Analyzis of CNCF CNF Testsuite tests for RA2

This page contains an analyzis on the list of test cases listed in the CNCF CNF Testsuite to determine if RA2 should contain related workload requirements.

Each test should be clearly documented - there is no documentation currently.
The test case description should be written describing expectation clearly

(eg Test if the CNF crashes when disk fill occurs

should be written as

Test that the CNF does not crash when disk fill occurs)

Notes

- Tests defined here: https://github.com/cncf/cnf-testsuite/blob/2d875c66352e8dc5650c6fe7a1c43e744a7a2871/embedded_files/points.yml (10 out of the 15 'essential' test must be passed to get the certification)
- Rationale of the tests: https://github.com/cncf/cnf-testsuite/blob/main/RATIONALE.md

Issues raised to CNCF CNF Testsuite during this work

Issue	Status
#1242 - [BUG]: Test case descriptions are not clear	Open, requested to create separate issues.
[BUG]: Link for rolling-update-replication-controller is broken	Fix merged in [BUG] 1243 1244 usage doc URL and description fixes
[BUG]: Bugs in "To check if the CNF is compatible with different CNIs"	Fix merged in [BUG] 1243 1244 usage doc URL and description fixes #1245
[BUG] Test titles are ambigous	Work on a fix is ongoing
[BUG]: Some tests are not clear on service types	Open, discussion is ongoing in the issue.
[BUG]: Elastic volume is not defined	Open, discussion is ongoing in the issue.
[BUG]: Test if the CNF crashes when node drain and rescheduling occurs. All configuration should be stateless test should be separated to two cases	Open, discussion is ongoing in the issue.
[BUG]: Crashing is not defined in several test cases	Open, discussion is ongoing in the issue.
[BUG] list of mandatory PaaS components is not clarified and justified	Closed, discussion is ongoing in the issue
[BUG] Network policies are under defined	
#1321 - [Documentation]: Upgrade related terms are not explained in the the testcase descriptions	Subcase of #1242
#1322 - [Documentation]: To check if a CNF uses Kubernetes alpha APIs test case description does not define when the tescase pass	Subcase of #1242
#1337 - [Documentation]: reasonable image size test description is unclear	Subcase of #1242
#1338 - [Documentation]: Description of To check if the CNF have a reasonable startup time are not clear	Subcase of #1242
#1339 - [Documentation]: Rationale of To check if the CNF has multiple process types within one container: single_process_type is incorrect	Subcase of #1242
#1340 - [Documentation]: Rationale of To test if the CNF uses local storage is unclear	Subcase of #1242
#1341 - [Documentation]: Description of To test if the CNF uses elastic volumes is not clear	Subcase of #1242
#1409 - [BUG]: Duplicate tests about privileged containers	

The analyzis

Test	Id and Category in CNF Conformance	Note	Verdict
To test the increasing and decreasing of capacity Rationale	increase_decrease_capacity essential	Do we request horizontal scaling from all CNF-s? Most (data plane, signalling, etc) but not all (eg OSS)	should be optional, or just fail if it scales incorrectly in case the CNF scales (`ra2.app.011`)
Test if the Helm chart is published Rationale	helm_chart_published	We should first decide on CNF packaging. RA2 can stay neutral, follow the O-RAN/ONAP ASD path or propose own solution.	should be fine - no HELM specs in RA2 today, unless some incompatible CNFs packaging specs (unlikely) (`ra2.app.012`, `ra2.app.013`)
Test if the Helm chart is valid Rationale	helm_chart_valid		
Test if the Helm deploys Rationale	helm_deploy	This should be more generic, like testing if the CNF deploys.	
Test if the install script uses Helm v3 Rationale			
To test if the CNF can perform a rolling update Rationale	rolling_update	As there's some CNFs that actually use rolling update without keeping the service alive (because they require some post-configuration), the test should make sure that there is service continuity. this might just be a health probe or testing the k8s service, or something sufficiently straightforward. In other words, CNF service/traffic should work during the whole process (before during and after a rolling upgrade)	Needed (`ra2.app.014`)
To check if a CNF version can be downgraded through a rolling_version_change Rationale	rolling_version_change	It is not clear what is the difference between a rolling downgrade and a rolling version change. **A:** Defined in the external docs in the usage guide. Some these are relevant for a ReplicaSet some of them are for a Deployment. Maybe when you request an arbitrary version?	
To check if a CNF version can be downgraded through a rolling_downgrade Rationale	rolling_downgrade	Same as above?	Needed (`ra2.app.015`)
To check if a CNF version can be rolled back rollback Rationale	rollback	It is not clear what is the difference between a rolling downgrade and a rolled back rollback.	
To check if the CNF is compatible with different CNIs Rationale	cni_compatible	This covers only the default CNI, does not cover the metaplugin part. Need additional tests for cases with multiple interfaces.	Ok but needs additional tests for multiple interfaces (`ra2.app.016`)
(PoC) To check if a CNF uses Kubernetes alpha APIs Rationale	alpha_k8s_apis	Alpha API-s are not recommended by `ra2.k8s.012`. It fails with alpha PoC: it might happen that these testcases are removed from the Testsuite and this will be not part of the CNF certification. Probably will be a bonus case.	Ok (`ra2.app.017`)

To check if the CNF has a reasonable image size Rationale	reasonable_image_size	It passes if the image size is smaller than 5GB. **A:** Whenever it is possible tests are configurable or parameters can be overwritten from the outside. This will be part of the CNF Certification. Valid for each image referred from the Helm chart.	Ok but should be documented or configurable? issue to clarify name should read "pod image size" (`ra2.app.018`)
To check if the CNF have a reasonable startup time Rationale	reasonable_startup_time	It is not clear what reasonable startup time is. It is about the startup time of the microservices inside the CNF. Should be Check if all the Pods in the CNF have a reasonable startup time. **A:** Reasonable time is 60 sec.	Ok but should be documented or configurable? issue to clarify name should read "pod startup time" (`ra2.app.019`)
To check if the CNF has multiple process types within one container Rationale	single_process_type essential	Containers in the CNF should have only one process type. even for exposing an API a separate process is required - should this test if the number of processes is less than a certain number instead? Multiple process types can lead also to memory leaks. **A:** Gergely to provide examples where this requirement restricts the architecture of telco apps.	
To check if the CNF exposes any of its containers as a service Rationale	service_discovery	Service type what? RA2 mandates that clusters must support Loadbalancer and ClusterIP, and should support Nodeport and ExternalName Should there be a test for the CNF to use Ingress or Gateway objects as well?	May need tweaking to add Ingress? issue to clarify service types
To check if the CNF has multiple microservices that share a database Rationale	shared_database	Clarify rationale? In some cases it is good for multiple Microservices to share a DB, eg when restoring the state of a transaction from a failed service. Also good to have a shared DB across multiple services for things like HSS etc.	should not be required Clarify issue to clarify name
Test if the CNF crashes when node drain and rescheduling occurs. All configuration should be stateless Rationale	node_drain essential	CNF should react gracefully (no loss of context/sessions/data/logs & service continues to run) to eviction and node draining The statelessness test should be made independent & Should be skipped for stateful pods eg Dns "crashes" actually means that either the liveness or readiness probe fails - this should be made explicit and the presence of probes should be made mandatory - added issue in RA2	Needed - but replace "crash" with "react gracefully" (no loss of context /sessions/data/logs & service continues to run) issue: Statelessness test should be separate (`ra2.app.020`)
To test if the CNF uses a volume host path Rationale	volume_hostpath_not_found	should pass if the cnf doesn't have a hostPath volume What's the rationale? - **A:** When a cnf uses a volume host path or local storage it makes the application tightly coupled to the node that it is on.	ok - just fix title (`ra2.app.007`)
To test if the CNF uses local storage Rationale	no_local_volume_configuration	should fail if local storage configuration found What's the rationale? ok, add to RA2 (attach to previous)	ok - needed (`ra2.app.021`)
To test if the CNF uses elastic volumes Rationale	elastic_volumes	should pass if the cnf uses an elastic volume What's an elastic volume? Does this mean Ephemeral? Or is this an AWS-specific test? There should be a definition of what an elastic volume is (besides ELASTIC_PROVISIONING_DRIVERS_REGEX)	What's an elastic volume? Does this mean Ephemeral? Or is this an AWS-specific test? issue to clarify elastic volume

To test if the CNF uses a database with either statefulsets, elastic volumes, or both Rationale	database_persistence	A database may use statefulsets along with elastic volumes to achieve a high level of resiliency. Any database in K8s should at least use elastic volumes to achieve a minimum level of resilience regardless of whether a statefulset is used. Statefulsets without elastic volumes is not recommended, especially if it explicitly uses local storage. The least optimal storage configuration for a database managed by K8s is local storage and no statefulsets, as this is not tolerant to node failure. There should be a definition of what an elastic volume is (besides ELASTIC_PROVISIONING_DRIVERS_REGEX)	What's an elastic volume? Does this mean Ephemeral? Or is this an AWS-specific test? issue to clarify elastic volume
Test if the CNF crashes when network latency occurs Rationale	pod_network_latency	How is this tested? Where is the test running? Some traffic against a service? Latency should be configurable (default is 2s)? What should happen if latency is exceeded? Should this be more stringent than "not crashing?" What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) **A:** Explanation added to https://github.com/cncf/cnf-testsuite/blob/main/USAGE.md#heavy_check_mark-test-if-the-cnf-crashes-when-network-latency-occurs **Check this with RA2 - should be ok**	Needed but needs clarification issue on defining "crashing - it's probes (`ra2.app.028`)
Test if the CNF crashes when disk fill occurs Rationale	disk_fill	What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) **RM/RA2 should add infra monitoring recommendation for disk usage alerting**	Needed issue on defining "crashing - it's probes (`ra2.app.022`)
Test if the CNF crashes when pod delete occurs Rationale	pod_delete	What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic)	Needed issue on defining "crashing - it's probes (`ra2.app.023`)
Test if the CNF crashes when pod memory hog occurs Rationale	pod_memory_hog	What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) title should read "CNF pod runs out of memory"? RA2 should add recommendation to add pod memory reservation: > *A CNF can fail due to running out of memory. This can be mitigated by using two levels of memory policies (pod level and node level) in K8s. If the memory policies for a CNF are not fine grained enough, the CNFs out-of-memory failure blast radius will result in using all of the system memory on the node.*	Needed issue on defining "crashing - it's probes (`ra2.app.024`)
Test if the CNF crashes when pod io stress occurs Ratoinale	pod_io_stress	What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) title should read "pod disk I/O"	Needed issue on defining "crashing (`ra2.app.025`)
Test if the CNF crashes when pod network corruption occurs Rationale	pod_network_corruption	It is not clear what network corruption is in this context. What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) Rationale explains traffic manipulation: > *A higher quality CNF should be resilient to a lossy/flaky network. This test injects packet corruption on the specified CNF's container by starting a traffic control (tc) process with netem rules to add egress packet corruption.*	Needed issue on defining "crashing - it's probes (`ra2.app.026`)
Test if the CNF crashes when pod network duplication occurs Rationale	pod_network_duplication	It is not clear what network duplication is in this context. What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic)	Needed issue on defining "crashing - it's probes (`ra2.app.027`)

To test if there is a liveness entry in the Helm chart Rationale	liveness essential	Liveness probe should be mandatory, but RA2 does not mandate Helm at the moment. (it's in the pod definition rather than helm - maybe fix the title) RA2 now mandates helm3 - it's the pod definition - added issue to recommend probes in RA2 CH4	Needed (ra2.app.030)
To test if there is a readiness entry in the Helm chart Rationale	readiness essential	Readiness probe should be mandatory, but RA2 does not mandate Helm at the moment. (it's in the pod definition rather than helm - maybe fix the title) RA2 now mandates helm3 - it's the pod definition - added issue to recommend probes in RA2 CH4	Needed (ra2.app.031)
To check if logs are being sent to stdout /stderr Rationale	log_output essential	optional, as there's no way to accurately figure out if we're missing something from stdout /stderr title reads "instead of a log file" **A:** RA2 should recommend that the application streams logs out of stdout/stderr	
To check if prometheus is installed and configured for the cnf Rationale	prometheus_traffic	There is a chapter for Additional required components (4.10), but without any content. should ra2 mandate prometheus? **A:** All the PaaS components are optionally tested, as bonus tests. RM/RA right now doesn't require specific PaaS tools	Not needed question on mandatory paas tools
To check if logs and data are being routed through an *Unified Logging Layer* Rationale	**routed_logs**	There is a chapter for Additional required components (4.10), but without any content. should ra2 mandate fluent? **A:** All the PaaS components are optionally tested, as bonus tests.	Not needed question on mandatory paas tools
To check if Open Metrics is being used and or compatible. Rationale	open_metrics	There is a chapter for Additional required components (4.10), but without any content. should ra2 mandate open metrics? **A:** All the PaaS components are optionally tested, as bonus tests.	Not needed question on mandatory paas tools
To check if tracing is being used with Jaeger Rationale	tracing	There is a chapter for Additional required components (4.10), but without any content. should ra2 mandate jaeger? **A:** All the PaaS components are optionally tested, as bonus tests.	Not needed question on mandatory paas tools
To check if a CNF is using container socket mounts	container_sock_mounts essential	what is being tested? Make sure to not mount /var/run/docker.sock, /var/run/containerd.sock or /var/run/crio.sock on the containers?	Needed (ra2.app.032)
To check if containers are using any tiller images		ie test if it's NOT helm v2?	ok if not helm v2
To check if any containers are running in privileged mode Rationale	privileged_containers essential	ie NOT privileged?	Needed issue to clarify name (ra2.app.033)
To check if a CNF is running services with external IP's	external_ips	does this mean "k8s service?" RA2 mandates that clusters must support Loadbalancer and ClusterIP, and should support Nodeport and ExternalName	issue to clarify name issue to clarify service types
To check if any containers are running as a root user Rationale	non_root_user	ie not Root?	Needed issue to clarify name (ra2.app.034)
To check if any containers allow for privilege escalation Rationale	privilege_escalation	ie not allowed?	Needed issue to clarify name (ra2.app.035)

To check if an attacker can use a symlink for arbitrary host file system access Rationale	symlink_file_system	ok if not According to the CVE this is not valid anymore in Kubernetes 1.23.	Not needed issue to clarify name
To check if there are service accounts that are automatically mapped Rationale	application_credentials	what is the expectation? **Application Credentials:** Developers store secrets in the Kubernetes configuration files, such as environment variables in the pod configuration. Such behavior is commonly seen in clusters that are monitored by Azure Security Center. Attackers who have access to those configurations, by querying the API server or by accessing those files on the developer's endpoint, can steal the stored secrets and use them. Check if the pod has sensitive information in environment variables, by using list of known sensitive key names. Check if there are configmaps with sensitive information. **Remediation:** Use Kubernetes secrets or Key Management Systems to store credentials. See more at ARMO-C0012	Needed issue to clarify name (`ra2.app.036`)
To check if there is a host network attached to a pod Rationale	host_network	should be ok with or without - eg when exposing services via cluster network as opposed to nodeport?	Needed (`ra2.app.037`)
To check if there are service accounts that are automatically mapped Rationale		Disable automatic mounting of service account tokens to PODs either at the service account level or at the individual POD level, by specifying the automountServiceAccountToken: false. Note that POD level takes precedence. See more at ARMO-C0034	Seems to be a duplicate.
To check if there is an ingress and egress policy defined Rationale	ingress_egress_blocked	ok - maybe more stringent? **A:** There is an answer here: https://github.com/cncf/cnf-testsuite/issues/1282#issuecomment-1081228008 **Check this with RA2**	issue to have more stringent network policies (only allow predefined subnets ie not 0/0 for ingress, only allow limited number of protocols/ports)
To check if there are any privileged containers Rationale		duplicate?	#1409 - [BUG]: Duplicate tests about privileged containers
To check for insecure capabilities Rationale	insecure_capabilities	what is the expectation?	issue to clarify name
To check for dangerous capabilities Rationale		what is the expectation?	issue to clarify name
To check if namespaces have network policies defined Rationale		ok - maybe more stringent? duplicate?	issue to have more stringent network policies
To check if containers are running with non-root user with non-root membership Rationale	non_root_containers essential	duplicate?	ok (`ra2.app.038`)

To check if containers are running with hostPID or hostIPC privileges Rationale	host_pid_ipc_privileges	ok if not	ok if not (`ra2.app.039`)
To check if security services are being used to harden containers Rationale	linux_hardening	what services? should be configurable or optional **Linux Hardening:** Check if there is AppArmor, Seccomp, SELinux or Capabilities are defined in the securityContext of container and pod. If none of these fields are defined for both the container and pod, alert. **Remediation:** In order to reduce the attack surface, it is recommended to harden your application using security services such as SELinux®, AppArmor®, and seccomp. Starting from Kubernetes version 1.22, SELinux is enabled by default, therefore I do not think that we need to require anything in RA2. Read more at ARMO-C0055	not needed
To check if containers have resource limits defined Rationale	resource_policies essential	ok	ok (`ra2.app.040`)
To check if containers have immutable file systems Rationale	immutable_file_systems	ok	ok (`ra2.app.041`)
To check if containers have hostPath mounts Rationale	hostpath_mounts essential	ok if not	ok, issue to clarify name (`ra2.app.042`)
To check if containers are using labels	require_labels	ok - maybe mandate some mandatory labels?	ok (`ra2.app.043`)
To test if there are versioned tags on all images using OPA Gatekeeper Rationale	versioned_tag	ok	ok (`ra2.app.044`)
To test if there are any (non-declarative) hardcoded IP addresses or subnet masks Rationale	ip_addresses	ok - there shouldn't be any internal hardcoded nw anyway This was replaced by hardcoded_ip_addresses_in_k8s_runtime_configuration	ok (`ra2.app.045`)
To test if there are node ports used in the service configuration Rationale	nodeport_not_used	ok but service type LB should be better	ok, issue to clarify service types (`ra2.app.046`)
To test if there are host ports used in the service configuration Rationale	hostport_not_used essential	hostports should not be used	

To test if there are any (non-declarative) hardcoded IP addresses or subnet masks in the K8s runtime configuration Rationale	hardcoded_ip_addresses_in_k8s_runtime_configuration essential	Not a duplicate anymore	
To check if a CNF version uses immutable configmaps Rationale	immutable_configmap	ok	ok (ra2.app.047)
Test if the CNF crashes when pod dns error occurs	pod_dns_error	What is the expectation? (not crashing = not exit with error code or (better) not stopping to process traffic) Not crashing = answering to probes	ok (ra2.app.028)
To check if a CNF uses K8s secrets Rationale	**secrets_used**		
To check if any pods in the CNF use sysctls with restricted values Rationale	**sysctls**		New
	helm_tiller		New There is no rationale for this
To check if selinux has been configured properly Rationale	**selinux_options** essential	If SELinux options is configured improperly it can be used to escalate privileges and should not be allowed. Not applicable if SELinux is not installed, but if SELinux is installed a proper configuration is needed.	
To check if a CNF is using the default namespace Rationale	default_namespace		New
To test if mutable tags being used for image versioning (Using Kyverno) Rationale	**latest_tag** essential	*"You should avoid using the :latest tag when deploying containers in production as it is harder to track which version of the image is running and more difficult to roll back properly."*	

Derived RA2 requirements

Ref	Specification	Details	Requirement Trace	Reference Implementation Trace
ra2.app.011	Horizontal scaling	Increasing and decreasing of the CNF capacity must be implemented using horizontal scaling. If horizontal scaling is supported automatic scaling must be possible using Kubernetes Horizontal Pod Autoscale (HPA) feature.	CNCF CNF Testsuite	
ra2.app.012	Published helm chart	Helm charts of the CNF must be published into a helm registry and must not be used from local copies.	CNCF CNF Testsuite	
ra2.app.013	Valid Helm chart	Helm charts of the CNF must be valid and should pass the `helm lint` validation.	CNCF CNF Testsuite	
ra2.app.014	Rolling update	The CNF must be able to perform a rolling update using Kubernetes deployments.	CNCF CNF Testsuite	

`ra2.app.015`	Rolling downgrade	The CNF must be able to perform a rolling downgrade using Kubernetes deployments.	CNCF CNF Testsuite	
`ra2.app.016`	CNI compatibility	The CNF must use CNI compatible networking plugins.	CNCF CNF Testsuite	
`ra2.app.017`	API stability	The CNF must not use any Kubernetes alpha API-s.	CNCF CNF Testsuite	
`ra2.app.018`	CNF image size	The different container images of the CNF should not be bigger than 5GB.	CNCF CNF Testsuite	
`ra2.app.019`	CNF startup time	Startup time of the Pods of a CNF should not be more than 60s where startup time is the time between starting the Pod until the readiness probe outcome is Success.	CNCF CNF Testsuite	
`ra2.app.020`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of a node drain and rescheduling occurs.	CNCF CNF Testsuite	
`ra2.app.021`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of network latency occurs	CNCF CNF Testsuite	
`ra2.app.022`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of disk fill occurs.	CNCF CNF Testsuite	
`ra2.app.023`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of pod delete occurs.	CNCF CNF Testsuite	
`ra2.app.024`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of pod memory hog occurs.	CNCF CNF Testsuite	
`ra2.app.025`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of pod I/O stress occurs.	CNCF CNF Testsuite	
`ra2.app.026`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of pod network corruption occurs.	CNCF CNF Testsuite	
`ra2.app.027`	CNF resiliency	CNF must not loose data, must continue to run and its readiness probe outcome must be Success even in case of pod network duplication occurs.	CNCF CNF Testsuite	
`ra2.app.028`	CNF resiliency	CNF must not loose data, shmust ould continue to run and its readiness probe outcome must be Success even in case of pod DNS error occurs.		
`ra2.app.029`	CNF local storage	CNF must not use local storage.	CNCF CNF Testsuite	
`ra2.app.030`	Liveness probe	The CNF must have `livenessProbe` defined.	CNCF CNF Testsuite	
`ra2.app.031`	Readiness probe	The CNF must have `readinessProbe` defined.	CNCF CNF Testsuite	
`ra2.app.032`	No access to container daemon sockets	The CNF must not have any of the container daemon sockets (e.g.: /var/run/docker.sock, /var/run/containerd.sock or /var/run/crio.sock) mounted.	CNCF CNF Testsuite	
`ra2.app.033`	No privileged mode	None of the Pods of the CNF should run in privileged mode.	CNCF CNF Testsuite	
`ra2.app.034`	No root user	None of the Pods of the CNF should run as a root user.	CNCF CNF Testsuite	
`ra2.app.035`	No privilege escalation	None of the containers of the CNF should allow privilege escalation.	CNCF CNF Testsuite	
`ra2.app.036`	No automatic service account mapping	Non specified service accounts must not be automatically mapped. To prevent this *automountServiceAccountToken: false* *flag* must be set in all Pods of the CNF.	CNCF CNF Testsuite	

ra2.app.037	No host network access	Host network must not be attached to any of the Pods of the CNF. hostNetwork attribute of the Pod specifications must be False or should not be specified.	CNCF CNF Testsuite	
ra2.app.038	Non-root user	All Pods of the CNF should be able to execute with a non-root user having a non-root group. Both runAsUser and runAsGroup attributes should be set to a greater value than 999.	CNCF CNF Testsuite	
ra2.app.039	Host process namespace separation	Pods of the CNF must not share the host process ID namespace or the host IPC namespace. Pod manifests must not have the hostPID or the hostIPC attribute set to true.	CNCF CNF Testsuite	
ra2.app.040	Resource limits	All containers and namespaces of the CNF must have defined resource limits for at least CPU and memory resources.	CNCF CNF Testsuite	
ra2.app.041	Read only filesystem	It is recommended that the containers of the CNF have read only filesystem. The readOnlyRootFilesystem attribute of the Pods in the their securityContext should be set to true.	CNCF CNF Testsuite	
ra2.app.042	No host path mounts	Pods of the CNF must not use hostPath mounts.	Kubernetes documentation	
ra2.app.043	labels	Pods of the CNF should define at least the following labels: app.kubernetes.io/name, app.kubernetes.io/version and app.kubernetes.io/part-of	Kubernetes documentation	
ra2.app.044	Container image tags	All referred container images in the Pod manifests must be referred by a version tag pointing to a concrete version of the image. latest tag must not be used.		
ra2.app.045	No hardcoded IP addresses	The CNF must not have any hardcoded IP addresses in its Pod specifications.	CNCF CNF Testsuite	
ra2.app.046	No node ports	Service declarations of the CNF must not contain nodePort definition.	Kubernetes documentation	
ra2.app.047	Immutable config maps	ConfigMaps used by the CNF must be immutable.	Kubernetes documentation	