VOLKSWAGEN

AKTIENGESELLSCHAFT

Quality Assurance and Traceability in Containerized Continuous Delivery Process

Edinburgh, Sept. 2019

Agenda

Agenda

Containerization process and software quality assurance

What is containerization?

What benefit you get from containerization?

How to use containerization?

How looks a typical container-based delivery process?

Why quality assurance is important?

What is containerization?

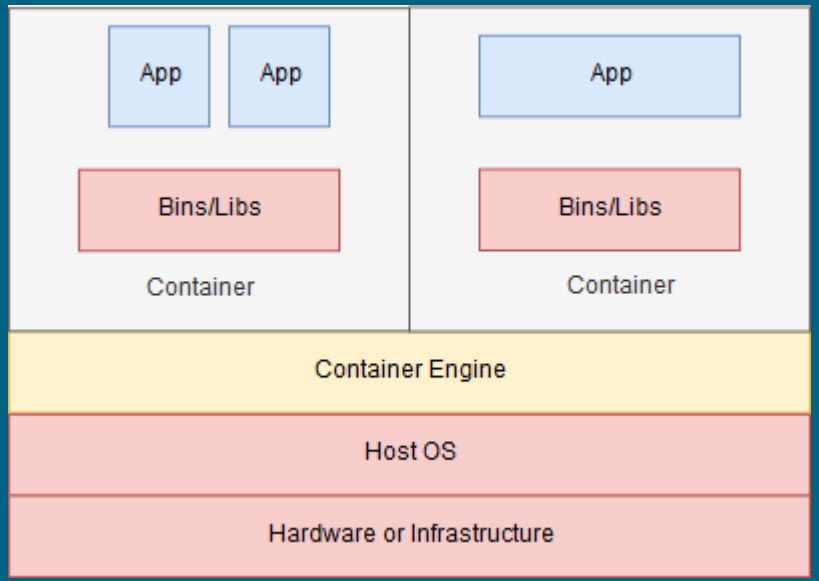
- Virtualization technology based separation of execution environments
  → software workloads like applications share common infrastructure
- Resources of infrastructure are assigned to specific workloads
  → individual amount of resources can be defined for each workload
- Starting containers is lightweight
  → predefined application workloads are “stored” as container-image

→ different workloads can easily share common (hardware) infrastructure

What is containerization?

- Lightweight virtualization of infrastructure

→ Needs a process to manage the configuration and for traceability



What benefit do you get from containerization?

- Container are fast to launch, destroy and run
  → add an additional workload container to use resources more efficient
- Dedicated assignment of infrastructure resources to a container
  - Define minimum resources → “define small machines for workloads”
  - Realize fast scaling → allocate only necessary infrastructure
  - Have pre-configured and installed images → fast start of services

→ High flexibility on start, run and stop different workloads on infrastructure

How to use containerization?

- Define workload environment and set parameters of the application
  → bake container image
- Store the container image in a “ready to use” state
  → use an image store (a public hub or private one)
- Setup the image instance as running container
  → fast start of the workload

→ Easy way to distribute software in a parameterized fashion

How does a typical container-based delivery process looks?

- Developer
  → initiate with a commit to the build process of the software
- Developer
  → package the software for deployment / distribution
- Developer or Operation
  → packs the image
- Operation
  → start the container based on the offered image

→ Process focus on fast delivery of new features and capabilities

Why quality assurance is important

- Reduce failures in production
  assure that fast delivery capabilities do not lead to bad quality
  → establish automated quality checks in the forward chain

- Root-cause issues and solve them efficient
  ensure capability for efficient and fast issue-analysis
  → establish systematic configuration management with traceability

→ Identify relevant quality assurance stages

Agenda

01 Containerization Process and Software Quality Assurance

02 Traceability in Containerized Delivery Process

03 Generic Methodology to Bi-Directional Traceability Implementation

04 Summary & outlook

Objective of traceability

Wikipedia defines bi-directional traceability

- In systems and software development, the term traceability (or Requirements Traceability) refers to the ability to link product requirements **back** to stakeholders' rationales and **forward** to *corresponding design artifacts, code, and test cases.*

We need to add to the definition

- *…. corresponding design artifacts, code, test cases* ***and deployment artifacts.***

Objective for automation

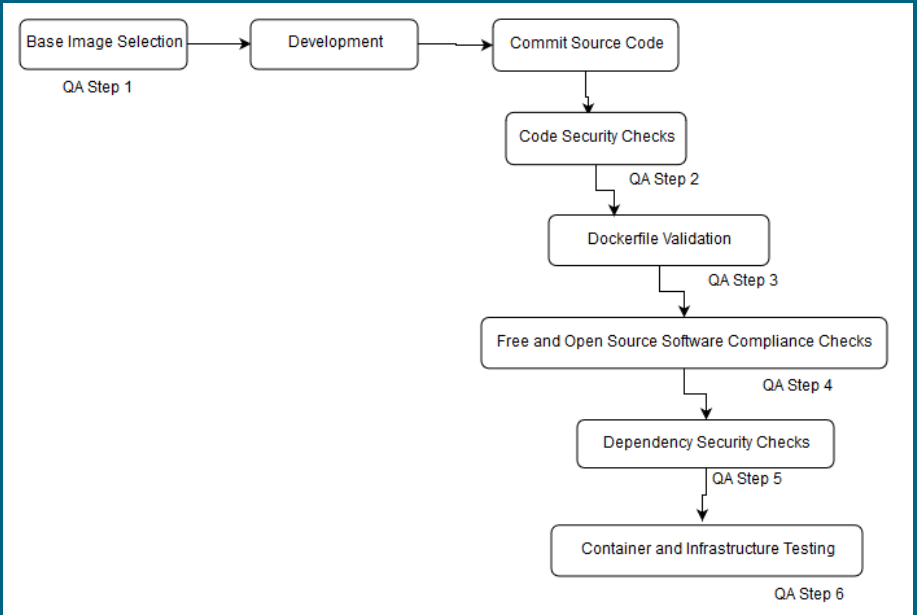
Setting

- Deployments on stages are automated triggered

Demands

- For transparency (compliance case) traceability is needed
- In case of issues, a root-cause analysis back to the failure is needed

A generic workflow for quality assurance of a containerization process



Base image selection

Mostly manual decision

- Engineer select pre-baked certified/validated image as base
- Image has to fulfill functional and non-functional demands

Selection defines

- Technology base and
- External dependency of the product

Code security checks

Mostly automated

- Tools identify risky patterns like described on OWASP* recommendations

Findings have to be handled manually

- Evaluate the risks of the findings
- Decide how to mitigate the risks

* OWASP - Open Web Application Security Project

Dockerfile validation

Mostly automated

- Tools running to check alignment with best practices
- Tools apply specific practices checks (domain, company defined)

Findings have to be handled manually

- Evaluate demand to act in the specific context
- Decide and apply actions

FOSS* compliance checks

Mostly automated

- Tools running to identify the included FOSS artifacts
- Tools checks against predefined “OK”-license-list

Findings have to be handled manually

- Evaluate alternative options
- Decide and apply actions

*FOSS – Free and Open Source Software

Security checks

Mostly automated

- Tools running to check against know issues of libs and other artifacts

Findings have to be handled manually

- Evaluate the risks of the findings
- Decide how to mitigate the risks

Container infrastructure testing

Mostly automated

- Tools running to check against know issues based on CIS* etc.

Findings have to be handled manually

- Evaluate the risks of the findings
- Decide how to mitigate the risks

*CIS – Center of Information Security

Agenda

01 Containerization Process and Software Quality Assurance

02 Traceability in Containerized Delivery Process

03 Generic Methodology to Bi-Directional Traceability Implementation

04 Summary & outlook

Tag vs Digest

Tag

- Human readable
- Defines a release stage
- Can be changed

Digest (pinning)

- Unique and fix ID
- Sign the image and
- Rollback to a specific version

How it works?

Digest: X, Y, Z etc. for a unique and fix ID

Tag: a, b, c etc. for a passed stage

Attempts/ Procedure Steps	1	2	3	4	5	6	Final Artifacts
I	X	Xa	Xab				Xab
II	Y	Ya	Yab	Yabc			Yabc
III	Z	Za	Zab	Zabc	Zabcd	Zabcde	Zabcde

Outcome: structured process and transparency with automation

Deterministic procedure

Integrated configuration management

Bi-directional traceability

Propagation based on safeguarding

Failure root-cause analysis support

Statistical analysis of deployments

Data for process improvement

Agenda

01 Containerization Process and Software Quality Assurance

02 Traceability in Containerized Delivery Process

03 Generic Methodology to Bi-Directional Traceability Implementation

04 Summary & outlook

Summary

Systematic containerization needs processes

- High automation of workflows for fast “positive” propagation
- Mostly manual handling of “negative” checks

Traceability have to be build in

- To be able to check root-cause issues of the automated propagations
- To know what is running where

Outlook

Higher automation

- of issue handling with for example better classification of risks
- In suggestions to handle issues with derivation patterns

Work on the objective:

Establish more DevOps for faster and better delivery

Thank you for your interest! – Do you have any questions?

We are hiring...

- for our Group IT in Wolfsburg
- for our labs in Berlin, Munich, Wolfsburg and all over the world!

Join us at www.join-it-volkswagen.com