Протокол № 24-007 проведения совместных испытаний “Deckhouse Kubernetes Platform” версии 1.63.11, “Deckhouse Virtualization Platform” версии 0.13.2, модуля csi-yadro версии 0.1.0 и системы хранения данных TATLIN.UNIFIED GEN2 версии 3.1.1.

| г. Москва | 25.10.2024 |
| --- | --- |

## Предмет испытаний

В настоящем протоколе зафиксирован факт проведения в период с 01.08.2024 по 25.10.2024 совместных испытаний программного обеспечения:

* «Deckhouse Kubernetes Platform» редакции ЕЕ версии 1.63.11 (далее – Платформа контейнеризации), модуль csi-yadro версии 0.1.0, разработанных АО «Флант», и системы хранения данных TATLIN.UNIFIED GEN2 версии 3.1.1 (далее - Система хранения данных), разработанной ООО «КНС Групп».
* “Deckhouse Virtualization Platform” версии 0.13.2 (далее – Платформа виртуализации), разработанного АО «Флант», и системы хранения данных TATLIN.UNIFIED GEN2 версии 3.1.1 (далее - Система хранения данных), разработанной ООО «КНС Групп».

## Объект испытаний

Платформа виртуализации входит (как модуль) в состав Платформы контейнеризации, поэтому для неё отдельный инсталлятор не требуется.

Перечень компонентов, эксплуатирующихся в ходе проведения данных испытаний, относящихся к Платформе контейнеризации, приведен в Таблице 1.

*Таблица 1 - Перечень компонентов, относящихся к Платформе контейнеризации*

| Описание | Наименование | Источник |
| --- | --- | --- |
| Файл программного пакета дистрибутива Платформы контейнеризации | Deckhouse Kubernetes Platform редакции EE v1.63.11 | Источник в сети “Интернет”, адрес:https://registry.deckhouse.ru/deckhouse/ee/install:stable |
| Официальное руководство по эксплуатации Платформы контейнеризации в электронном формате | Deckhouse Kubernetes Platform | https://deckhouse.ru/documentation/v1/deckhouse-overview.html |
| Документация на модуль | Документация на модуль csi-yadro v0.1.0 | <https://deckhouse.ru/products/kubernetes-platform/modules/csi-yadro/stable/>  |
| Официальное руководство по эксплуатации Платформы виртуализации в электронном формате | Deckhouse Virtualization Platform | https://deckhouse.ru/products/kubernetes-platform/modules/virtualization/stable/ |

## Ход испытаний

1. В ходе выполнения настоящих испытаний были выполнены проверки корректности работы Платформы контейнеризации и Платформы виртуализации во взаимодействии с Системой хранения данных в объеме, указанном в Приложении 1.
2. Перечень официальных репозиториев ПО, которые эксплуатировались в ходе испытаний:
	1. Система хранения данных TATLIN.UNIFIED GEN2 3.1.1.
	2. Платформа контейнеризации“Deckhouse Kubernetes Platform” редакции ЕЕ 1.63.11
	3. Модуль csi-yadro версии 0.1.0
	4. Платформа виртуализации “Deckhouse Virtualization Platform” версии 0.13.2
3. Неофициальные репозитории не использовались в рамках испытаний.
4. Испытания включали в себя сценарии, использующие интеграцию с API системы хранения данных.

## Результат испытаний

В ходе функциональных проверок подтверждена корректная работа с API Системы хранения данных.

Исключения:

* Изменение размера диска работает не в полном объёме, совместимость не подтверждена.

## Вывод

1. ПО и Система хранения данных совместимы, учитывая “Ход испытаний”, раздел “Результат испытаний”, Приложение 2, Приложение 3.

## Состав рабочей группы и подписи сторон

Данный протокол составлен участниками рабочей группы:

* Со стороны ПО (АО “Флант”):
	+ Ведущий инженер по информационной безопасности и архитектурным решениям Зимин А.В.
	+ Инженер архитектурных решений Демчук Н.В.
	+ Инженер архитектурных решений Олейников В.М.

Директор по работе с технологическими партнёрами АО “Флант”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Юдин Д.А.

## Приложение 1 к Протоколу № 24-007

**Перечень проверок совместимости Платформы контейнеризации и Системы хранения данных**

*Таблица 2 - Список проверок*

| **№ п/п** | **Наименование проверки** | **Результат проверки** |
| --- | --- | --- |
| 1 | Создание тома в файловом режиме | Успешно |
| 2 | Изменение размера тома в файловом режиме | Успешно |
| 3 | Удаление тома | Успешно |
| 4 | rwx в блочном режиме  | Успешно |
| 5 | Запуск e2e тестов от Kubernetes | Успешно |
|  | Запуск e2e тестов от Storage | Успешно |

**Перечень проверок совместимости Платформы контейнеризации и Системы хранения данных**

*Таблица 3 - Список проверок*

| **№ п/п** | **Наименование проверки** | **Результат проверки** |
| --- | --- | --- |
| 1 | Включение модуля виртуализации, создание ВМ | Успешно |
| 2 | Изменение размера тома | Успешно |
| 3 | live миграция виртуальных машин | Успешно |
| 4 | Удаление ВМ и дисков | Успешно |
| 5 | Запуск e2e тестов от Storage | Успешно |

## Приложение 2 к Протоколу № 24-007

**Инструкция по выполнению проверки совместимости Платформы контейнеризации и Системы хранения данных**

*Таблица 4 - Инструкции по проверке совместимости*

| **№ п/п** | **Описание требования** | **Критерий оценки успешности проверки** | **Примечание** |
| --- | --- | --- | --- |
| **1. Создание тома в файловом режиме** |
| 1.1 | Создать Pod + PVC в файловом режиме | Успешно создан Pod и связанная с ним PVC | # Создайте storage class и все необходимые переменные окруженияexport NAMESPACE=<namespace name for tests>export STORAGE\_CLASS\_NAME=<storage class name for tests>export POD\_NAME\_PREFIX=<pod name prefix for tests>export PVC\_NAME\_PREFIX=<pvc name prefix for tests>export VOLUME\_SNAPSHOT\_NAME\_PREFIX=<volume snapshot name prefix for tests>export VOLUME\_SNAPSHOT\_CLASS\_NAME=<volume snapshot class name for tests>cat > sc.yaml <EOFallowVolumeExpansion: trueapiVersion: storage.k8s.io/v1kind: StorageClassmetadata: annotations: storageclass.kubernetes.io/is-default-class: "true" name: ${STORAGE\_CLASS\_NAME}parameters: poolName: pool-ssd protocol: iscsi storageID: TEST01 volumeExportPort: p50,p51,p60,p61provisioner: csi-tatlinunified.yadro.comreclaimPolicy: DeletevolumeBindingMode: ImmediateEOFkubectl apply -f sc.yaml# Проверьте результатkubectl get sc ${STORAGE\_CLASS\_NAME}echo NAMESPACE=${NAMESPACE}echo STORAGE\_CLASS\_NAME=${STORAGE\_CLASS\_NAME}echo POD\_NAME\_PREFIX=${POD\_NAME\_PREFIX}echo PVC\_NAME\_PREFIX=${PVC\_NAME\_PREFIX}echo VOLUME\_SNAPSHOT\_NAME\_PREFIX=${VOLUME\_SNAPSHOT\_CLASS\_NAME}echo VOLUME\_SNAPSHOT\_CLASS\_NAME=${VOLUME\_SNAPSHOT\_CLASS\_NAME}""cat > pvc-fs.yaml <<EOFkind: PersistentVolumeClaimapiVersion: v1metadata: name: ${PVC\_NAME\_PREFIX}-fsspec: accessModes: - ReadWriteOnce resources: requests: storage: 1Gi storageClassName: ${STORAGE\_CLASS\_NAME} volumeMode: FilesystemEOFcat > pod-fs.yaml <<EOFkind: PodapiVersion: v1metadata: name: ${POD\_NAME\_PREFIX}-fsspec: containers: - name: test-nginx image: nginx volumeMounts: - name: test-storage mountPath: /data  restartPolicy: "Never" volumes: - name: test-storage persistentVolumeClaim: claimName: ${PVC\_NAME\_PREFIX}-fsEOF# Создайте ресурсы, описанные вышеkubectl -n $NAMESPACE apply -f pvc-fs.yamlkubectl -n $NAMESPACE apply -f pod-fs.yaml# Проверьте результат. Статус PVC должен быть Bound.Статус пода должен быть Running. kubectl -n $NAMESPACE get pvc ${PVC\_NAME\_PREFIX}-fskubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs -owide -w  |
| 1.2 | Создать файл размером 100МБ в созданном Pod | Файл создан успешно | kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs -- dd if=/dev/random of=/data/test-file bs=1M count=1000 |
| 1.3 | Получить sha256 созданного ранее файла | sha256 получен | export SHA256\_SUM\_OLD=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs -- sha256sum /data/test-file | cut -d ' ' -f 1)# Проверьте результат. В переменной должен быть сохранен хеш файлаecho SHA256\_SUM\_OLD=${SHA256\_SUM\_OLD} |
| 1.4 | Переместить Pod на другой узел | Pod успешно перемещён на другой узел | export NODE\_NAME=$(kubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs -o json | jq '.spec.nodeName' | tr -d '"')kubectl cordon $NODE\_NAMEkubectl -n $NAMESPACE delete po ${POD\_NAME\_PREFIX}-fskubectl -n $NAMESPACE apply -f pod-fs.yamlkubectl uncordon $NODE\_NAME# Проверьте, что под успешно создан на другом узле.export NEW\_NODE\_NAME=$(kubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs -o json | jq -r '.spec.nodeName')echo PREVIOUS\_NODE\_NAME=${NODE\_NAME}echo NEW\_NODE\_NAME=${NEW\_NODE\_NAME}# Проверьте, что статус пода Runningkubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs -owide -w |
| 1.5 | Проверить sha256 | sha256 получен, значение совпадает с шагом 1.4 | export SHA256\_SUM\_NEW=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_SUM\_OLD=${SHA256\_SUM\_OLD}echo SHA256\_SUM\_NEW=${SHA256\_SUM\_NEW} |
| **2. Изменение размера тома в файловом режиме** |
| *Предварительное условие: Должен быть файл (можно использовать Pod из 1. Создание тома в файловом режиме)* |
| 2.1 | Увеличить размер PVC на 1Gi (в параметре spec pvc) | Размер pvc изменён успешно, проверить итоговое значение нужно в параметре status.capacity.storage | export NEW\_SIZE="50Gi"kubectl -n $NAMESPACE patch pvc ${PVC\_NAME\_PREFIX}-fs --type='json' -p="[{\"op\": \"replace\", \"path\": \"/spec/resources/requests/storage\", \"value\": \""$NEW\_SIZE\"}]"# необходимо подождать примерно 1-2 минуты, пока завершится resizekubectl -n $NAMESPACE get pvc ${PVC\_NAME\_PREFIX}-fs -ojson -w | jq '.status.capacity' |
| 2.2 | Проверить размер файловой системы | Успешно изменился размер файловой системы. Размер должен быть равен величине, указанной в пункте 2.1 | kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs -- df -h | grep /data | awk '{print $2}' |
| 2.3 | Убедиться, что файл (размер) и sha256 файла остались неизменны | Размер файла и sha256 не изменились | export SHA256\_SUM\_AFTER\_RESIZE=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_SUM\_NEW=${SHA256\_SUM\_NEW}echo SHA256\_SUM\_AFTER\_RESIZE=${SHA256\_SUM\_AFTER\_RESIZE} |
| **3. Удаление тома** |
| 3.1 | Получить имя PV, которая была созданная ранее в тестах | Успешно получено имя PV | export PV\_NAME=$(kubectl -n ${NAMESPACE} get pvc ${PVC\_NAME\_PREFIX}-fs -ojson | jq -r '.spec.volumeName')echo $PV\_NAME |
| 3.2 | Найти соответствующий тому ресурс в СХД/SDS | На СХД найден ресурс, имя которого равно имени PV, полученного в пункте 3.1 | Зайти на веб-интерфейс СХД - "логические объекты" - "ресурсы".(имя ресурса на СХД = имя pv)Найти по имени pv ресурс, проверить текущий размер пула, в котором он создан |
| 3.3 | Удалить Pod | Pod удалён | kubectl -n $NAMESPACE delete pod ${POD\_NAME\_PREFIX}-fs |
| 3.4 | Удалить PVC | PVC удалена, удалена так же связанная с ним PV | kubectl -n $NAMESPACE delete pvc ${PVC\_NAME\_PREFIX}-fspatch kubectl get pv ${PV\_NAME} |
| 3.5 | На СХД/SDS должен удалиться ресурс и должно освободиться место | Ресурс удалён, место освобождено | Зайти на веб-интерфейс СХД - "логические объекты" - "ресурсы"(имя ресурса на СХД = имя pv)Проверить, что ресурс удалился, а количество свободного места в пуле увеличилось |
| **4. rwx в блочном режиме** |
| 4.1 | Создать Pod + PVC в блочном режиме | Успешно создан Pod и связанная с ним PVC | # Ресурсы, для тестов блочного режима:cat > pvc-block-rwx.yaml <<EOFapiVersion: v1kind: PersistentVolumeClaimmetadata: name: ${PVC\_NAME\_PREFIX}-block-rwxspec: accessModes: - ReadWriteMany  resources: requests: storage: 4Gi storageClassName: ${STORAGE\_CLASS\_NAME} volumeMode: BlockEOFcat > deployment-block-rwx.yaml <<EOFapiVersion: apps/v1kind: Deploymentmetadata: name: ${POD\_NAME\_PREFIX}-block-rwx labels: app: ${POD\_NAME\_PREFIX}-block-rwxspec: replicas: 2 selector: matchLabels: app: ${POD\_NAME\_PREFIX}-block-rwx  template: metadata: labels: app: ${POD\_NAME\_PREFIX}-block-rwx spec: affinity: podAntiAffinity: requiredDuringSchedulingIgnoredDuringExecution: - labelSelector: matchLabels: app: ${POD\_NAME\_PREFIX}-block-rwx topologyKey: "kubernetes.io/hostname" containers: - name: test-nginx image: nginx volumeDevices: - name: test-storage devicePath: /dev/my-block-device  volumes: - name: test-storage persistentVolumeClaim: claimName: ${PVC\_NAME\_PREFIX}-block-rwxEOF# Создаём ресурсы:kubectl -n $NAMESPACE apply -f pvc-block-rwx.yamlkubectl -n $NAMESPACE apply -f deployment-block-rwx.yaml# Проверяем, что PVC в статусе Bound.kubectl -n $NAMESPACE get pvc ${PVC\_NAME\_PREFIX}-block-rwx# Проверяем, что поды создались на разных узлах и их статус Runningkubectl -n $NAMESPACE get pod -w -owide -l app=${POD\_NAME\_PREFIX}-block-rwx |
| 4.2 | Создать Pod + PVC в блочном режиме на другом узле | Успешно создан Pod и связанная с ним PVC на другом узле | # Проверить, что том присутствует на обоих подах. В выводе команды должны быть имена 2х подовexport POD\_NAMES=$(kubectl -n $NAMESPACE get pods -l app=${POD\_NAME\_PREFIX}-block-rwx -o json | jq -r '.items[] | select(.spec.volumes = "test-storage") | .metadata.name')echo POD\_NAMES=${POD\_NAMES}# Проверить, что блочное устройство смонтировано на обоих подах и его размер соответствует заданномуfor pod in ${POD\_NAMES}; do export BLOCK\_DEVICE\_SIZE=$(kubectl -n $NAMESPACE exec -it ${pod} -- blockdev --getsize64 /dev/my-block-device | awk '{print $1/1073741824 " GB"}') echo pod="${pod}" size="${BLOCK\_DEVICE\_SIZE}"done |
| **5. Создание снэпшотов** |
| **5.1** | Подготовка окружения | Включен snapshot-controllerУспешно создан Pod и связанная с ним PVC | # Включить модуль snapshot controllercat > snapshot-controller-module-config.yaml <<EOFapiVersion: deckhouse.io/v1alpha1kind: ModuleConfigmetadata: name: snapshot-controllerspec: enabled: true version: 1EOFkubectl apply -f snapshot-controller-module-config.yaml# Проверить, что модуль включенkubectl get mc snapshot-controllerkubectl get module snapshot-controller -w |
| **5.2** | Создать Pod + PVC в файловом режиме | Успешно создан Pod и связанная с ним PVC | cat > pvc-fs-for-snapshot.yaml <<EOFkind: PersistentVolumeClaimapiVersion: v1metadata: name: ${PVC\_NAME\_PREFIX}-fs-for-snapshotspec: accessModes: - ReadWriteOnce resources: requests: storage: 2Gi storageClassName: ${STORAGE\_CLASS\_NAME} volumeMode: FilesystemEOFcat > pod-fs-for-snapshot.yaml <<EOFkind: PodapiVersion: v1metadata: name: ${POD\_NAME\_PREFIX}-fs-for-snapshotspec: containers: - name: test-nginx image: nginx volumeMounts: - name: test-storage mountPath: /data  restartPolicy: "Never" volumes: - name: test-storage persistentVolumeClaim: claimName: ${PVC\_NAME\_PREFIX}-fs-for-snapshotEOF# Создайте ресурсы, описанные вышеkubectl -n $NAMESPACE apply -f pvc-fs-for-snapshot.yamlkubectl -n $NAMESPACE apply -f pod-fs-for-snapshot.yaml# Проверьте результат. Статус PVC должен быть Bound.Статус пода должен быть Running. kubectl -n $NAMESPACE get pvc ${PVC\_NAME\_PREFIX}-fs-for-snapshotkubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs-for-snapshot -owide -w |
| **5.3** | Убедиться, что ресурсы созданы в хранилище, записать тестовые данные на том | Тома присутстуют в хранилищеЗаписаны тестовые данныеСнята их sha256-контрольная сумма | # Получить имя PVexport PV\_NAME=$(kubectl -n ${NAMESPACE} get pvc ${PVC\_NAME\_PREFIX}-fs-for-snapshot -ojson | jq -r '.spec.volumeName')echo PV\_NAME=$PV\_NAME# Зайти на используемое хранилище - веб-интерфейс Tatlinunified, nfs-сервер, ceph-пул - и проверить, что том созданной нами pvc существует# Записать на том тестовые данные kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot -- dd if=/dev/random of=/data/test-file bs=1M count=1000# Подсчитать контрольную суммуexport SHA256\_ORIGINAL=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_ORIGINAL=${SHA256\_ORIGINAL} |
| **5.4** | Создать снэпшот | Созданы volumesnapshot и volumesnapshotclass | # Создаём snapshotcat > volume-snapshot-1.yaml <<EOFapiVersion: snapshot.storage.k8s.io/v1kind: VolumeSnapshotmetadata: name: ${VLOLUME\_SNAPSHOT\_NAME\_PREFIX}-1spec: volumeSnapshotClassName: ${VOLUME\_SNAPSHOT\_CLASS\_NAME} source: persistentVolumeClaimName: ${PVC\_NAME\_PREFIX}-fs-for-snapshotEOFkubectl -n ${NAMESPACE} apply -f volume-snapshot-1.yaml# Проверяем, что volume shapshot создался и его READYTOUSE перешел в truekubectl -n ${NAMESPACE} get volumesnapshot ${VLOLUME\_SNAPSHOT\_NAME\_PREFIX}-1# Проверяем наличие volume snapshot content. Его READYTOUSE тоже должен быть trueexport VLOLUME\_SNAPSHOT\_CONTENT\_NAME=$(kubectl -n ${NAMESPACE} get volumesnapshot ${VLOLUME\_SNAPSHOT\_NAME\_PREFIX}-1 -o json | jq -r '.status.boundVolumeSnapshotContentName')kubectl get volumesnapshotcontent ${VLOLUME\_SNAPSHOT\_CONTENT\_NAME}# Проверяем наличие снэпшота на СХД (по имени volume snapshot content) |
| **5.5** | Восстановить ресурсы из снэпшота | Из снэпшота успешно создан новый подНа томе остались записанные ранее данные | # Создаём новый под с томом, восстановленным из снэпшотаcat > pvc-fs-for-snapshot-restored.yaml <<EOFapiVersion: v1kind: PersistentVolumeClaimmetadata: name: ${PVC\_NAME\_PREFIX}-fs-for-snapshot-restoredspec: accessModes: - ReadWriteOnce resources: requests: storage: 2Gi storageClassName: ${STORAGE\_CLASS\_NAME} dataSource: name: ${VLOLUME\_SNAPSHOT\_NAME\_PREFIX}-1 kind: VolumeSnapshot apiGroup: snapshot.storage.k8s.ioEOFcat > pod-fs-for-snapshot-restored.yaml <<EOFkind: PodapiVersion: v1metadata: name: ${POD\_NAME\_PREFIX}-fs-for-snapshot-restoredspec: containers: - name: test-nginx image: nginx volumeMounts: - name: test-storage mountPath: /data  restartPolicy: ""Never"" volumes: - name: test-storage persistentVolumeClaim: claimName: ${PVC\_NAME\_PREFIX}-fs-for-snapshot-restoredEOFkubectl -n ${NAMESPACE} apply -f pvc-fs-for-snapshot-restored.yamlkubectl -n ${NAMESPACE} apply -f pod-fs-for-snapshot-restored.yaml# Проверяем, что ресурсы созданы и под запущенkubectl -n ${NAMESPACE} get pvc ${PVC\_NAME\_PREFIX}-fs-for-snapshot-restoredkubectl -n ${NAMESPACE} get pod ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -w |
| **5.6** | Проверить контрольную сумму данных, востановленных из снэпшота | Контрольная сумма не поменялась | # Сверяем контрольную сумму данных в томе, восстановленном из снэпшотаexport SHA256\_RESTORED=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_ORIGINAL=${SHA256\_ORIGINAL}echo SHA256\_RESTORED=${SHA256\_RESTORED} |
| **5.7** | Изменить данные в оригинале (старом поде) | В новом поде данные не изменились | # Изменяем данные в старом подеkubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot -- bash -c 'echo "new\_data" >> /data/test-file'# Получаем контрольную сумму измененных данных и сверяем ееexport SHA256\_ORIGINAL\_NEW=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_ORIGINAL=${SHA256\_ORIGINAL}echo SHA256\_RESTORED=${SHA256\_RESTORED}echo SHA256\_ORIGINAL\_NEW=${SHA256\_ORIGINAL\_NEW} |
| **5.8** | Изменить данные в восстановленном томе (новом поде) | Данные в новом поде успешно изменены | # Изменяем данные в новом подеkubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -- bash -c 'echo ""new\_data\_for\_restored"" >> /data/test-file'# Получаем контрольную сумму измененных данных и сверяем ееexport SHA256\_RESTORED\_NEW=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_ORIGINAL=${SHA256\_ORIGINAL}echo SHA256\_RESTORED=${SHA256\_RESTORED}echo SHA256\_ORIGINAL\_NEW=${SHA256\_ORIGINAL\_NEW}echo SHA256\_RESTORED\_NEW=${SHA256\_RESTORED\_NEW} |
| **5.9** | Удалить снэпшот | Снэпшот должен быть удален на СХД. Новый под должен продолжить работать даже после перезапуска пода. | kubectl -n $NAMESPACE delete -f volume-snapshot-1.yaml# Проверить удаление снэпшота на СХД# Проверить, что новый под после перезапуска не потерял данные.kubectl -n $NAMESPACE delete -f pod-fs-for-snapshot-restored.yamlwatch kubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs-for-snapshot-restoredkubectl -n $NAMESPACE apply -f pod-fs-for-snapshot-restored.yamlkubectl -n $NAMESPACE get po ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -wexport SHA256\_RESTORED\_NEW\_AFTER\_POD\_RECREATE=$(kubectl -n $NAMESPACE exec -it ${POD\_NAME\_PREFIX}-fs-for-snapshot-restored -- sha256sum /data/test-file | cut -d ' ' -f 1)echo SHA256\_ORIGINAL=${SHA256\_ORIGINAL}echo SHA256\_RESTORED=${SHA256\_RESTORED}echo SHA256\_ORIGINAL\_NEW=${SHA256\_ORIGINAL\_NEW}echo SHA256\_RESTORED\_NEW=${SHA256\_RESTORED\_NEW}echo SHA256\_RESTORED\_NEW\_AFTER\_POD\_RECREATE=${SHA256\_RESTORED\_NEW\_AFTER\_POD\_RECREATE} |
| **6. Запуск e2e тестов от Kubernetes** |
| 5.1 | Запустить e2e тесты | e2e тесты отработали, ошибок нет | Проверяют возможности csi-драйвераНастройки:StorageClass: FromExistingClassName: ${STORAGE\_CLASS\_NAME}DriverInfo: Name: csi-tatlinunified.yadro.com InTreePluginName: RequiredAccessModes: - ReadWriteOnce Capabilities: persistence: true block: true fsGroup: true multipods: false exec: true snapshotDataSource: true controllerExpansion: true nodeExpansion: true offlineExpansion: true onlineExpansion: true singleNodeVolume: false topology: true volumeLimits: false StressTestOptions: NumPods: 20 NumRestarts: 20 PerformanceTestOptions: ProvisioningOptions: VolumeSize: 10Gi Count: 30 ExpectedMetrics: AvgLatency: 25000000000 # 25s Throughput: 0.9 |
| **7. Запуск e2e тестов от Storage** |
| 6.1 | Запустить e2e тесты | e2e тесты отработали, ошибок нет | Тесты в цикле:- Создают ВМ + диски- Изменяют размер виртуального диска- Удаляют ВМ + диски |

## Приложение 3 к Протоколу № 24-007

**Инструкция по выполнению проверки совместимости Платформы виртуализации и Системы хранения данных**

*Таблица 5 - Инструкции по проверке совместимости*

| № п/п | Описание требования | Критерий оценки успешности проверки | Примечание |
| --- | --- | --- | --- |
| **1. Включение модуля виртуализации, создание ВМ** |
| 1.1 | Проверить поддержку вложенной виртуализации.Включить модуль виртуализации. | Вложенная виртуализация присутствует.Модуль virtualization включен. | **# На нодах выполните команду:** egrep "svm|vmx" /proc/cpuinfo**# Успешный вывод данных будет означать, что вложенная виртуализация доступна****# Проведите предварительную настройку окружения и ресурсов, указанных ниже:**export NAMESPACE=<namespace name for tests>export STORAGE\_CLASS\_NAME=<storage class name for tests>export DVCR\_STORAGE\_CLASS=<storage class name for dvcr registry volume> # should use a different from $STORAGE\_CLASS\_NAME**# Задайте переменные для доступа к ВМ**export SSH\_USER=$(echo $USER)export SSH\_PWD=$(openssl passwd -6 **<input your desired password here>**)export SSH\_PUBKEY=$(cat ~/.ssh/id\_rsa.pub)# Проверьте наличие ssh ключей командой ***echo $SSH\_PUBKEY***# Если вывод пустой - сгенерируйте тестовую пару командой ***ssh-keygen -t rsa -f ~/.ssh/id\_rsa -N ''***cat > sc\_dvcr.yaml <<EOFallowVolumeExpansion: trueapiVersion: storage.k8s.io/v1kind: StorageClassmetadata: name: ${DVCR\_STORAGE\_CLASS}parameters: csi.storage.k8s.io/fstype: ext4 poolName: SSD protocol: iscsi storageID: yadro-san-1 volumeExportPort: p50,p51,p60,p61provisioner: csi-tatlinunified.yadro.comreclaimPolicy: DeletevolumeBindingMode: ImmediateEOFkubectl apply -f sc\_dvcr.yaml**# Cоздайте moduleConfig и modulePullOverride для модуля виртуализации**cat > virt\_mc.yaml <<EOF apiVersion: deckhouse.io/v1alpha1kind: ModuleConfigmetadata: name: virtualizationspec: enabled: true settings: dvcr: storage: persistentVolumeClaim: size: 55G storageClassName: ${DVCR\_STORAGE\_CLASS} type: PersistentVolumeClaim virtualMachineCIDRs: - 10.66.10.0/24 - 10.66.20.0/24 - 10.66.30.0/24 version: 1EOFcat > virt\_mpo.yaml <<EOFapiVersion: deckhouse.io/v1alpha1 kind: ModulePullOverride metadata:  name: virtualization spec:  imageTag: main  scanInterval: 15s  source: deckhouseEOFkubectl apply -f virt\_mc.yamlkubectl apply -f virt\_mpo.yaml |
| 1.2 | Подготовить образ ВМ и storage class | storage class создан, образ загружен и в статусе Ready | **# Создайте storageClass и clusterVirtualImage, требующиеся для проведения теста.**cat > sc.yaml <<EOFallowVolumeExpansion: trueapiVersion: storage.k8s.io/v1kind: StorageClassmetadata: annotations: storageclass.kubernetes.io/is-default-class: "true" virtualdisk.virtualization.deckhouse.io/access-mode: ReadWriteMany virtualdisk.virtualization.deckhouse.io/volume-mode: Block name: csi-tatlinunified-iscsi-ssd-dvp-testparameters: poolName: SSD protocol: iscsi storageID: yadro-san-1 volumeExportPort: p50,p51,p60,p61provisioner: csi-tatlinunified.yadro.comreclaimPolicy: DeletevolumeBindingMode: ImmediateEOFcat > cvi.yaml <<EOFapiVersion: virtualization.deckhouse.io/v1alpha2kind: ClusterVirtualImagemetadata: name: ubuntu22spec: dataSource: type: "HTTP" http: url: "https://0e773854-6b4e-4e76-a65b-d9d81675451a.selstorage.ru/ubuntu/jammy-minimal-cloudimg-amd64.img"EOFkubectl apply -f sc.yamlkubectl apply -f cvi.yaml**# Дождитесь, пока cluster image не перейдёт в статус Ready:**kubectl get cvi ubuntu22 |
| 1.3 | Создать ВМ и 3 диска к ней | ВМ запущена, диски созданы | **# Подготовка файла с конфигурацией ВМ**cat > ubuntu.yaml <<EOFapiVersion: virtualization.deckhouse.io/v1alpha2kind: VirtualDiskmetadata: annotations: virtualdisk.virtualization.deckhouse.io/volume-mode: Block virtualdisk.virtualization.deckhouse.io/access-mode: ReadWriteOnce name: ubuntu22-1-3spec: persistentVolumeClaim: size: 20Gi storageClassName: ${STORAGE\_CLASS\_NAME}---apiVersion: virtualization.deckhouse.io/v1alpha2kind: VirtualDiskmetadata: name: ubuntu22-1-2 annotations: virtualdisk.virtualization.deckhouse.io/volume-mode: Block virtualdisk.virtualization.deckhouse.io/access-mode: ReadWriteOncespec: persistentVolumeClaim: size: 20Gi storageClassName: ${STORAGE\_CLASS\_NAME}---apiVersion: virtualization.deckhouse.io/v1alpha2kind: VirtualDiskmetadata: name: ubuntu22-1-1 annotations: virtualdisk.virtualization.deckhouse.io/volume-mode: Block virtualdisk.virtualization.deckhouse.io/access-mode: ReadWriteOncespec: persistentVolumeClaim: size: 30Gi storageClassName: ${STORAGE\_CLASS\_NAME}---apiVersion: virtualization.deckhouse.io/v1alpha2kind: VirtualMachinemetadata: name: ubuntu22-1 labels: vm: linux service: v1spec: virtualMachineClassName: host runPolicy: AlwaysOn osType: Generic bootloader: BIOS virtualMachineIPAddressName: ubuntu22-1 cpu: cores: 1 memory: size: 2Gi blockDeviceRefs: - kind: ClusterVirtualImage name: ubuntu22 - kind: VirtualDisk name: ubuntu22-1-1 - kind: VirtualDisk name: ubuntu22-1-2 - kind: VirtualDisk name: ubuntu22-1-3  provisioning: type: UserData userData: | #cloud-config package\_update: true packages: - qemu-guest-agent runcmd: - [ hostnamectl, set-hostname, ubuntu22-0 ] - [ systemctl, daemon-reload ] - [ systemctl, enable, --now, qemu-guest-agent.service ] user: ubuntu password: ubuntu ssh\_pwauth: True chpasswd: { expire: False } users: - name: ${SSH\_USER} passwd: ${SSH\_PWD} ssh\_authorized\_keys: - ${SSH\_PUBKEY} shell: /bin/bash sudo: ALL=(ALL) NOPASSWD:ALL chpasswd: { expire: False } lock\_passwd: false---apiVersion: virtualization.deckhouse.io/v1alpha2kind: VirtualMachineIPAddressmetadata: name: ubuntu22-1spec: staticIP: 10.66.10.10 **type: Static****EOF****# Создайте ВМ и дождитесь, когда она перейд**ёт в статус Running:kubectl -n $NAMESPACE apply -f ubuntu.yamlkubectl -n $NAMES**PACE get vm -owide -w****# Проверьте что диски созданы успешно - все они должны б**ыть в статусе Ready:kubectl -n $NAMESPACE get vd -w |
| **2. Изменение размера тома** |
| 2.1 | создать файл на диске, проверить его sha256 сумму | Файл создан, сумма считана | cat > test.sh << EOF#!/bin/bash(echo n; echo p; echo -ne '\n'; echo -ne '\n'; echo -ne '\n'; echo w) | fdisk /dev/sdd >/dev/null 2>&1 | mkdir /mnt/disk0 >/dev/null 2>&1 && yes | mkfs.ext4 /dev/sdd >/dev/null 2>&1 && mount /dev/sdd /mnt/disk0dd if=/dev/random of=/mnt/disk0/test\_file bs=1M status=none count=100echo "===== SHA cheksum: ===="sha256sum /mnt/disk0/test\_fileEOFchmod +x test.sh**# Выполните этот скрипт на ВМ следующей командой**ssh ubuntu@10.66.10.10 'sudo -i && bash -s' < test.sh **# Укажите пароль "*ubuntu*" при подключении к ВМ, и дождитесь, пока на тестовом диске не создастся и будет примонтирован разделы, а так же не создастся тестовый файл. В конце скрипт выведет контрольную сумму тестового файла, скопируйте её для дальнейшего сравнения.****# Пример вывода:****## ===== SHA cheksum: ====****## 787c6e00ca09c254d9c44ca61fea4bec73718e74ee2735df4ae3a8e38e871874 /mnt/disk0/test\_file****# Проверьте и запомните текущий размер тестового файла.**ssh ubuntu@10.66.10.10 'sudo ls -lah /mnt/disk0/test\_file' |
| 2.2 | Увеличить размер виртуального диска | Размер диска изменён успешно | **# Укажите желаемый размер для диска**export NEW\_VD\_SIZE=31Giexport VD\_NAME=ubuntu22-1-1**# Проверьте текущий размер диска**kubectl get -n $NAMESPACE vd $VD\_NAME**# Измените размер**kubectl -n "${NAMESPACE}" patch virtualdisk "${VD\_NAME}" --type merge -p '{"spec":{"persistentVolumeClaim":{"size":"'${NEW\_VD\_SIZE}'"}}}'**# Дождитесь, пока размер дисков изменится:**kubectl -n ${NAMESPACE} get vd ${VD\_NAME} -w |
| 2.3 | Проверить размер PVC | Успешно изменился размер PVC на величину, указанную в пункте 2.1 | **# Проверьте текущий размер pvc для данного диска - он должен быть равен тому, что был указан при ресайзе самого диска**kubectl -n $NAMESPACE get vd $VD\_NAME -o json | jq -r '.status.target.persistentVolumeClaimName' | xargs -I {} kubectl -n $NAMESPACE get pvc {} -owide |
| 2.4 | Убедиться, что файл (размер) и sha256 файла остались неизменны | Размер файла и sha256 не изменились | **# Проверьте, что sha256 сумма осталась прежней:****# При подключении снова укажите пароль ubuntu**ssh ubuntu@10.66.10.10 'sudo sha256sum /mnt/disk0/test\_file; sudo ls -la /mnt/disk0/test\_file'  |
| **3. live миграция виртуальных машин** |
| 3.1 | Создать файлы на дисках ВМ | Файлы созданы, получены sha256 суммы | **#Создайте скрипт для настройки дисков на ВМ:**cat > test.sh <<EOF#!/bin/bash(echo n; echo p; echo -ne '\n'; echo -ne '\n'; echo -ne '\n'; echo w) | fdisk /dev/sdc >/dev/null 2>&1 | mkdir /mnt/disk1 >/dev/null 2>&1 && yes | mkfs.ext4 /dev/sdc >/dev/null 2>&1 && mount /dev/sdc /mnt/disk1(echo n; echo p; echo -ne '\n'; echo -ne '\n'; echo -ne '\n'; echo w) | fdisk /dev/sdd >/dev/null 2>&1 | mkdir /mnt/disk2 >/dev/null 2>&1 && yes | mkfs.ext4 /dev/sdd >/dev/null 2>&1 && mount /dev/sdd /mnt/disk2dd if=/dev/random of=/mnt/disk1/test\_file1 bs=1M status=none count=100dd if=/dev/random of=/mnt/disk2/test\_file2 bs=1M status=none count=100echo "===== SHA cheksums: ===="sha256sum /mnt/disk1/test\_file1sha256sum /mnt/disk2/test\_file2EOFchmod +x [test.sh](http://test.sh/)**# Dыполните этот скрипт на ВМ следующей командой**ssh ubuntu@10.66.10.10 'sudo -i && bash -s' < test.sh **# Укажите пароль ubuntu при подключении к ВМ, и дождитесь, пока на тестовых дисках не создатутся и будут примонтированы разделы, а так же не создадутся тестовые файлы. В конце скрипт выведет контрольные суммы созданных тестовых файлов, скопируйте их для дальнейшего сравнения.** |
| 3.2 | Выполнить "живую" миграцию ВМ | ВМ смигрированы | **# Проверьте текущий узел, на котором запущена ВМ:**kubectl -n $NAMESPACE get vm -owide**# Создайте ресурс для миграции вм и примените его:**cat > migrate.yaml << EOFapiVersion: internal.virtualization.deckhouse.io/v1kind: InternalVirtualizationVirtualMachineInstanceMigrationmetadata: name: migrate-ubuntu-vm-1spec: vmiName: ubuntu22-1EOFkubectl -n $NAMESPACE apply -f migrate.yaml**# Наблюдайте за ВМ, пока миграция не завершится:**kubectl -n $NAMESPACE get vm -owide -w |
| 3.3 | Убедиться, что файлы так же смигрировали, их sha256 не изменился | Фалы успешно смигрировалиsha256 файлов не именился | **# Когда миграция завершилась (сменился узел, и ВМ перешла из статуса Migrating в Running) - проверьте, что sha256 суммы остались прежними:**ssh ubuntu@10.66.10.10 'sudo sha256sum /mnt/disk1/test\_file1; sudo sha256sum /mnt/disk2/test\_file2'**# При подключении снова укажите пароль ubuntu** |
| **4. Удаление ВМ и дисков** |
| 4.1 | Получить имя PVC + PV для тестового VirtualDisk | Данные получены | **# Для диска ubuntu22-1-3 найдите имя PVC и PV ресурсов**export PVC\_NAME=$( k -n $NAMESPACE get vd $VD\_NAME -owide | awk 'NR>1{print $6}')export PV\_NAME=$(kubectl -n $NAMESPACE get pvc $PVC\_NAME -owide | awk 'NR>1{print $3}')echo $PVC\_NAMEecho $PV\_NAME |
| 4.2 | Найти соответствующий тому ресурс в СХД/SDS | На СХД найден ресурс, имя которого равно имени PV, полученного в пункте 3.1 | Зайти на веб-интерфейс СХД - "логические объекты" - "ресурсы"(имя ресурса на СХД = вывод команды **echo $PV\_NAME**)Найти по имени pv ресурс, проверить текущий размер пула, в котором он создан |
| 4.3 | Удалить ВМ и сопутствующие ресурсы | Удалена ВМ, диски и vmip | kubectl -n $NAMESPACE delete -f ubuntu.yaml |
| 4.4 | Проверить, что ресурсы удалены | Ресурсов, выделенных под ВМ, нет в кластере | **# Проверьте, что ВМ с дисками и виртуальным ip адресом удалена - вывод нижеследующих команд не должен выдать каких-либо ресурсов**kubectl -n $NAMESPACE get vmkubectl -n $NAMESPACE get vdkubectl -n $NAMESPACE get vmip**# А так же, проверьте, что удалились ресурсы тестового диска**kubectl -n $NAMESPACE get pvc $PVC\_NAMEkubectl get pv $PV\_NAME |
| 4.5 | На СХД/SDS должен удалиться ресурс и должно освободиться место | Ресурс на СХД удалён, место освобождено | Зайти на веб-интерфейс СХД - "логические объекты" - "ресурсы"(имя ресурса на СХД = вывод команды **echo $PV\_NAME**)Проверить, что ресурс удалился, а количество свободного места в пуле увеличилось |
| **5. Запуск e2e тестов от Storage** |
| 5.1 | Запустить e2e тесты | e2e тесты отработали, ошибок нет | Тесты в цикле:- Создают ВМ + диски- Изменяют размер виртуального диска- Удаляют ВМ + диски |