations)

Docker Hub Registry

Docker Trusted Registry on-cloud

Azure Container Registry

AWS Container Registry

Google Container Registry

Quay Registry

Other Cloud

Public Cloud
(specific vendors)

Network Security

- Firewalls
- Source IP Range Policies
- Port Policies

Image Digital Signatures

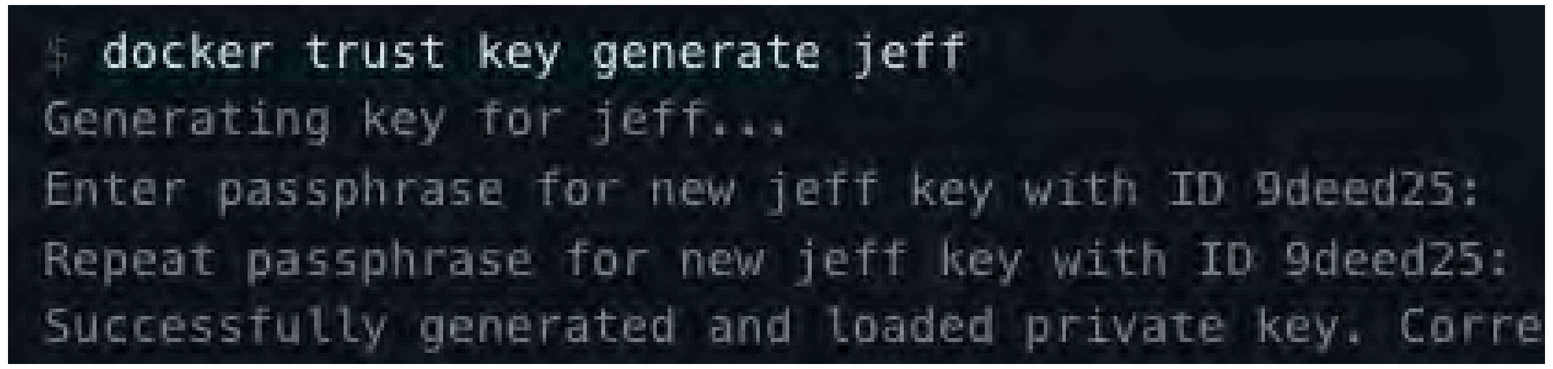
Docker Contest Trust (DCT) docs:

“DCT provides the ability to use digital signatures for data sent to and received from remote Docker registries. These signatures allow client-side or runtime verification of the integrity and publisher of specific image tags.”

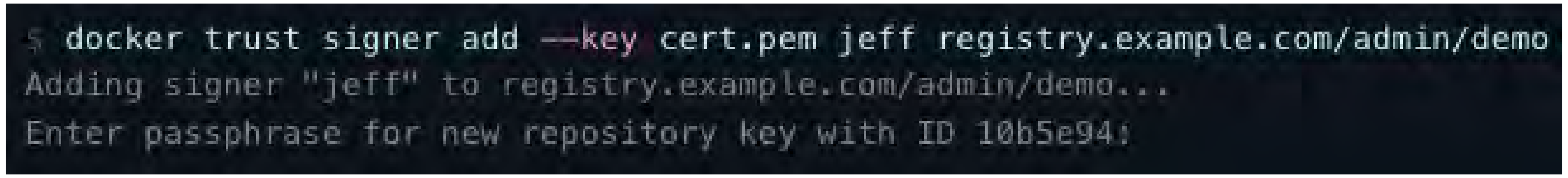
Azure Container Registry Docs:

*“Image signing or fingerprinting can provide a **chain of custody** that enables you to verify the integrity of the containers”*

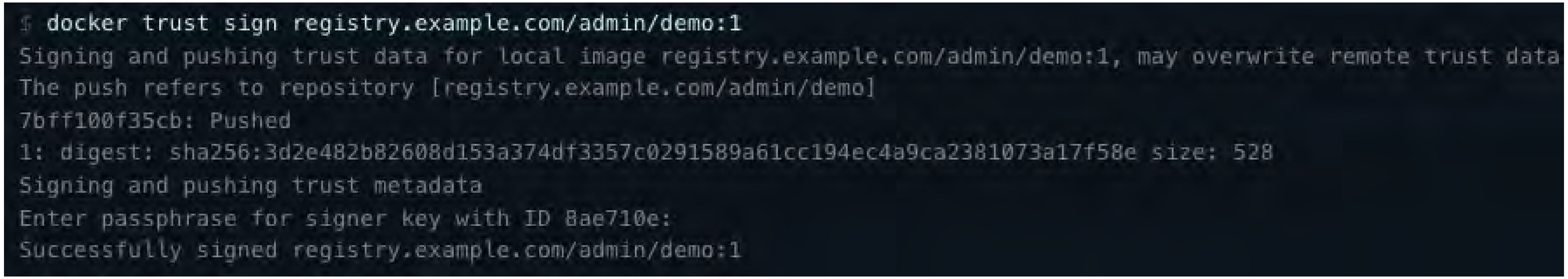
Image Digital Signatures



```
$ docker trust key generate jeff
Generating key for jeff...
Enter passphrase for new jeff key with ID 9deed25:
Repeat passphrase for new jeff key with ID 9deed25:
Successfully generated and loaded private key. Corre
```

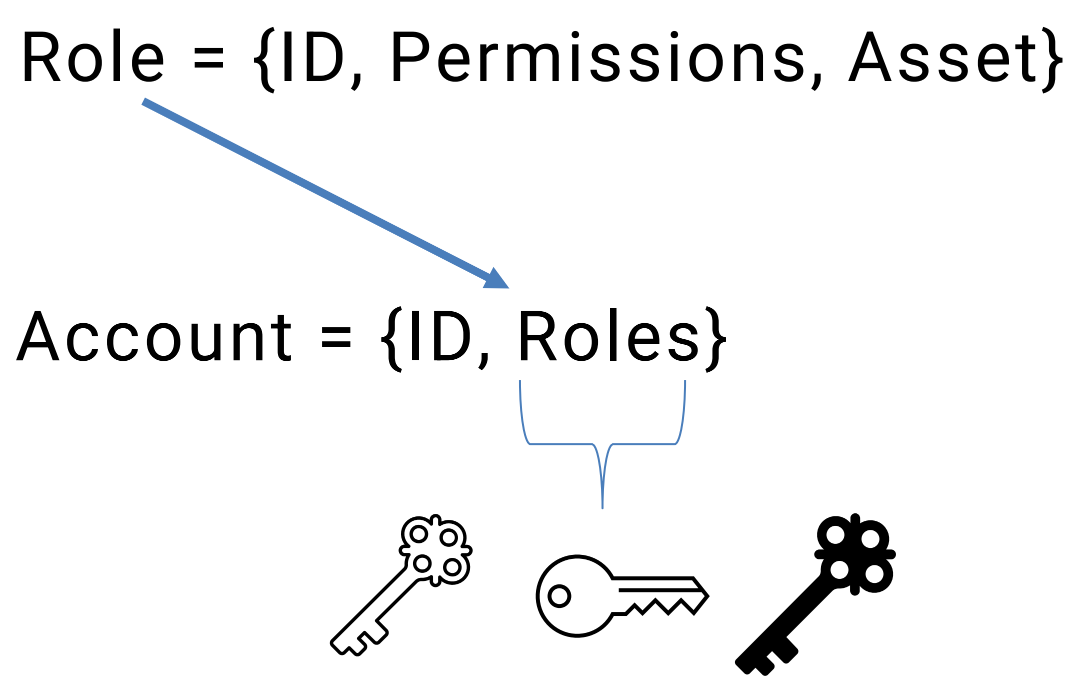


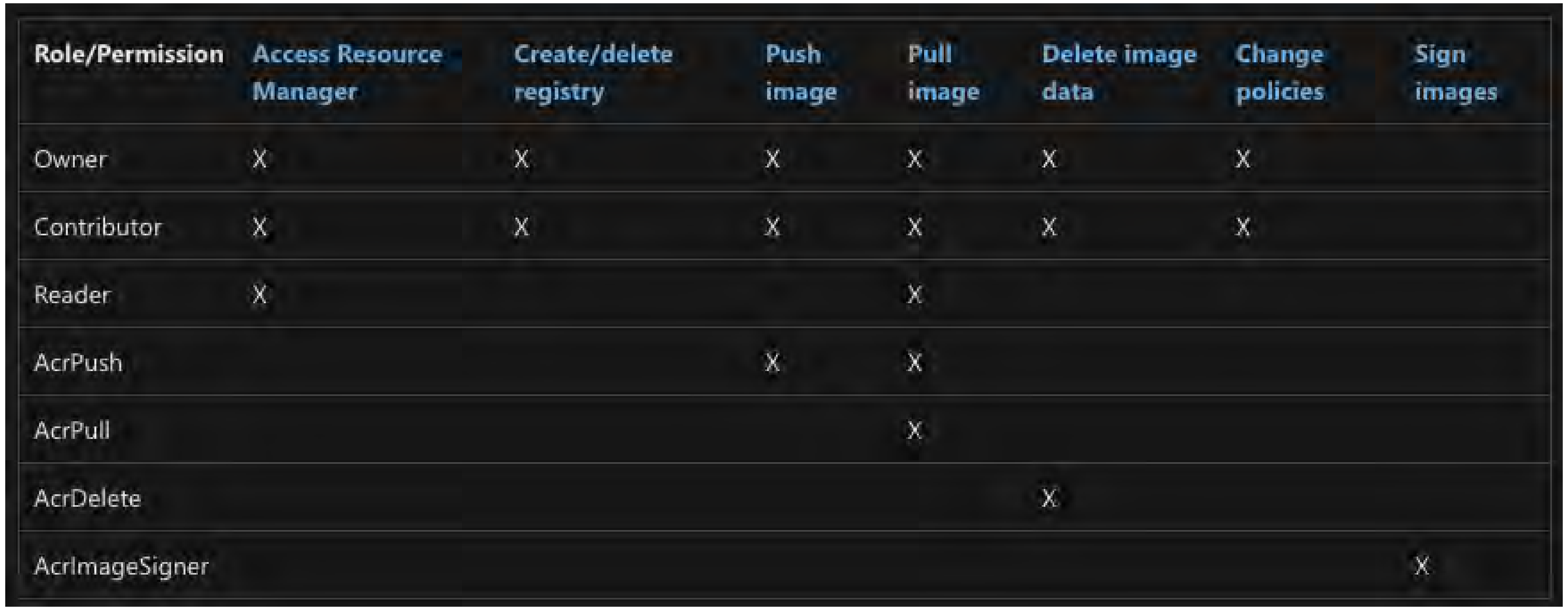
```
$ docker trust signer add --key cert.pem jeff registry.example.com/admin/demo
Adding signer "jeff" to registry.example.com/admin/demo...
Enter passphrase for new repository key with ID 10b5e94:
```



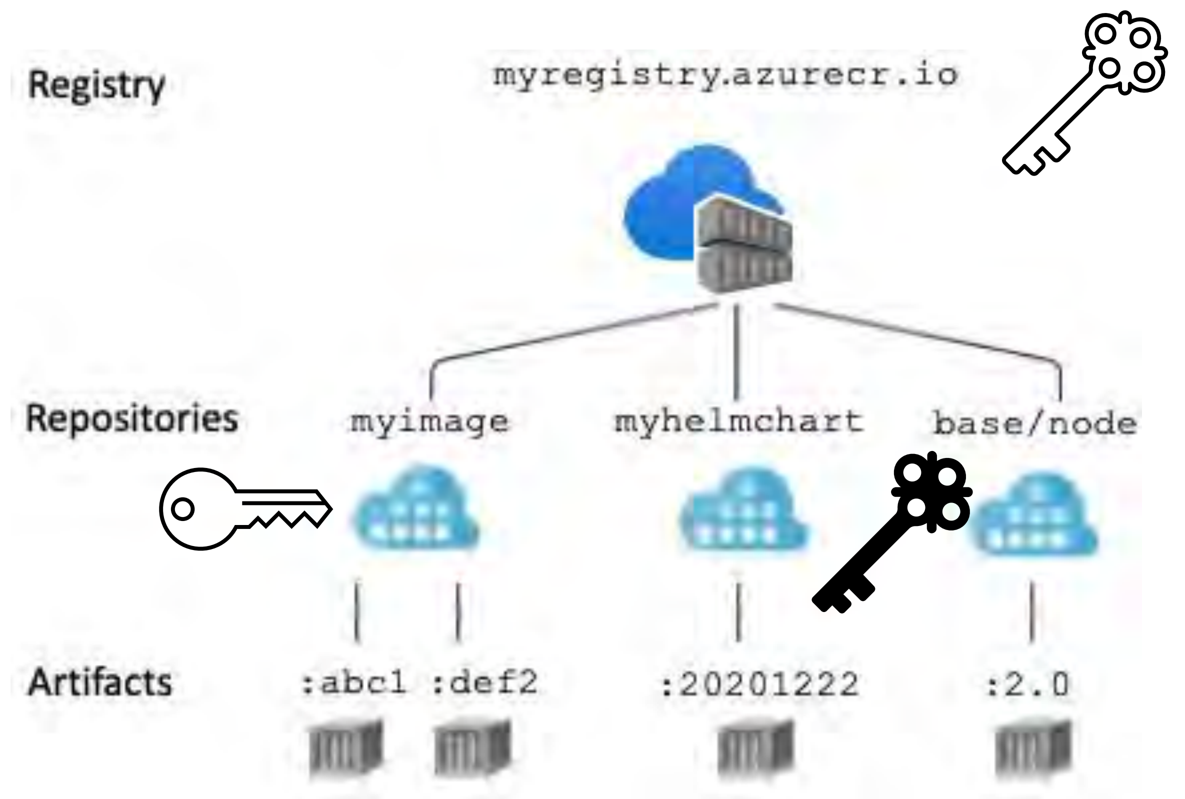
```
$ docker trust sign registry.example.com/admin/demo:1
Signing and pushing trust data for local image registry.example.com/admin/demo:1, may overwrite remote trust data
The push refers to repository [registry.example.com/admin/demo]
7bff100f35cb: Pushed
1: digest: sha256:3d2e482b82608d153a374df3357c0291589a61cc194ec4a9ca2381073a17f58e size: 528
Signing and pushing trust metadata
Enter passphrase for signer key with ID 8ae710e:
Successfully signed registry.example.com/admin/demo:1
```

IDAM and RBAC





Role/Permission	Access Resource Manager	Create/delete registry	Push image	Pull image	Delete image data	Change policies	Sign images
Owner	X	X	X	X	X	X	
Contributor	X	X	X	X	X	X	
Reader	X			X			
AcrPush			X	X			
AcrPull				X			
AcrDelete					X		
AcrImageSigner							X



Azure Container Registry Roles

Granular Asset RBAC

Where Does Security Live in Containers

Continuous Integration (CI) Agent Registry Access

Container Scanning

CI Stages

Securing The Container Registry

How do we protect our Container Images?

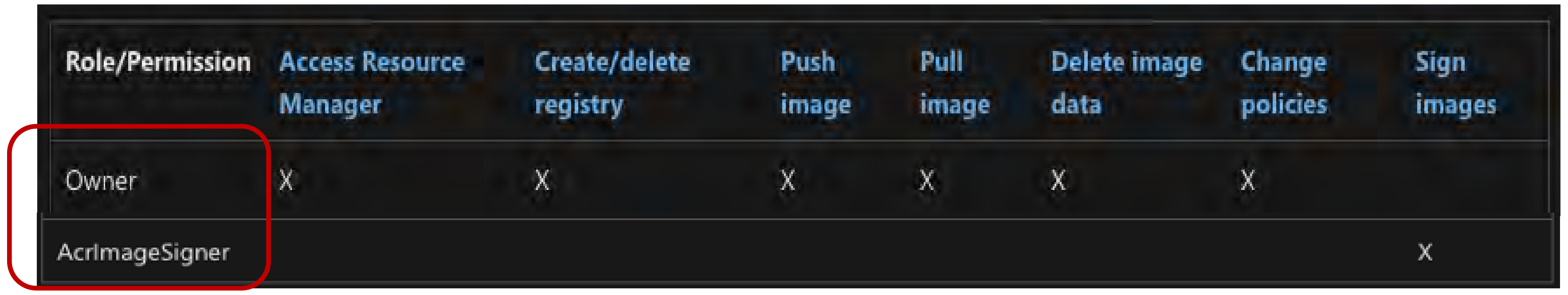
Container DevSecOps

How do we scan images for vulnerabilities? How do we test images?

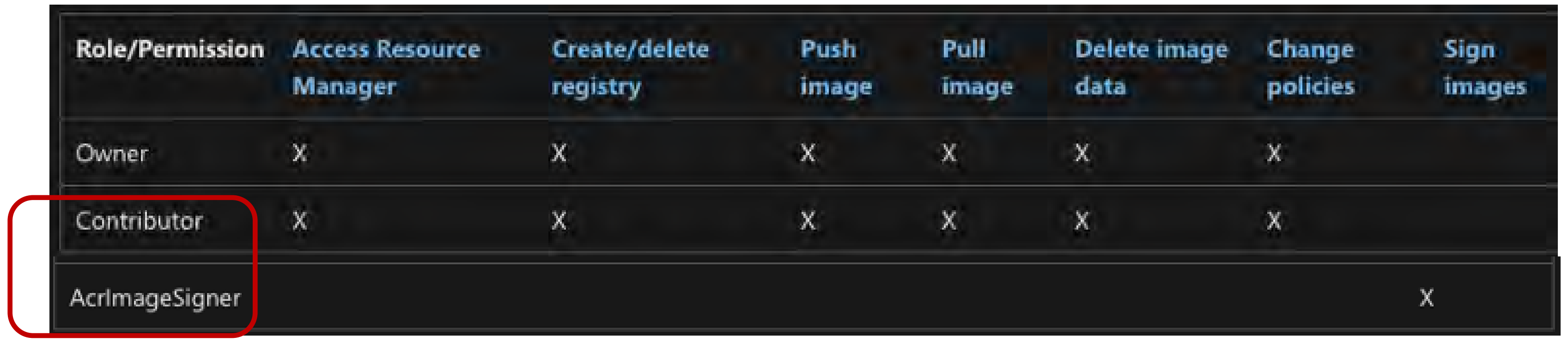
CI and Container Registry

Registry Access

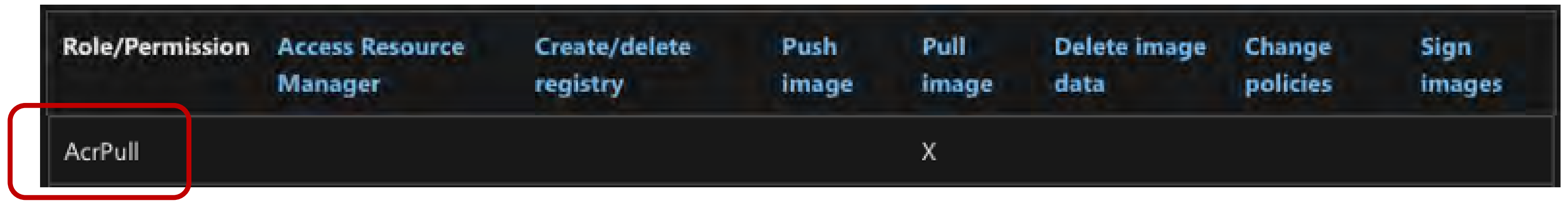
Azure Container Registry RBAC



Role/Permission	Access Resource Manager	Create/delete registry	Push image	Pull image	Delete image data	Change policies	Sign images
Owner	X	X	X	X	X	X	
AcrImageSigner							X

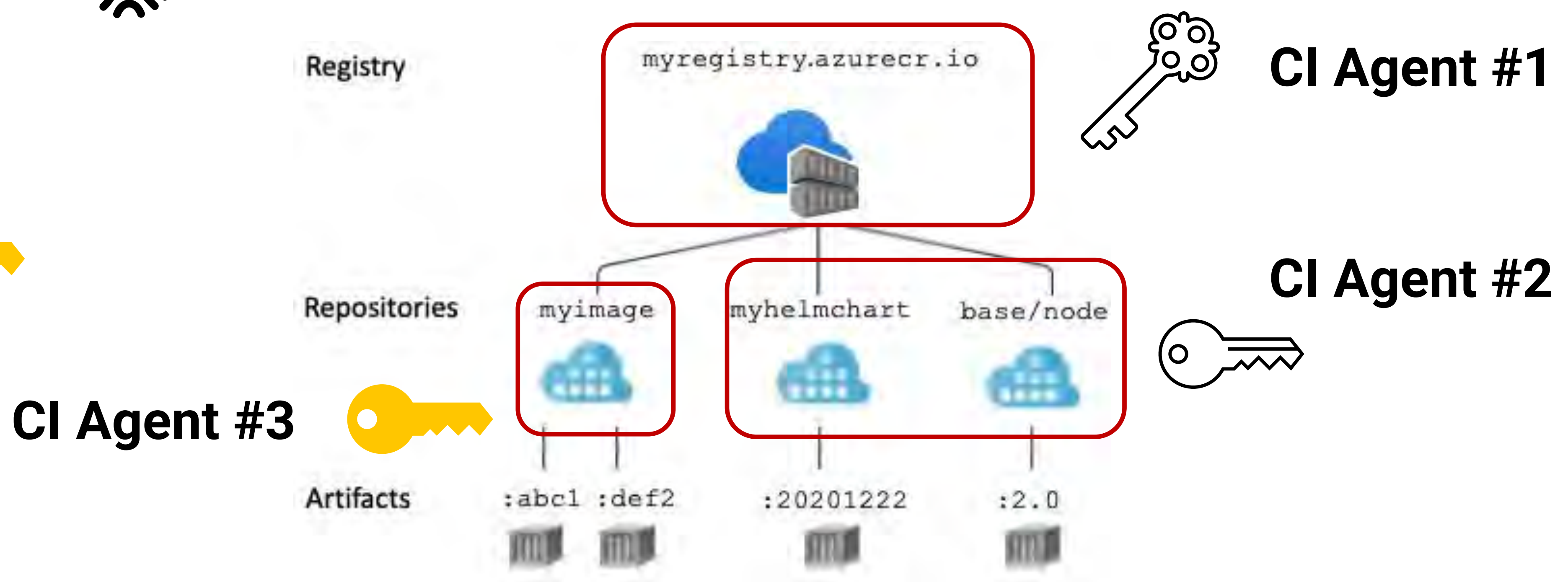


Role/Permission	Access Resource Manager	Create/delete registry	Push image	Pull image	Delete image data	Change policies	Sign images
Owner	X	X	X	X	X	X	
Contributor	X	X	X	X	X	X	
AcrImageSigner							X



Role/Permission	Access Resource Manager	Create/delete registry	Push image	Pull image	Delete image data	Change policies	Sign images
AcrPull				X			

Granular RBAC



Automation

Building Containers with Azure DevOps using DevTest Pattern

Container code change

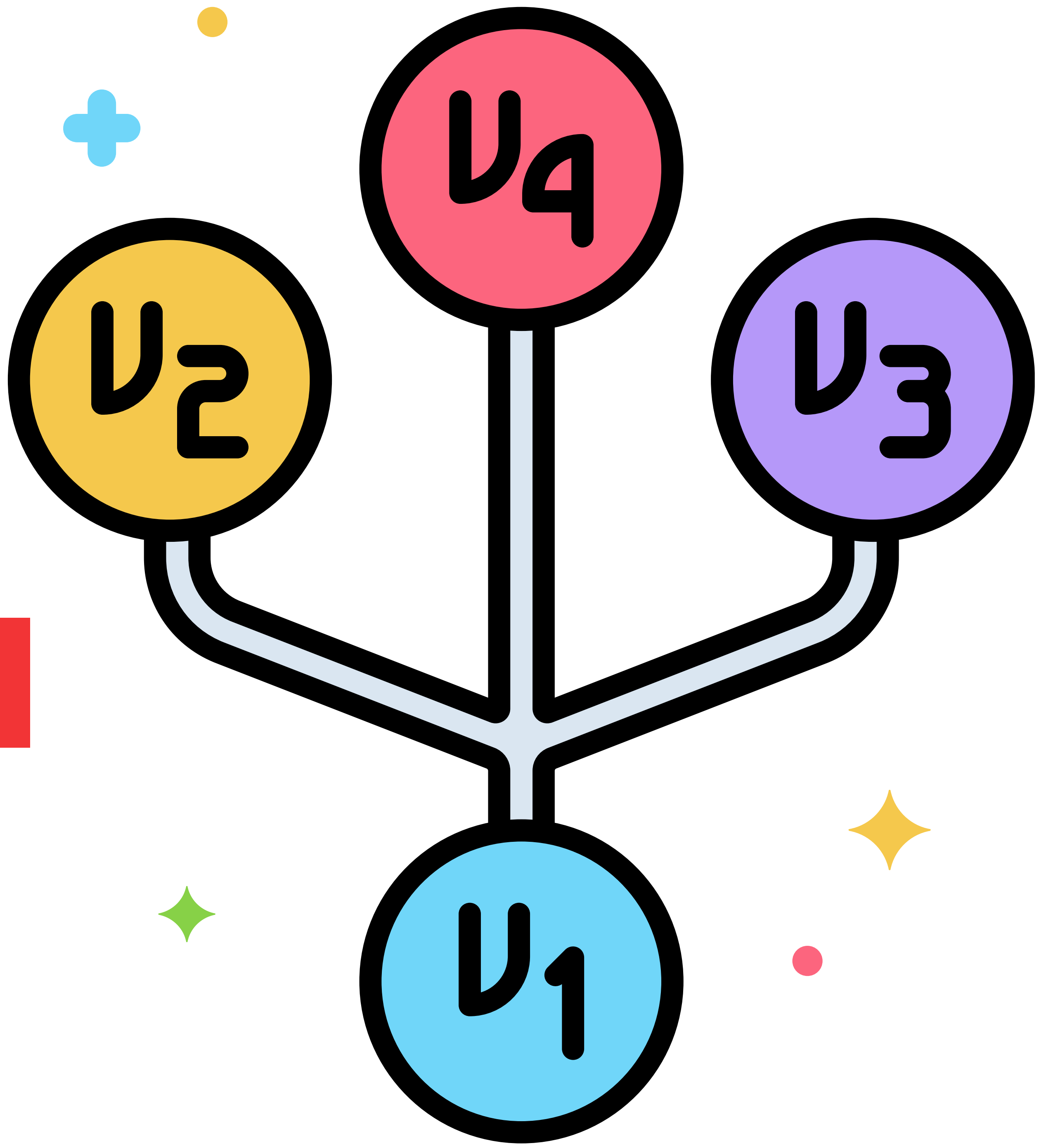
Step 1: **Build**

Step 2: **Test**

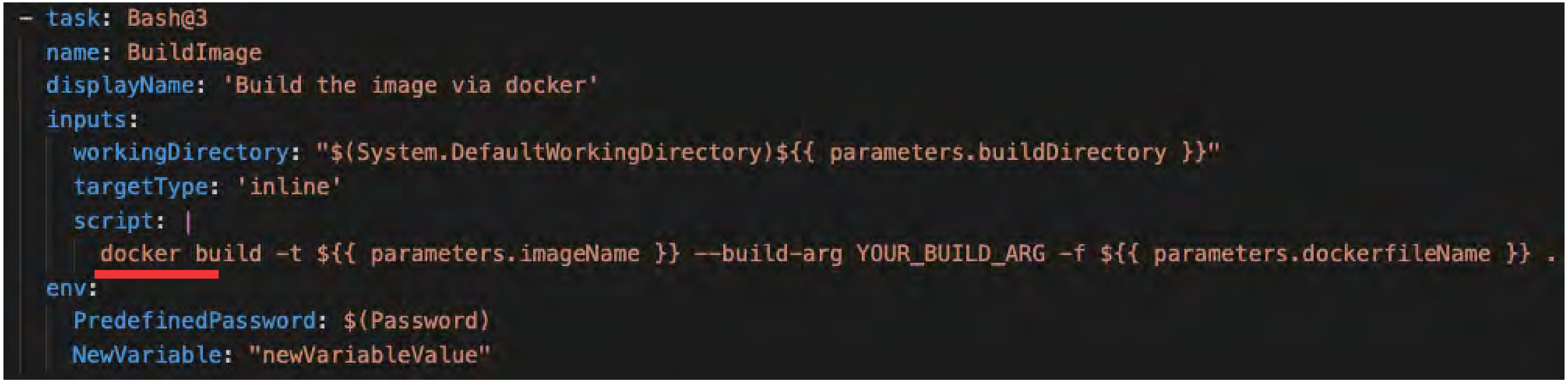
Step 3: **Scan**

Review Autoscan

Step 4: **Version + Publish**

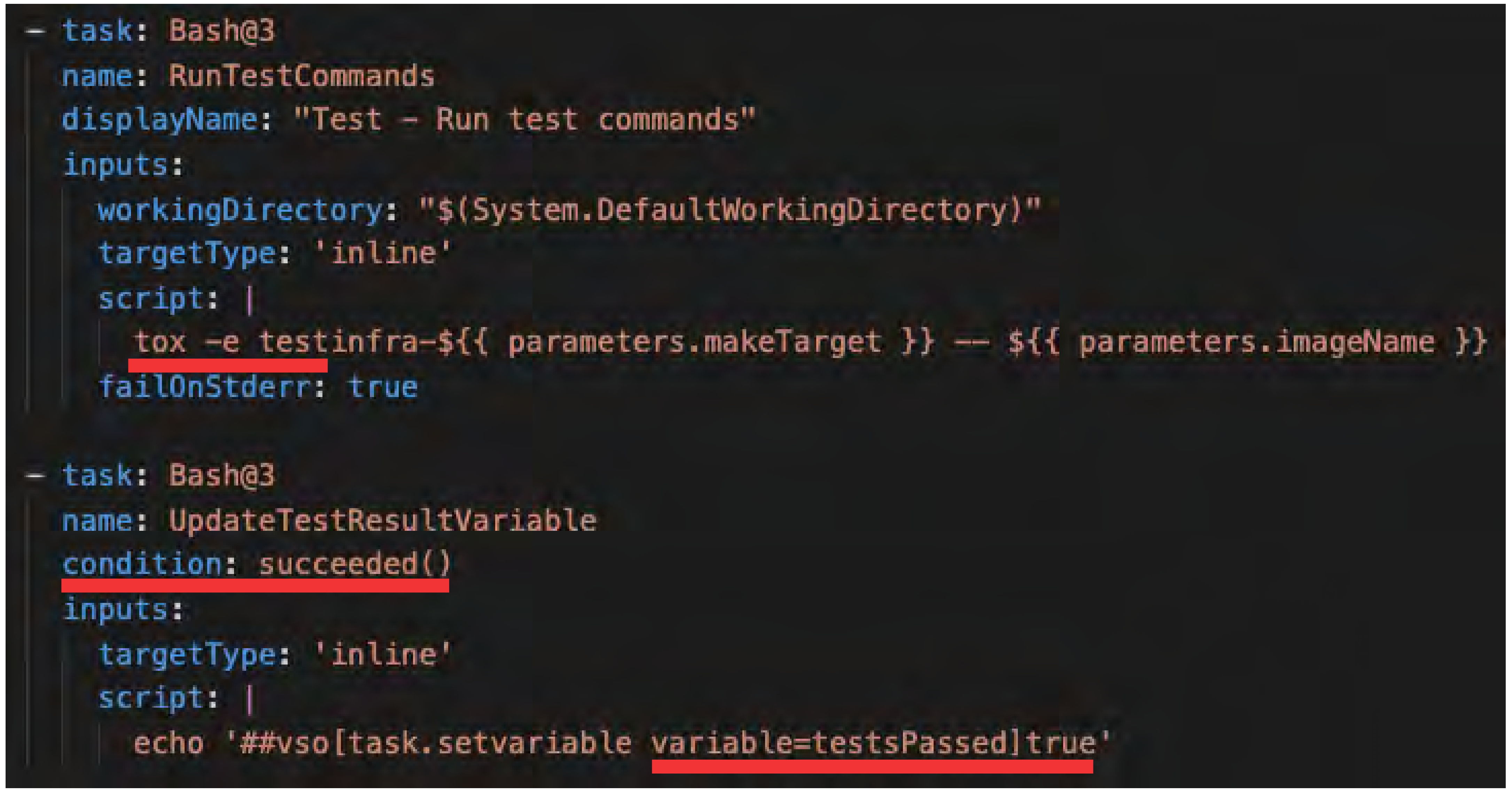


Step 1: **Build**



```
- task: Bash@3
  name: BuildImage
  displayName: 'Build the image via docker'
  inputs:
    workingDirectory: "$(System.DefaultWorkingDirectory)${{ parameters.buildDirectory }}"
    targetType: 'inline'
    script: |
      docker build -t ${{ parameters.imageName }} --build-arg YOUR_BUILD_ARG -f ${{ parameters.dockerfileName }} .
  env:
    PredefinedPassword: $(Password)
    NewVariable: "newVariableValue"
```

Step 2: **Test**



```
- task: Bash@3
  name: RunTestCommands
  displayName: "Test - Run test commands"
  inputs:
    workingDirectory: "$(System.DefaultWorkingDirectory)"
    targetType: 'inline'
    script: |
      tox -e testinfra-${{ parameters.makeTarget }} -- ${{ parameters.imageName }}
    failOnStderr: true

- task: Bash@3
  name: UpdateTestResultVariable
  condition: succeeded()
  inputs:
    targetType: 'inline'
    script: |
      echo '##vso[task.setvariable variable=testsPassed]true'
```

Azure DevOps

Tox init file and command

standardise testing in Python

Tox virtualenv management and test CLI

Triggers pytest command

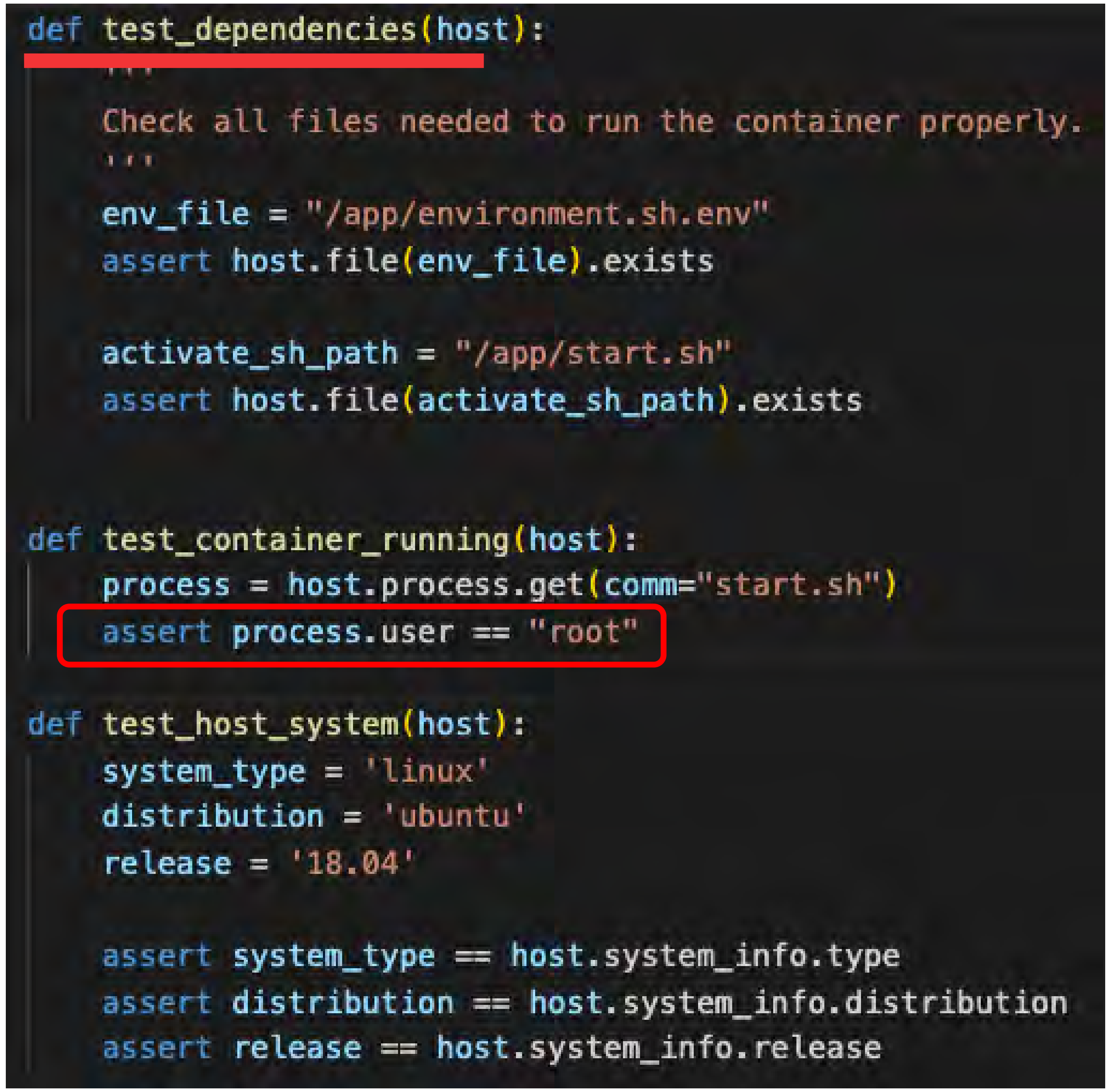
test infra

PyTest testinfra package

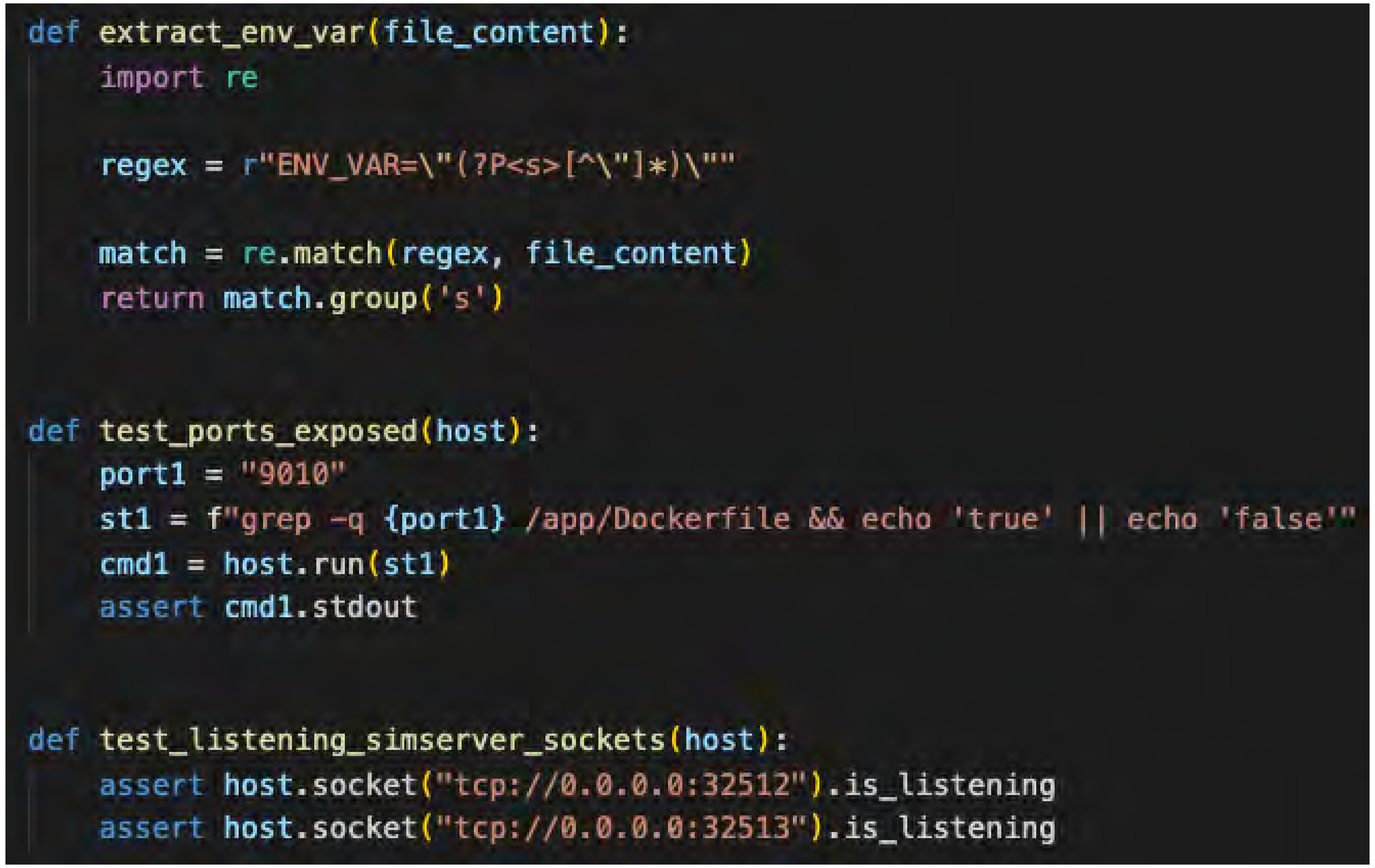
Python container test code

Step 2: **Test**

Python container test code



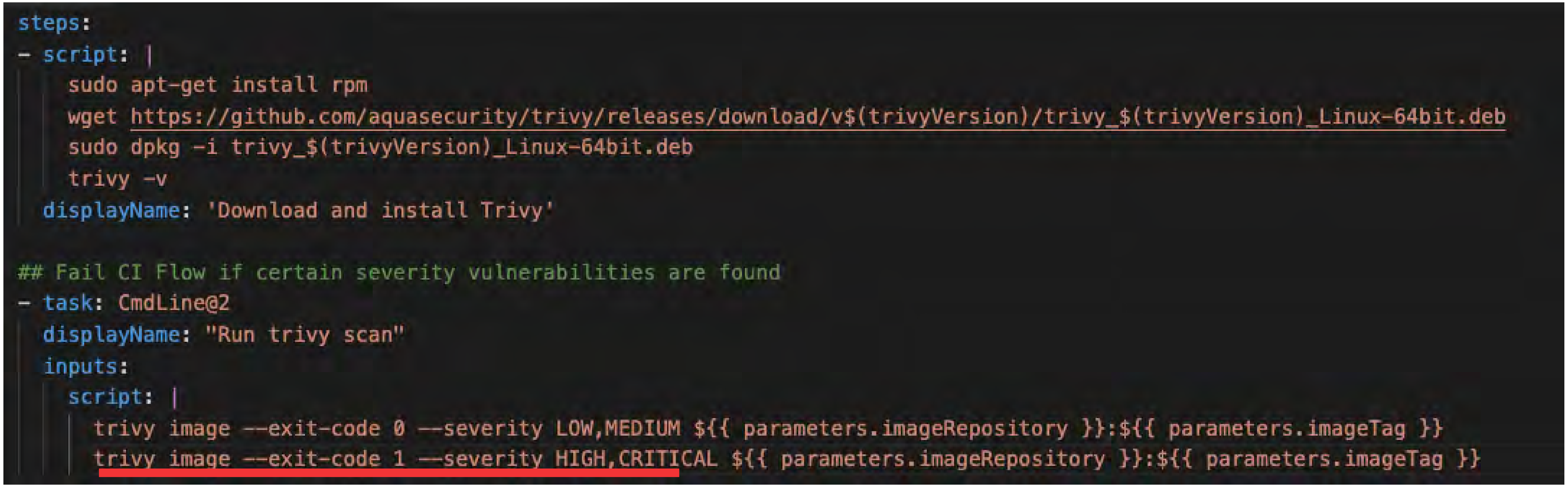
```python
def test_dependencies(host):
    '''
    Check all files needed to run the container properly.
    '''
    env_file = "/app/environment.sh.env"
    assert host.file(env_file).exists

    activate_sh_path = "/app/start.sh"
    assert host.file(activate_sh_path).exists


def test_container_running(host):
    process = host.process.get(comm="start.sh")
    assert process.user == "root"

def test_host_system(host):
    system_type = 'linux'
    distribution = 'ubuntu'
    release = '18.04'

    assert system_type == host.system_info.type
    assert distribution == host.system_info.distribution
    assert release == host.system_info.release
```



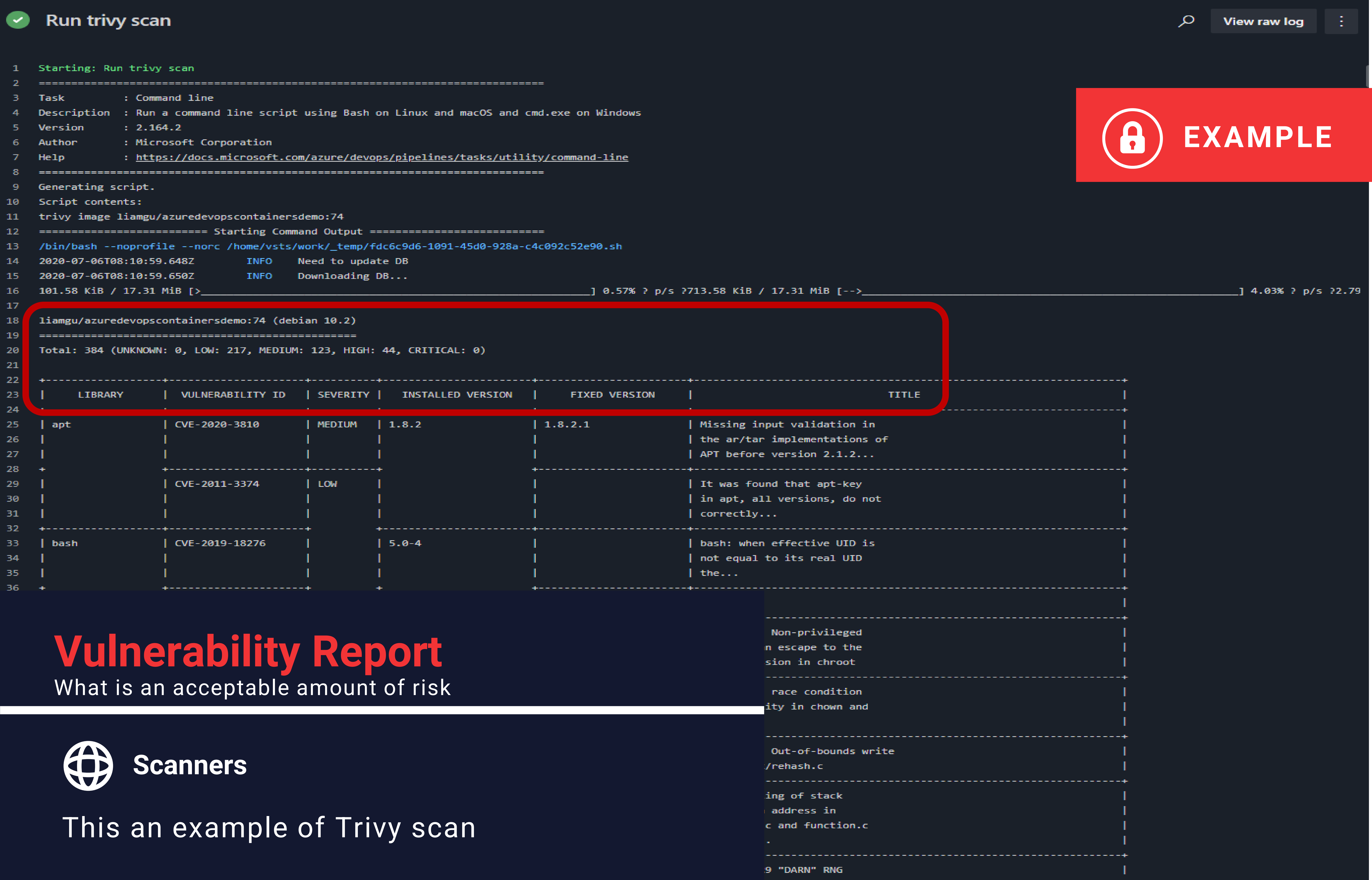
```python
def extract_env_var(file_content):
    import re

    regex = r"ENV_VAR=\"(?P<s>[^\"]*)\""

    match = re.match(regex, file_content)
    return match.group('s')


def test_ports_exposed(host):
    port1 = "9010"
    st1 = f"grep -q {port1} /app/Dockerfile && echo 'true' || echo 'false'"
    cmd1 = host.run(st1)
    assert cmd1.stdout


def test_listening_simserver_sockets(host):
    assert host.socket("tcp://0.0.0.0:32512").is_listening
    assert host.socket("tcp://0.0.0.0:32513").is_listening
```

Step 3: **Scan**

trivy



```
steps:
- script: |
    sudo apt-get install rpm
    wget https://github.com/aquasecurity/trivy/releases/download/v$(trivyVersion)/trivy_$(trivyVersion)_Linux-64bit.deb
    sudo dpkg -i trivy_$(trivyVersion)_Linux-64bit.deb
    trivy -v
  displayName: 'Download and install Trivy'

## Fail CI Flow if certain severity vulnerabilities are found
- task: CmdLine@2
  displayName: "Run trivy scan"
  inputs:
    script: |
      trivy image --exit-code 0 --severity LOW,MEDIUM ${{ parameters.imageRepository }}:${{ parameters.imageTag }}
      trivy image --exit-code 1 --severity HIGH,CRITICAL ${{ parameters.imageRepository }}:${{ parameters.imageTag }}
```



```
Starting: Run trivy scan
==============================================================================
Task         : Command line
Description  : Run a command line script using Bash on Linux and macOS and cmd.exe on Windows
Version      : 2.164.2
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/command-line
==============================================================================
Generating script.
Script contents:
trivy image liamgu/azuredevopscontainersdemo:74
========================== Starting Command Output ===========================
/bin/bash --noprofile --norc /home/vsts/work/_temp/fdc6c9d6-1091-45d0-928a-c4c092c52e90.sh
2020-07-06T08:10:59.648Z        INFO    Need to update DB
2020-07-06T08:10:59.650Z        INFO    Downloading DB...
101.58 KiB / 17.31 MiB [>____________________________________________________________] 0.57% ? p/s ?713.58 KiB / 17.31 MiB [-->____________________________________________________________] 4.03% ? p/s ?2.79

liamgu/azuredevopscontainersdemo:74 (debian 10.2)
=================================================
Total: 384 (UNKNOWN: 0, LOW: 217, MEDIUM: 123, HIGH: 44, CRITICAL: 0)

+------------------+---------------------+----------+-------------------+-------------------+----------------------------------------------+
|     LIBRARY      |  VULNERABILITY ID   | SEVERITY | INSTALLED VERSION |   FIXED VERSION   |                    TITLE                     |
+------------------+---------------------+----------+-------------------+-------------------+----------------------------------------------+
| apt              | CVE-2020-3810       | MEDIUM   | 1.8.2             | 1.8.2.1           | Missing input validation in                  |
|                  |                     |          |                   |                   | the ar/tar implementations of                |
|                  |                     |          |                   |                   | APT before version 2.1.2...                  |
+                  +---------------------+----------+                   +-------------------+----------------------------------------------+
|                  | CVE-2011-3374       | LOW      |                   |                   | It was found that apt-key                    |
|                  |                     |          |                   |                   | in apt, all versions, do not                 |
|                  |                     |          |                   |                   | correctly...                                 |
+------------------+---------------------+          +-------------------+-------------------+----------------------------------------------+
| bash             | CVE-2019-18276      |          | 5.0-4             |                   | bash: when effective UID is                  |
|                  |                     |          |                   |                   | not equal to its real UID                    |
|                  |                     |          |                   |                   | the...                                       |
+                  +---------------------+          +                   +-------------------+----------------------------------------------+
                                                                                                                                            |
                                                                                                -------------------------------------------+
                                                                                                 Non-privileged                            |
                                                                                                n escape to the                            |
                                                                                                sion in chroot                             |
                                                                                                -------------------------------------------+
                                                                                                 race condition                            |
                                                                                                ity in chown and                           |
                                                                                                                                           |
                                                                                                -------------------------------------------+
                                                                                                 Out-of-bounds write                       |
                                                                                                /rehash.c                                  |
                                                                                                -------------------------------------------+
                                                                                                ing of stack                               |
                                                                                                 address in                                |
                                                                                                c and function.c                           |
                                                                                                 .                                         |
                                                                                                -------------------------------------------+
                                                                                                9 "DARN" RNG                               |
```

EXAMPLE

Vulnerability Report

What is an acceptable amount of risk

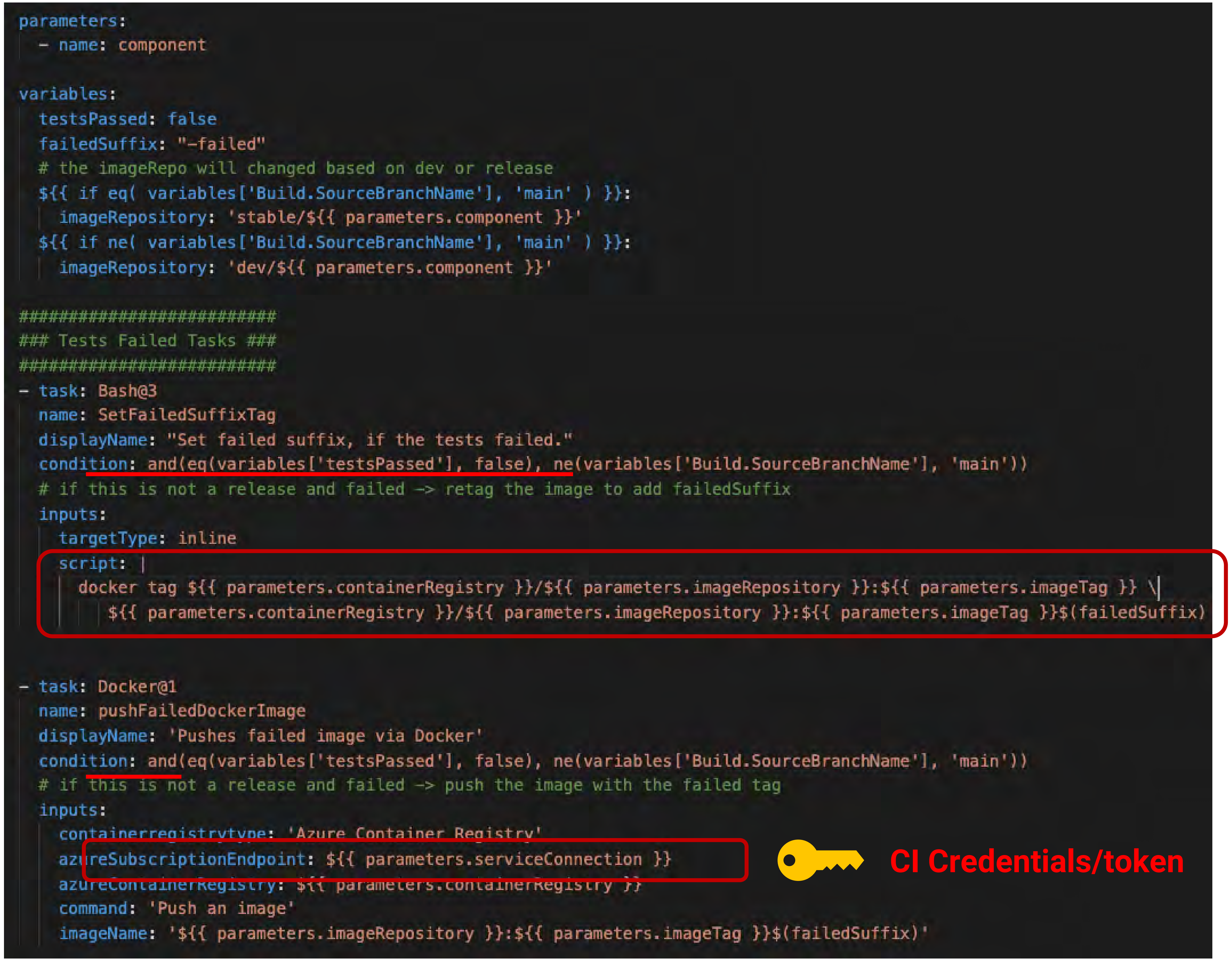
Scanners

This an example of Trivy scan

Container Scanning Tools

- Trivy - a simple and comprehensive vulnerability scanner for containers (doesn't support Windows containers)
- Aqua - dependency and container scanning for applications running on AKS, ACI and Windows Containers. Has an integration with AzDOpipelines.
- Dependency-Check Plugin for SonarQube - OnPrem dependency scanning
- WhiteSource - Open Source Scanning Software

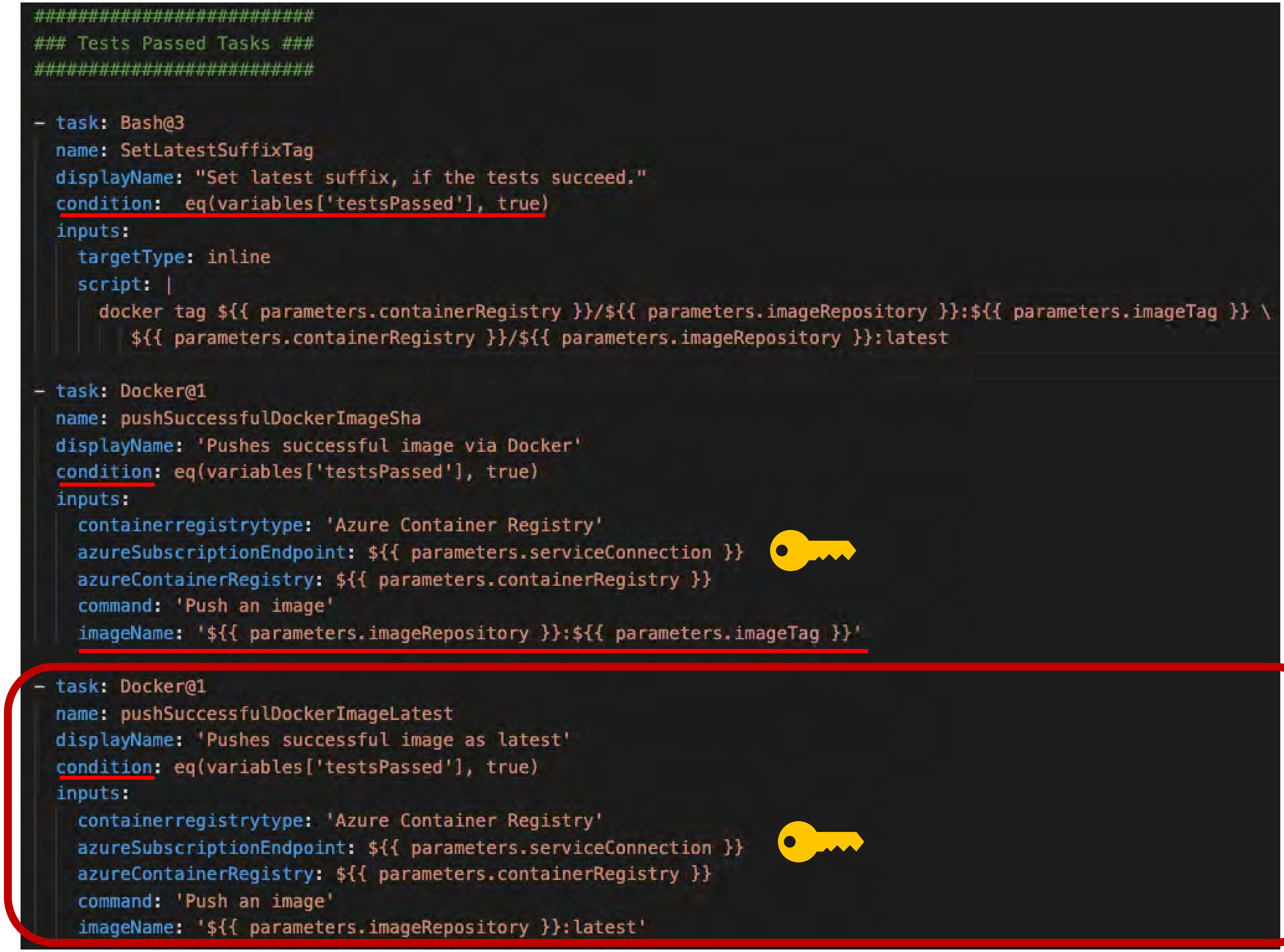
Step 4: Version + Publish



```
parameters:
  - name: component

variables:
  testsPassed: false
  failedSuffix: "-failed"
  # the imageRepo will changed based on dev or release
  ${{ if eq( variables['Build.SourceBranchName'], 'main' ) }}:
    imageRepository: 'stable/${{ parameters.component }}'
  ${{ if ne( variables['Build.SourceBranchName'], 'main' ) }}:
    imageRepository: 'dev/${{ parameters.component }}'

##########################
### Tests Failed Tasks ###
##########################
- task: Bash@3
  name: SetFailedSuffixTag
  displayName: "Set failed suffix, if the tests failed."
  condition: and(eq(variables['testsPassed'], false), ne(variables['Build.SourceBranchName'], 'main'))
  # if this is not a release and failed -> retag the image to add failedSuffix
  inputs:
    targetType: inline
    script: |
      docker tag ${{ parameters.containerRegistry }}/${{ parameters.imageRepository }}:${{ parameters.imageTag }} \
         ${{ parameters.containerRegistry }}/${{ parameters.imageRepository }}:${{ parameters.imageTag }}$(failedSuffix)


- task: Docker@1
  name: pushFailedDockerImage
  displayName: 'Pushes failed image via Docker'
  condition: and(eq(variables['testsPassed'], false), ne(variables['Build.SourceBranchName'], 'main'))
  # if this is not a release and failed -> push the image with the failed tag
  inputs:
    containerregistrytype: 'Azure Container Registry'
    azureSubscriptionEndpoint: ${{ parameters.serviceConnection }}
    azureContainerRegistry: ${{ parameters.containerRegistry }}
    command: 'Push an image'
    imageName: '${{ parameters.imageRepository }}:${{ parameters.imageTag }}$(failedSuffix)'
```

CI Credentials/token

Step 4: Version + Publish



```
########################
### Tests Passed Tasks ###
########################

- task: Bash@3
  name: SetLatestSuffixTag
  displayName: "Set latest suffix, if the tests succeed."
  condition:  eq(variables['testsPassed'], true)
  inputs:
    targetType: inline
    script: |
      docker tag ${{ parameters.containerRegistry }}/${{ parameters.imageRepository }}:${{ parameters.imageTag }} \
         ${{ parameters.containerRegistry }}/${{ parameters.imageRepository }}:latest

- task: Docker@1
  name: pushSuccessfulDockerImageSha
  displayName: 'Pushes successful image via Docker'
  condition: eq(variables['testsPassed'], true)
  inputs:
    containerregistrytype: 'Azure Container Registry'
    azureSubscriptionEndpoint: ${{ parameters.serviceConnection }}
    azureContainerRegistry: ${{ parameters.containerRegistry }}
    command: 'Push an image'
    imageName: '${{ parameters.imageRepository }}:${{ parameters.imageTag }}'

- task: Docker@1
  name: pushSuccessfulDockerImageLatest
  displayName: 'Pushes successful image as latest'
  condition: eq(variables['testsPassed'], true)
  inputs:
    containerregistrytype: 'Azure Container Registry'
    azureSubscriptionEndpoint: ${{ parameters.serviceConnection }}
    azureContainerRegistry: ${{ parameters.containerRegistry }}
    command: 'Push an image'
    imageName: '${{ parameters.imageRepository }}:latest'
```

Where Does Security Live in Containers

Securing Production

Once a Container is in use, how do we ensure it remains secure?

Container DevSecOps

How do we scan images for vulnerabilities? How do we test images?

Enforce network segmentation

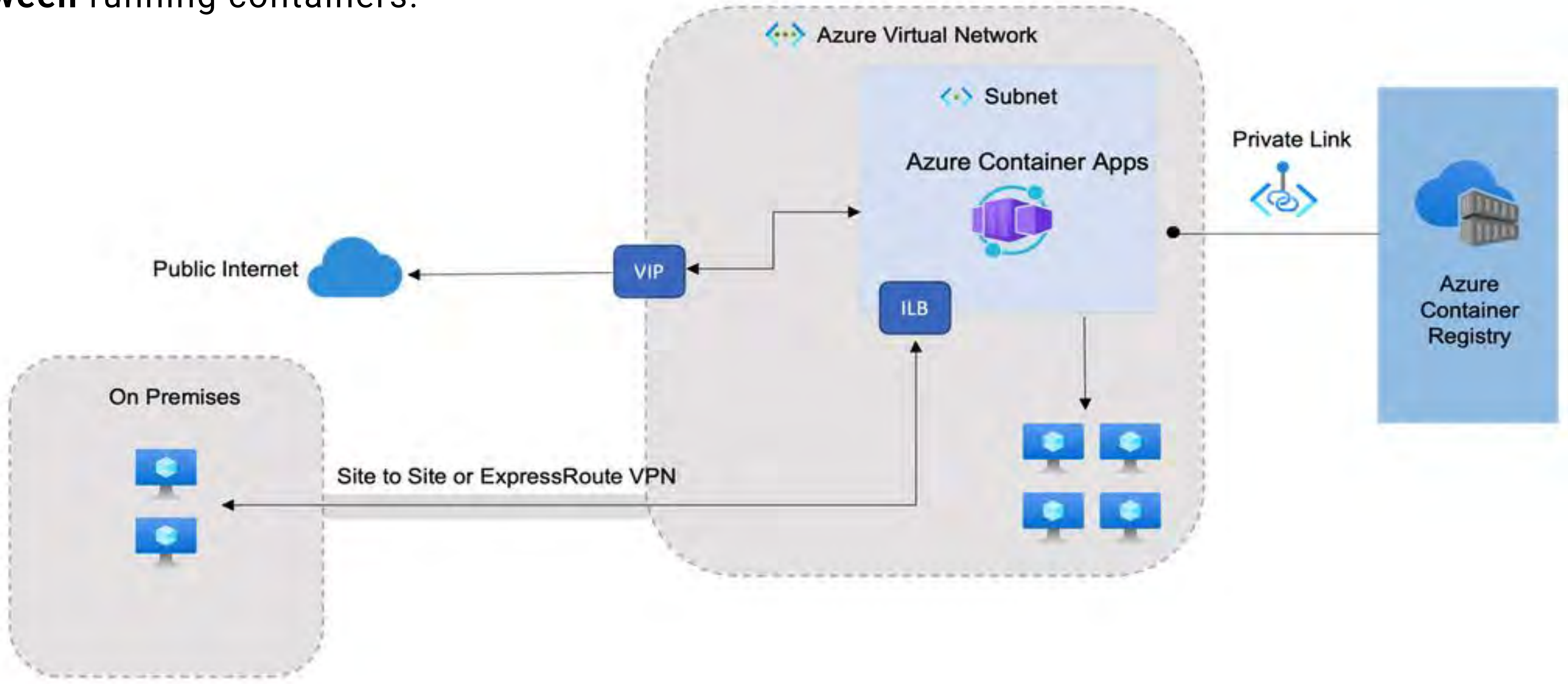
Configure resource quotas

Continuous Monitor container activity

- container user access
- container resource activity

Network Segmentation

Network segmentation (or nano-segmentation) or segregation **between** running containers.



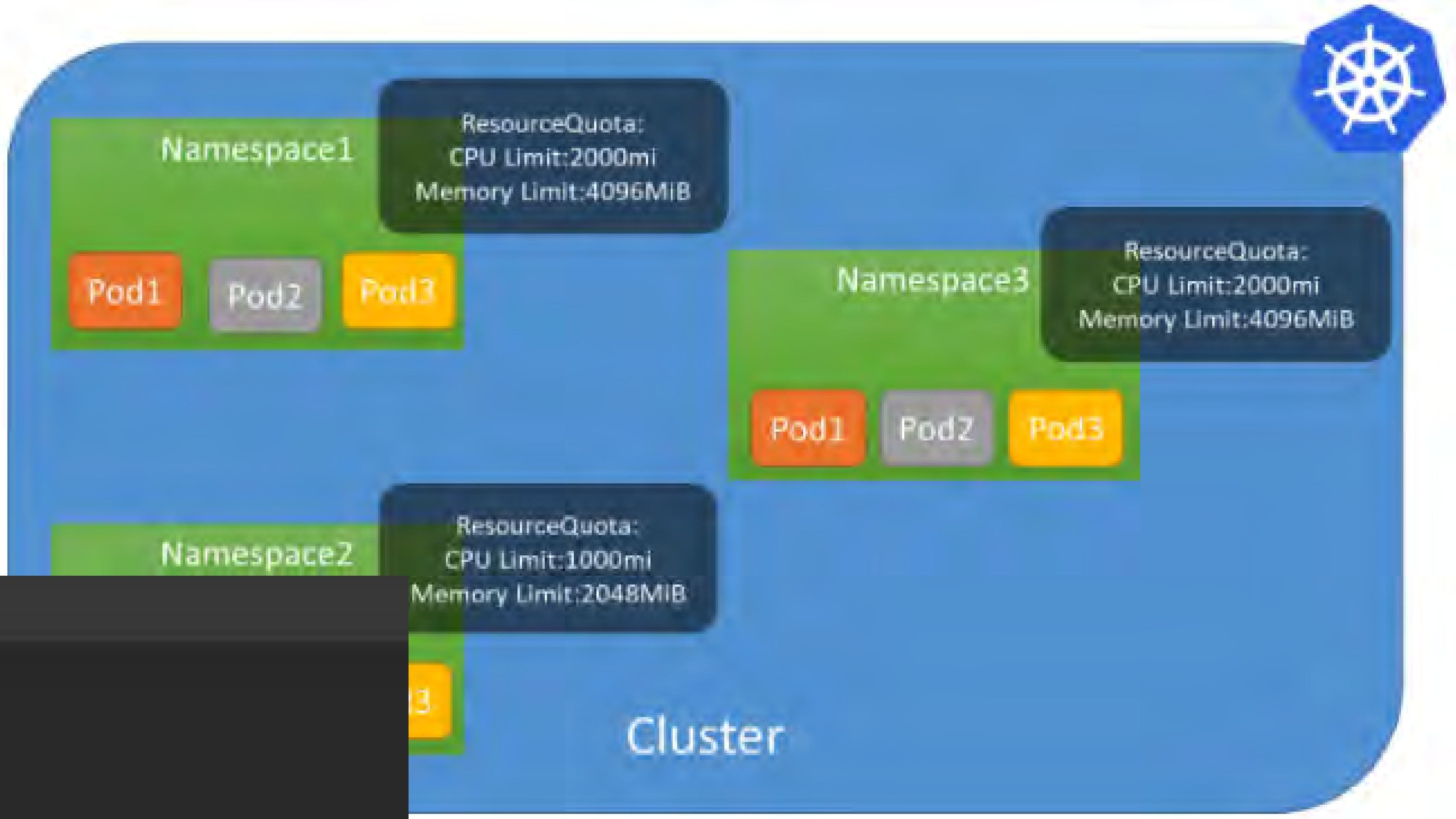
Resource Quotas

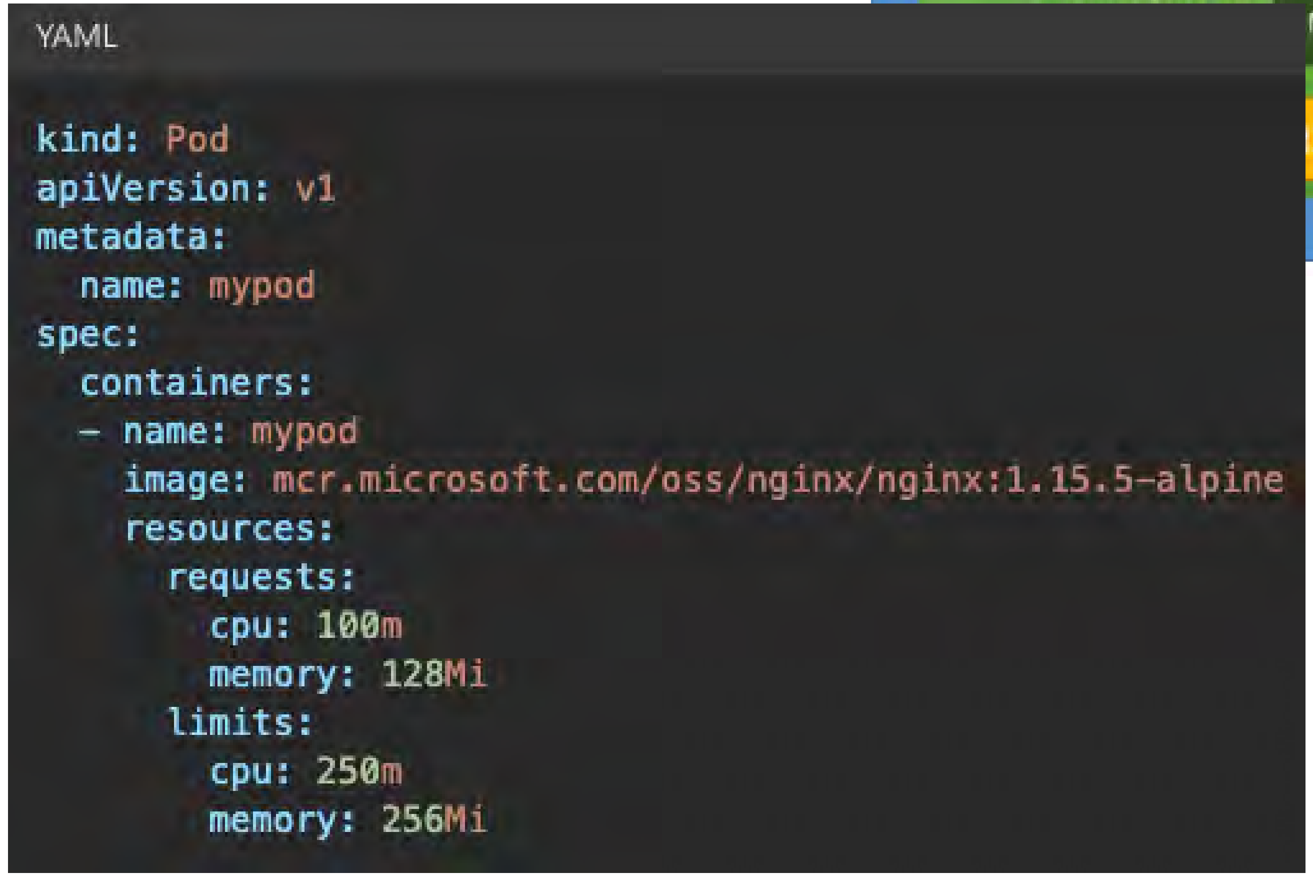
By default, a container has **no resource constraints**.

Mitigations:

- Docker
  - Limit container's access to memory
  - Limit container's access to CPU resources

- Kubernetes
  - Namespace CPU/Memory/Storage quotas
  - Container (Pod) request limits and CPU/Memory limits




YAML

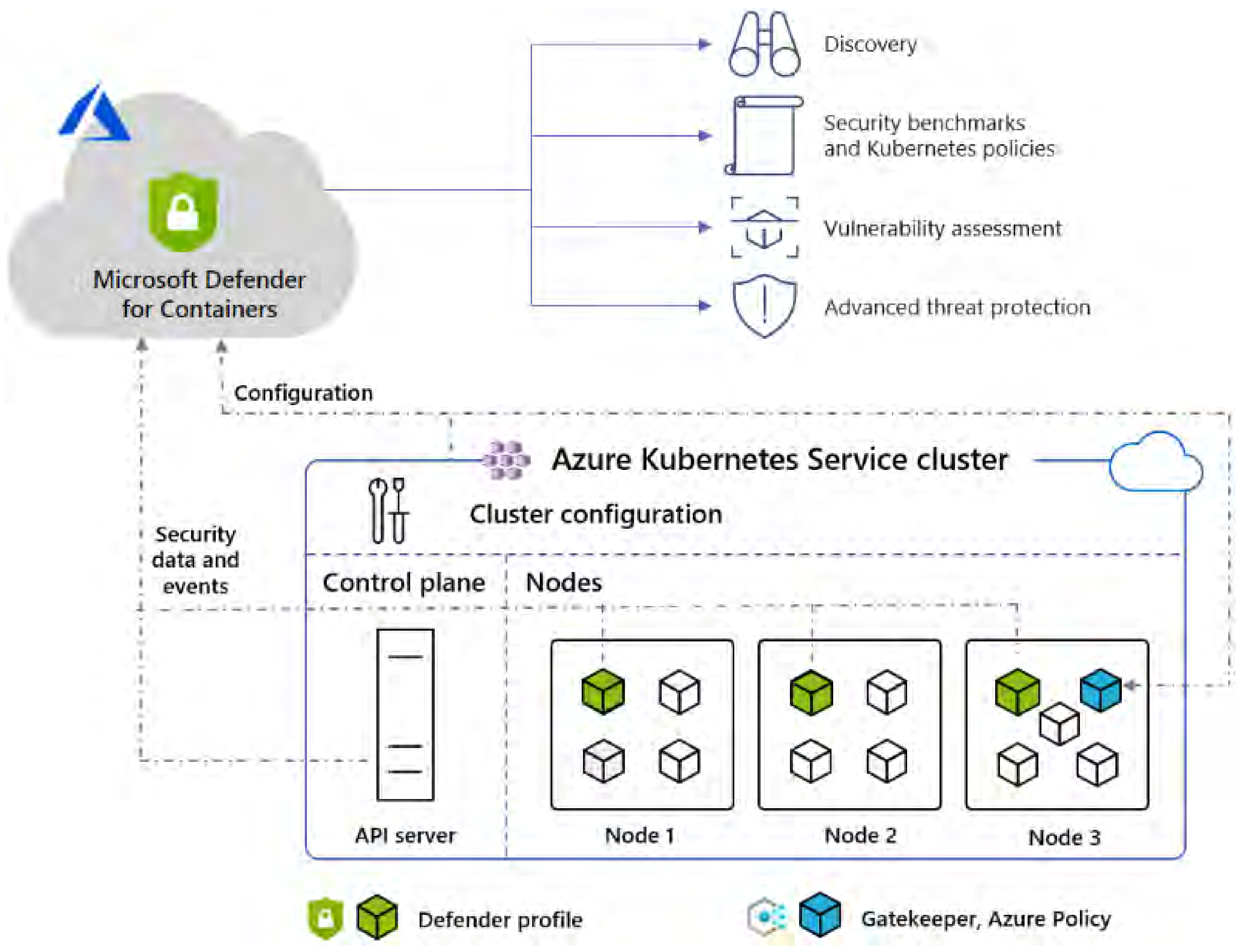
```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: mcr.microsoft.com/oss/nginx/nginx:1.15.5-alpine
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 250m
        memory: 256Mi
```

Container Monitoring

Environment hardening
Continuously assess clusters to provide visibility into misconfigurations and guidelines to help mitigate threats.

Vulnerability assessment
Vulnerability assessment and management tools for images stored in registries and running in hosting platform (Azure Kubernetes Service).

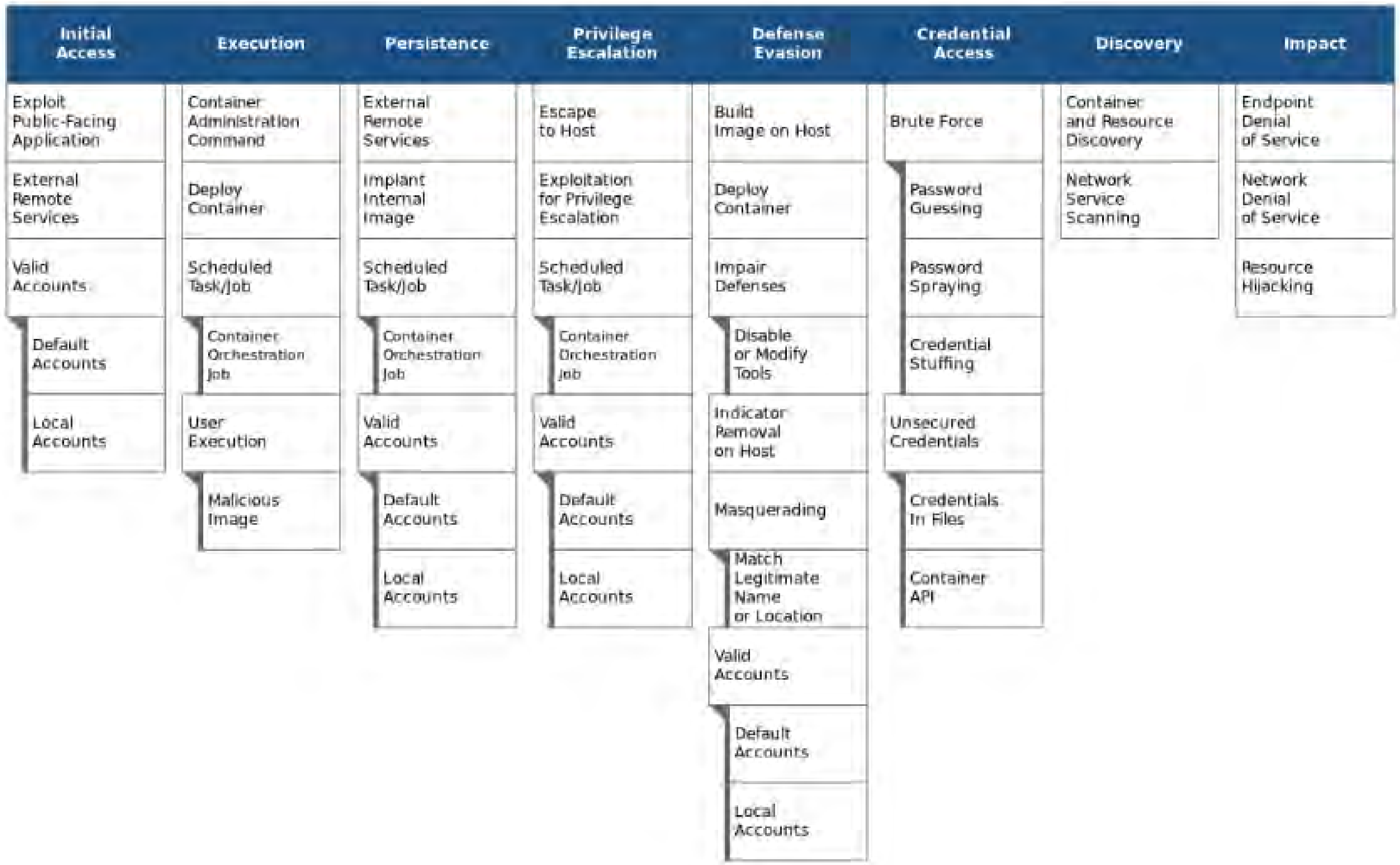
Run-time threat protection for nodes and clusters
Threat protection for clusters and Linux nodes generates security alerts for suspicious activities.



Run-time Protection

Container Monitoring

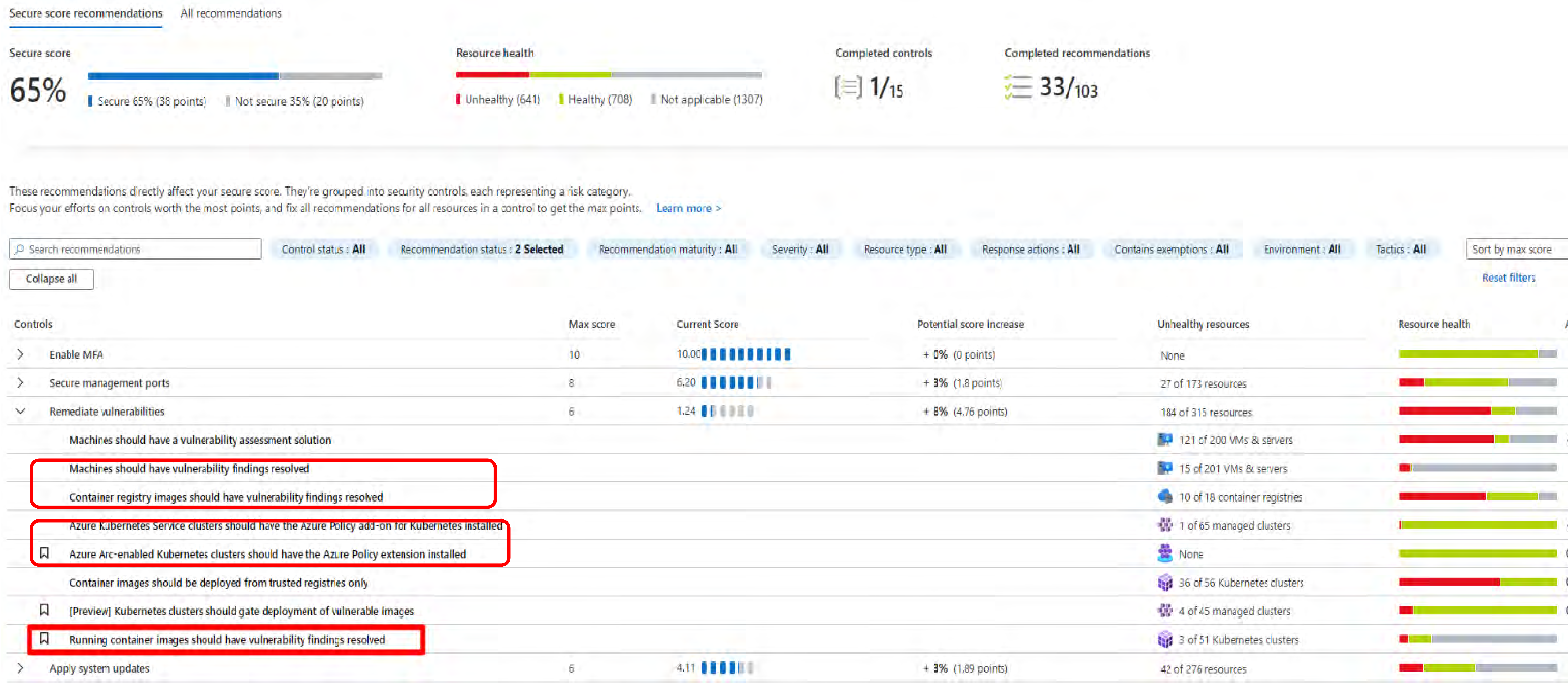
[Center for Threat-Informed Defense teams up with Microsoft, partners to build the ATT&CK® for Containers matrix - Microsoft Security Blog]



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Impact
Exploit Public-Facing Application	Container Administration Command	External Remote Services	Escape to Host	Build Image on Host	Brute Force	Container and Resource Discovery	Endpoint Denial of Service
External Remote Services	Deploy Container	Implant Internal Image	Exploitation for Privilege Escalation	Deploy Container	Password Guessing	Network Service Scanning	Network Denial of Service
Valid Accounts	Scheduled Task/Job	Scheduled Task/Job	Scheduled Task/Job	Impair Defenses	Password Spraying		Resource Hijacking
Default Accounts	Container Orchestration Job	Container Orchestration Job	Container Orchestration Job	Disable or Modify Tools	Credential Stuffing		
Local Accounts	User Execution	Valid Accounts	Valid Accounts	Indicator Removal on Host	Unsecured Credentials		
	Malicious Image	Default Accounts	Default Accounts	Masquerading	Credentials In Files		
		Local Accounts	Local Accounts	Match Legitimate Name or Location	Container API		
				Valid Accounts			
				Default Accounts			
				Local Accounts			

Vulnerability Assessment

Container Monitoring



Secure score recommendations | All recommendations

Secure score **65%** ▮ Secure 65% (38 points) ▮ Not secure 35% (20 points)

Resource health ▮ Unhealthy (641) ▮ Healthy (708) ▮ Not applicable (1307)

Completed controls 1/15

Completed recommendations 33/103

These recommendations directly affect your secure score. They're grouped into security controls, each representing a risk category.
Focus your efforts on controls worth the most points, and fix all recommendations for all resources in a control to get the max points. Learn more >

Search recommendations | Control status : **All** | Recommendation status : **2 Selected** | Recommendation maturity : **All** | Severity : **All** | Resource type : **All** | Response actions : **All** | Contains exemptions : **All** | Environment : **All** | Tactics : **All** | Sort by max score

Collapse all | Reset filters

Controls	Max score	Current Score	Potential score increase	Unhealthy resources	Resource health
Enable MFA	10	10.00	+ **0%** (0 points)	None	
Secure management ports	8	6.20	+ **3%** (1.8 points)	27 of 173 resources	
Remediate vulnerabilities	6	1.24	+ **8%** (4.76 points)	184 of 315 resources	
Machines should have a vulnerability assessment solution				121 of 200 VMs & servers	
Machines should have vulnerability findings resolved				15 of 201 VMs & servers	
Container registry images should have vulnerability findings resolved				10 of 18 container registries	
Azure Kubernetes Service clusters should have the Azure Policy add-on for Kubernetes installed				1 of 65 managed clusters	
Azure Arc-enabled Kubernetes clusters should have the Azure Policy extension installed				None	
Container images should be deployed from trusted registries only				36 of 56 Kubernetes clusters	
[Preview] Kubernetes clusters should gate deployment of vulnerable images				4 of 45 managed clusters	
Running container images should have vulnerability findings resolved				3 of 51 Kubernetes clusters	
Apply system updates	6	4.11	+ **3%** (1.89 points)	42 of 276 resources	

Microsoft CSE Engineering Playbook

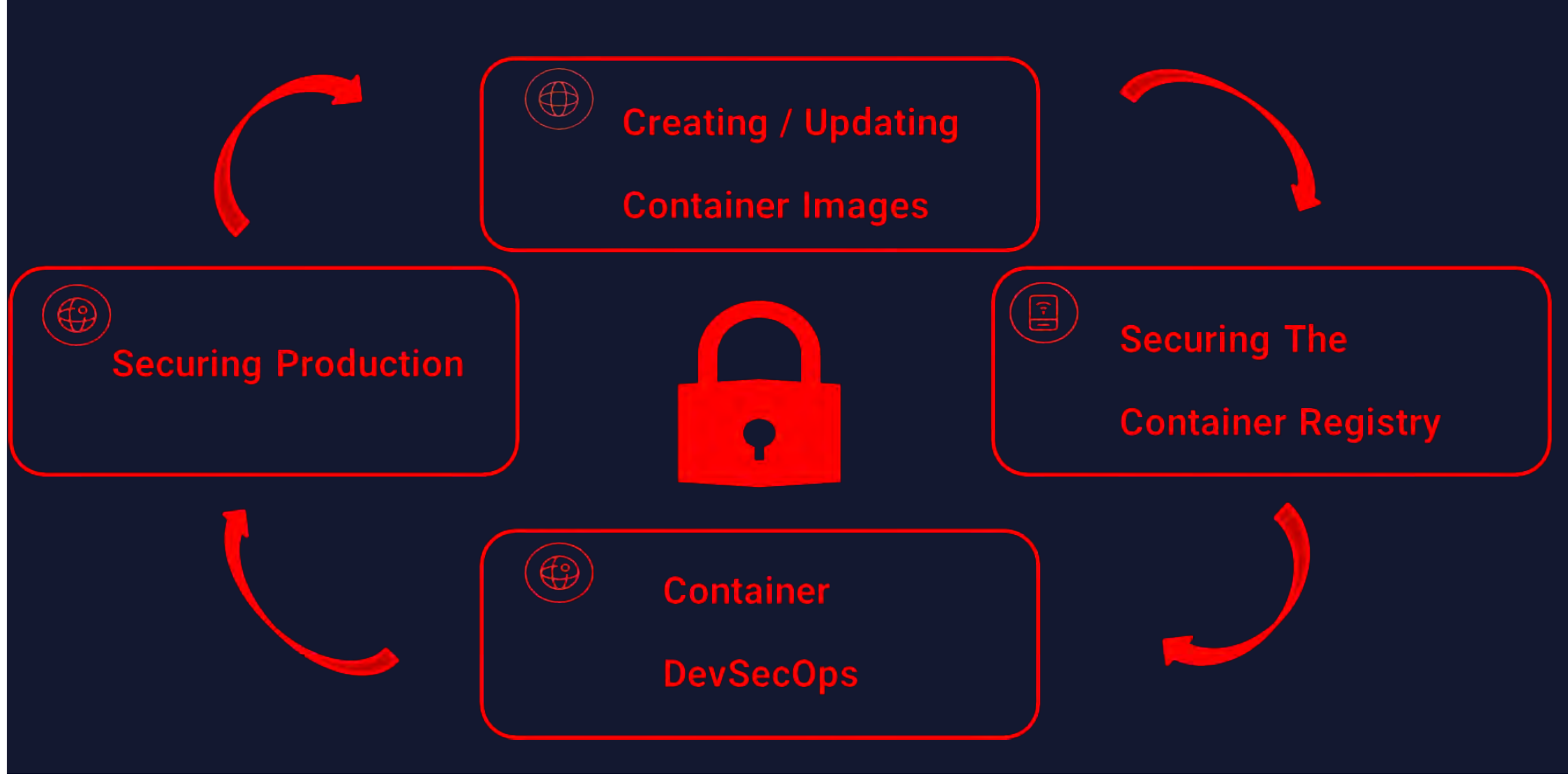
A collection of fundamentals, frameworks, and principals that guides developers and teams to deliver high quality solutions.

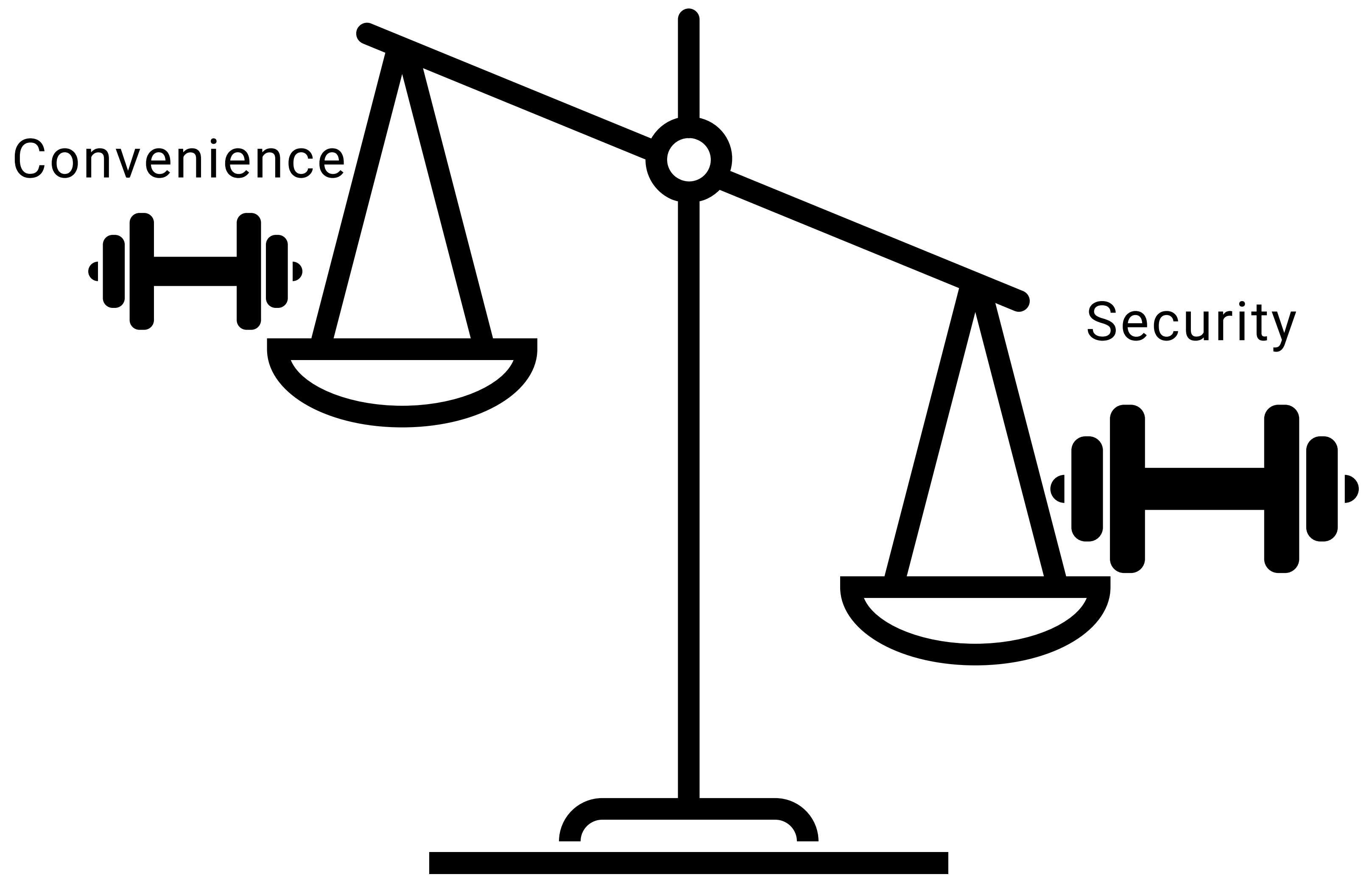
Microsoft Open Source DevContainers

Repository contains a set of DevContainer Images which are Docker images built with certain features for various use cases

Takeaways

- Ensure team-wide awareness on Container DevSecOps practices
- Enforce RBAC to prevent potential disabling of “control gates”
- Hold all members of the team accountable for adhering to secure container management
- Influencing change is most effective when done as a community
- Start with **weighting Security more** than Convenience. It’s less costly to shift balance this way


Creating / Updating Container Images
Securing The Container Registry
Container DevSecOps
Securing Production



Let's Connect!

Thank You

linkedin.com/in/**adrian-g-gonzalez/**