Security

Build Pipeline

description:

- typically the least secure, security in development slows developers down
- malicious source code changes
- malicious changes to build scripts (expose credentials)

good practice:

- remove libraries and unneeded packages - reduces attack surface
- check libraries against the Common Vulnerabilities and Exposures (CVE)
- Code Analysis
- scan containers for security vulnerabilities

Container Contents

description:

- what is installed inside the container?
- open ports, security bugs in installed software

good practice:

- don't run container processes as 'root', narrow down roles and access privileges
- check for hard-coded passwords, keys, and other sensitive items in the container

Container Runtime

description:

- Think of Docker Daemon. PodSandbox.

good practice:

- limit external endpoints and who has access to the container
- use trusted image repository
- containers have a 'time to live' - new vulnerabilities may have been discovered.
- one image may instantiate into a thousand running containers - regularly patch and replace images and keep containers in sync
- enable AlwaysPullImages: images are always pulled with the correct authorization and cannot be re-used by other pods without first providing credentials.

Operating System

description:

- If an attack on the host platform succeeds it's pretty much game over for that cluster of containers, and may give malicious code sufficient access to attack other systems.

good practice:

- Host OS/Kernel Hardening: patch management

Orchestration Platform

description:

- incredibly complex
- focus on scalability and ease of management — not security
- insecure default configurations
- permission escalation
- code injection vulnerabilities

good practice:

- use latest Kubernetes version (critical bug fixes and new security features)
- isolate resources using namespaces
- use network policies: configure how components communicate among each other and with outside endpoints (pods, containers, service, namespaces) https://ahmet.im/blog/kubernetes-network-policy/
- use Network plugin (custom network interface - CNI): TLS, NSG, VNET
- use secrets: passwords, tokens or keys should always be stored in Kubernetes secrets objects (KeyVault)
- Authentication: use an identity provider (basic authentication, certificates, bearer token - Active Directory, OAuth)
- Authorization: use RBAC to limit access to Kubernetes Resources (ServiceAccount, Role, RoleBinding), avoid giving admin access as much as possible
- disable default service account: default service account is automatically assigned to new pods - has a wide range of permissions
- use PodSecurityPolicy: e.g. enforce pods are not able to run in privileged context, pods cannot bind to hostNetwork, pods must run as a particular user
- use securityContext in Pods/Deployments: runAsNonRoot, readOnlyRootFileSystem, allowPriviligeEscalation, capabilities (drop:["all"], add["NET_BIND_SERVICE"])
- use ResourceQuota to limits resource consumption for each namespace (CPU, MEM, how many pods)

Monitoring

description:

- discover the unexpected stuff
- learn what is effective, track what's really happening in your environment, and detect what's broken.

good practice:

- enable Kubernetes audit logging: record a sequence of activities that users or system components perform
- enable Kubernetes logging: understand what is happening inside your cluster as well as debug problems and monitor activity