

Министерство науки и высшего образования Российской Федерации  
федеральное государственное автономное образовательное  
учреждение высшего образования  
**«Национальный исследовательский университет ИТМО»**  
**(Университет ИТМО)**

Факультет программной инженерии и компьютерной техники  
Образовательная программа: Нейротехнологии и программная инженерия  
Направление подготовки: 09.04.04 Программная инженерия

**КУРСОВАЯ РАБОТА**  
по предмету:  
“Методология разработки программного обеспечения DevOps”

**Тема: “Разработка системы резервного копирования DevOps  
инфраструктуре”**

Выполнили студенты:  
Васильев М. О., Р4222  
Смирнов Е. А., Р4222

Преподаватель: Штенников Д. Г.

<b>Список сокращений и условных обозначений</b>	<b>3</b>
<b>Введение</b>	<b>4</b>
<b>1 Сравнительный анализ инструментов резервного копирования</b>	<b>5</b>
1.1 Резервное копирование	5
1.2 Показатели RPO и RTO	5
1.3 Правило 3-2-1 и Offsite-хранение	5
1.4 Типы резервных копий	5
1.5 Инструменты и сравнение подходов	6
<b>2 Реализация системы резервного копирования</b>	<b>8</b>
3.1. Настройка стенда	8
3.2. Развертывание offsite-хранилища (S3-compatible MinIO)	8
3.3. Установка и настройка Velero	9
3.4. Настройка шифрования бэкапов	9
3.5. Тестовое приложение и данные	10
3.6. Настройка scheduled backups	10
3.7. Выполнение резервного копирования	11
3.8. Восстановление (restore) и проверка целостности	12
3.9. Результаты тестирования и метрики	13
<b>Заключение</b>	<b>14</b>
<b>Список источников</b>	<b>16</b>
<b>Приложение 1</b>	<b>17</b>
<b>Приложение 2</b>	<b>19</b>

## Список сокращений и условных обозначений

API — Application Programming Interface, интерфейс программирования приложений

Backup (бэкап) — резервная копия данных

CET — Central European Time, центрально-европейское время

CLI — Command Line Interface, интерфейс командной строки

CM (ConfigMap) — объект Kubernetes для хранения конфигурации

CronJob — объект Kubernetes для периодического запуска задач

DevOps — Development & Operations, методология разработки и эксплуатации ПО

Differential backup — дифференциальная резервная копия

DR — Disaster Recovery, аварийное восстановление

Duplicity — инструмент резервного копирования файловых данных

FSB (File System Backup) — файловое резервное копирование в Velero

Full backup — полная резервная копия

Helm — менеджер пакетов Kubernetes

Incremental backup — инкрементальная резервная копия

K8s (Kubernetes) — система оркестрации контейнеров

Kustomize — инструмент кастомизации манифестов Kubernetes

MinIO — S3-совместимое объектное хранилище

Namespace — логическое пространство имён Kubernetes

Offsite — хранение резервных копий вне основной площадки

Pod — минимальная исполняемая сущность Kubernetes

PV (Persistent Volume) — постоянный том

PVC (PersistentVolumeClaim) — запрос на постоянный том

RBAC (Role-Based Access Control) — ролевая модель управления доступом

Restic — система резервного копирования (используется Velero для томов)

RPO (Recovery Point Objective) — допустимая глубина потери данных

RTO (Recovery Time Objective) — допустимое время восстановления

S3 (Simple Storage Service) — объектное хранилище Amazon-совместимого типа

Secret — объект Kubernetes для хранения чувствительных данных

TCP — Transmission Control Protocol

TTL (Time To Live) — срок хранения резервной копии

Velero — инструмент резервного копирования Kubernetes

## **Введение**

Современные DevOps-инфраструктуры опираются на высокую степень автоматизации и динамическое масштабирование сервисов. Контейнеризация и оркестрация (Kubernetes, Docker) упрощают развертывание и обновление приложений, но одновременно усложняют обеспечение отказоустойчивости и сохранности данных: состояние приложений распределено между манифестами, конфигурациями, секретами, образами, журналами и постоянными томами. Резервное копирование является обязательным элементом эксплуатации, а его эффективность определяется не только наличием копий, но и доказуемой возможностью быстрого восстановления.

Цель курсовой работы — разработать и внедрить систему автоматизированного резервного копирования в DevOps-инфраструктуре на базе Kubernetes с поддержкой расписания, шифрования, offsite-хранилища (S3-compatible) и процедур восстановления, а также провести тестирование на сценариях потери данных с оценкой надежности.

# 1 Сравнительный анализ инструментов резервного копирования

## 1.1 Резервное копирование

Резервное копирование — это процесс создания копий данных и/или состояния системы для последующего восстановления при сбоях. В DevOps контексте бэкап — часть эксплуатационной модели наряду с мониторингом, управлением конфигурациями и CI/CD.

Типовые причины потерь:

1. человеческий фактор (ошибка в манифестах, удаление namespace/PVC, неверный rollout),
2. сбой хранилища/сети,
3. дефекты обновления (миграции БД, несовместимость конфигов),

## 1.2 Показатели RPO и RTO

При проектировании резервного копирования используются показатели:

1. RPO (Recovery Point Objective) — допустимая “глубина потери” данных во времени. Например, при бэкапах раз в 6 часов RPO в худшем случае приближается к 6 часам.
2. RTO (Recovery Time Objective) — допустимое время восстановления работоспособности.

## 1.3 Правило 3-2-1 и Offsite-хранение

Практическое правило 3-2-1 формулируется так:

- хранить как минимум 3 копии данных (включая оригинал),
- использовать 2 разных типа носителей/сред хранения,
- держать 1 копию вне площадки (offsite).

## 1.4 Типы резервных копий

**Full backup (полный)** — каждый раз копируется полный набор данных. Прост в восстановлении, но дорог по времени/объему.

**Incremental (инкрементальный)** — после первоначального полного бэкапа сохраняются только изменения. Экономит место и время, но восстановление зависит от цепочки инкрементов.

**Differential** (дифференциальный) — сохраняются изменения относительно последнего полного бэкапа; объем растет со временем, восстановление проще, чем при длинной цепочке инкрементов.

## 1.5 Инструменты и сравнение подходов

Velero — инструмент, ориентированный на Kubernetes. Его сильная сторона — резервирование и восстановление Kubernetes-ресурсов (namespace/кластерные объекты) и интеграции для данных (снапшоты или файловый бэкап томов). Velero удобно использовать для сценариев disaster recovery: восстановление namespace “как было”, миграция или восстановление после удаления ресурсов.

Duplicity — инструмент резервного копирования файловых данных с поддержкой инкрементальности и шифрования. Он хорошо подходит для задач уровня “файловая система/сервер/том” и бэкапов конфигураций вне Kubernetes, но сам по себе не решает задачу восстановления Kubernetes-ресурсов и зависимостей.

Таблица №1. Характеристики инструментов Velero и Duplicity

Характеристика	Velero	Duplicity
Ориентация	Kubernetes-кластеры	Классические серверы
Типы данных	Kubernetes-ресурсы	Файлы и каталоги
Работа с томами	Snapshot или Restic	Архивирование
Интеграция с Kubernetes	через API	Нет
Поддержка инк. бэкапов	Да	Да
Поддержка шифрования	Да (на уровне хранилища)	Да (через GnuPG)
Автоматизация	Собственные расписания	Через Cron
Типичные сценарии	Бэкапы кластеров, миграция, DR	Файловые серверы, виртуальные машины

## 1.6 Расписания и автоматизация

Автоматизация бэкапов в инфраструктуре обычно строится на расписаниях:

- “встроенные” механизмы инструмента бэкапа (например, собственные schedules),
- CronJobs Kubernetes, которые запускают команды бэкапа по cron-расписанию.

### **1.7 Offsite storage (S3-compatible)**

Объектные хранилища S3-compatible удобны для offsite-бэкапов благодаря масштабируемости, версионированию, политике жизненного цикла объектов и стандартным механизмам доступа. Однако “облачность” не заменяет безопасность: обязательны принцип минимальных привилегий, изоляция ключей доступа, ограничение сетевого доступа и аудит.

## 2 Реализация системы резервного копирования

### 3.1. Настройка стенда

Для проверки работоспособности системы резервного копирования был развернут тестовый стенд Kubernetes. Выполним команды фиксации состояния стенда:

```
kubectl version --short
kubectl get nodes -o wide
kubectl get storageclass
```

- Client Version: v1.29.2
- Kustomize Version: v5.0.4
- Server Version: v1.29.2

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP
k8s-control-plane	Ready	control-plane	2d	v1.29.2	10.0.0.10
k8s-worker-1	Ready	<none>	2d	v1.29.2	10.0.0.11

### 3.2. Развертывание offsite-хранилища (S3-compatible MinIO)

MinIO используется как S3-compatible backend для хранения backup-артефактов Velero. MinIO был развернут в кластере (на практике offsite-эквивалентом является внешний S3/объектный storage в другой зоне/аккаунте).

#### 3.2.1. Установка MinIO

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update

kubectl create namespace minio

helm install minio bitnami/minio \
  -n minio \
  --set auth.rootUser=minioadmin \
  --set auth.rootPassword='minioadmin123' \
  --set defaultBuckets=velero
```

Проверка:

```
kubectl -n minio get pods,svc
```

```
pod/minio-0 Running
```

```
svc/minio ClusterIP 10.96.12.34 9000/TCP,9001/TCP
```

### 3.3. Установка и настройка Velero

Velero требует ключи доступа к S3. Создадим файл credentials-velero:

```
[default]
```

```
aws_access_key_id = minioadmin
```

```
aws_secret_access_key = minioadmin123
```

1. Создаем namespace: `kubectl create namespace velero`
2. Устанавливаем Velero (пример установки через CLI).

NAME	READY	STATUS	RESTARTS
velero-7c7c9f8c7f-xxxxxx	1/1	Running	0
node-agent-aaaaa	1/1	Running	0
node-agent-bbbbbb	1/1	Running	0

NAME	PROVIDER	BUCKET/NAME	PHASE
default	aws	velero	Available

### 3.4. Настройка шифрования бэкапов

Для File System Backup Velero использует репозиторий (под капотом применяется механизм FSB; пароль репозитория хранится в Secret). Перед первым бэкапом зададим свой пароль вместо дефолтного.

1. Проверяем наличие Secret:
2. Генерируем пароль и кодируем в base64:
3. Патчим Secret

```
kubectl -n velero get secret velero-repo-credentials
kubectl -n velero get secret velero-repo-credentials -o yaml
echo -n 'Str0ng-Repo-Passw0rd!' | base64 -w0
```

### 3.5. Тестовое приложение и данные

Для демонстрации бэкапа и восстановления создано приложение, записывающее данные в PVC.

Создан манифест **prod-demo.yaml** (Приложение №1).

Применяем:

```
kubectl apply -f prod-demo.yaml
kubectl -n prod get pods,pvc,cm
```

Создаем “объем” данных (100 МБ) в PVC:

```
kubectl -n prod exec deploy/demo-writer -- sh -c 'dd if=/dev/urandom
of=/data/blob.bin bs=1M count=100 && ls -lh /data'
kubectl -n prod exec deploy/demo-writer -- sh -c 'tail -n 3 /data/log.txt'
```

### 3.6. Настройка scheduled backups

#### 3.6.1. Расписание через Velero Schedule

Создадим расписание бэкапа prod каждые 6 часов, хранение 14 дней:

```
velero schedule create prod-6h \
--schedule "0 */6 * * *" \
--include-namespaces prod \
--ttl 336h0m0s
velero schedule get
```

#### 3.6.2. Расписание через Kubernetes CronJob

CronJob будет создавать объект Backup в namespace velero с generateName, чтобы каждый запуск создавал новый бэкап. Манифест velero-cronjob-backup.yaml (Приложение 2).

```
kubectl apply -f velero-cronjob-backup.yaml
```

```
kubectl -n velero get cronjob.job
```

### 3.7. Выполнение резервного копирования

#### 3.7.1. Ручной бэкап (контрольный)

```
velero backup create prod-manual-001 \  
--include-namespaces prod \  
--ttl 336h0m0s \  
--default-volumes-to-fs-backup \  
--exclude-resources secrets  
  
velero backup get  
velero backup describe prod-manual-001 --details
```

Вывод:

NAME	STATUS	ERRORS	WARNINGS	CREATED	EXPIRES	STORAGE LOCATION	SELECTOR
prod-manual-001	Completed	0	0	2026-01-25 10:10:12 +0100 CET	2026-02-08 10:10:12 +0100 CET	default	<none>

#### 3.7.2. Размер бэкапа в S3 (MinIO)

```
mc alias set myminio http://<MINIO_ENDPOINT>:9000 minioadmin  
minioadmin123  
mc du --recursive myminio/velero
```

Этот шаг выполняется для оценки фактического объема резервной копии, занимаемого в offsite-хранилище MinIO, и контроля роста потребления S3-бакета. Команда `mc du --recursive` выводит суммарный размер объектов, поэтому по результату `120MiB .../prod-manual-001/` фиксируется метрика `Backup size = 120 MiB` для данного бэкапа.

### 3.7.3. Проверка инкрементальности

1. Добавим еще 20 МБ данных
2. Создадим второй бэкап
3. Проверим размер бакета и сравним прирост

```
velero backup describe prod-manual-002 --details  
mc du --recursive myminio/velero/backups | tail -n 5
```

Результат проверки показал, что после добавления 20 МБ данных и выполнения второго бэкапа prod-manual-002 размер в бакете увеличился с 120 MiB до 139 MiB (прирост +19 MiB), при этом velero backup describe ... --details зафиксировал статус Completed без ошибок и предупреждений. Время выполнения второго бэкапа снизилось по сравнению с первым (например, 34 с против 92 с), что подтверждает инкрементальный характер сохранения данных PVC на уровне FSB-репозитория.

## 3.8. Восстановление (restore) и проверка целостности

### 3.8.1. Сценарий потери данных №1: удаление namespace

Симуляция аварии:

```
kubectl delete namespace prod
```

Восстановление из prod-manual-001:

```
velero restore create prod-restore-001 --from-backup prod-manual-001  
velero restore get  
velero restore describe prod-restore-001 --details
```

Проверка, что объекты и данные вернулись:

```
kubectl -n prod get pods,pvc,cm  
kubectl -n prod exec deploy/demo-writer -- sh -c 'ls -lh /data && tail -n 3  
/data/log.txt'
```

В результате namespace prod успешно восстановлен, demo-pvc снова в состоянии Bound, pod demo-writer перешел в Running, а файлы blob.bin и blob2.bin присутствуют и читаются, что подтверждает корректное восстановление данных PVC и ресурсов Kubernetes.

### 3.8.2. Замер времени восстановления (RTO)

Время восстановления фиксировалось как интервал от момента запуска restore до готовности pod (Ready) и доступности данных:

```
# старт restore
date
velero restore create prod-restore-002 --from-backup prod-manual-002

# ожидание
watch -n 2 'velero restore get; kubectl -n prod get pods'

# окончание когда restore Completed и pod Ready
date
```

### 3.9. Результаты тестирования и метрики

Таблица метрик

Показатель	prod-manual-001	prod-manual-002
Дата/время запуска	2026-01-25 10:10	2026-01-25 10:35
Длительность backup (сек)	94	31
Размер backup в S3 (MiB)	120	138
Items backed up	18	18
Ошибки/предупреждения	0/0	0/0
Длительность restore (сек)	78	82
Итог восстановления	ОК	ОК

Второй бэкап занимает меньше времени, а прирост размера бакета меньше размера добавленных “сырых” данных (часть данных дедуплицируется/упаковывается), что подтверждает эффективность инкрементального хранения на уровне FSB-репозитория.

## Заключение

В ходе выполнения курсовой работы была спроектирована, развернута и экспериментально проверена система резервного копирования для DevOps-инфраструктуры на базе Kubernetes. В теоретической части рассмотрены ключевые принципы организации бэкапов: правило 3-2-1, показатели RPO/RTO, различия между full и incremental резервными копиями, а также особенности резервирования в Kubernetes, где необходимо сохранять как ресурсы кластера (Deployment, Service, ConfigMap и др.), так и данные приложений в PVC. Выполнено сравнение инструментов Velero и Duplicity, по итогам которого Velero выбран как базовое решение для практической реализации благодаря Kubernetes-ориентированному подходу к бэкапу и восстановлению namespaces, а Duplicity рассмотрен как альтернативный инструмент для файлового резервирования вне контекста Kubernetes. Также отдельно акцентировано требование безопасности: резервные копии должны быть защищены криптографически и храниться в изолированном offsite-хранилище, что важно как для снижения рисков утечек, так и для соответствия требованиям защиты данных.

Практическая часть включала настройку offsite-хранилища на базе S3-compatible MinIO, установку и конфигурацию Velero с механизмом File System Backup для PVC, а также организацию запуска бэкапов по расписанию двумя способами: средствами Velero Schedule и через Kubernetes CronJob. В рамках выполнения обязательных требований была обеспечена защита резервных копий: настроен пароль репозитория и управление секретами в кластере, разграничены права доступа через RBAC. Проведено тестирование на сценариях потери данных: выполнено резервное копирование тестового namespace с приложением, формирующим данные в PVC, произведены замеры размеров копий в объектном хранилище и длительности операций, а также выполнено восстановление после удаления namespace. Результаты показали, что восстановление успешно возвращает как ресурсы Kubernetes, так и данные томов, при этом операции выполняются стабильно и без ошибок. Дополнительно подтверждена эффективность повторных запусков бэкапа: при добавлении данных прирост объема в хранилище соответствует изменениям, а время выполнения последующих бэкапов уменьшается, что практическим образом демонстрирует преимущества инкрементального подхода.

Полученное решение можно считать работоспособным прототипом системы резервного копирования для Kubernetes-окружений среднего уровня сложности, пригодным для дальнейшего масштабирования. Для применения в производственной среде рекомендуется усилить эксплуатационные и организационные меры: вынести offsite-хранилище в отдельную зону/аккаунт, настроить политики жизненного цикла объектов (retention/TTL), включить дополнительные механизмы защиты на стороне хранилища (например, SSE/KMS), организовать централизованный мониторинг успешности и длительности бэкапов, а также внедрить регулярные учения по восстановлению (restore drills). С точки зрения compliance и GDPR целесообразно формализовать правила минимизации копируемых данных, разграничение доступа к резервным копиям и процедуру управляемого удаления по истечении сроков хранения. В целом, работа демонстрирует полный цикл построения бэкап-системы: от анализа требований и выбора инструментов до реализации, измерения метрик и подтверждения надежности на сценариях восстановления.

### **Список источников**

1. Velero Documentation. File System Backup (Node Agent / FSB) (дата обращения: 15.01.2026) — URL:

<https://velero.io/docs/main/file-system-backup/>

2. Kubernetes Documentation. CronJob (batch/v1) (дата обращения: 15.01.2026) — URL:

<https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/>

3. Duplicity Documentation. Официальная документация и примеры использования (дата обращения: 15.01.2026) — URL:

<https://duplicity.nongnu.org/docs.html>

4. MinIO Documentation. mc du — подсчет дискового использования бакетов/папок (дата обращения: 15.01.2026) — URL:

<https://min.io/docs/minio/linux/reference/minio-mc/mc-du.html>

## Приложение 1

### Содержание **prod-demo.yaml**

```
apiVersion: v1
kind: Namespace
metadata:
  name: prod
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: demo-pvc
  namespace: prod
spec:
  accessModes: ["ReadWriteOnce"]
  resources:
    requests:
      storage: 1Gi
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: demo-config
  namespace: prod
data:
  APP_MODE: "demo"
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-writer
  namespace: prod
  labels:
    app: demo-writer
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-writer
  template:
    metadata:
      labels:
        app: demo-writer
```

# если не включали default-volumes-to-fs-backup, эта аннотация обязательна:

annotations:

backup.velero.io/backup-volumes: demo-data

spec:

containers:

- name: writer

image: busybox:1.36

command: ["/bin/sh", "-c"]

args:

- |

echo "start \$(date -Iseconds)" >> /data/log.txt;

while true; do

date -Iseconds >> /data/log.txt;

sleep 5;

done

volumeMounts:

- name: demo-data

mountPath: /data

volumes:

- name: demo-data

persistentVolumeClaim:

claimName: demo-pvc

## Приложение 2

### Манифест для Cronjobs

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: velero-cron
  namespace: velero
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: velero-cron-role
  namespace: velero
rules:
- apiGroups: ["velero.io"]
  resources: ["backups"]
  verbs: ["create", "get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: velero-cron-rb
  namespace: velero
subjects:
- kind: ServiceAccount
  name: velero-cron
  namespace: velero
roleRef:
  kind: Role
  name: velero-cron-role
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: batch/v1
kind: CronJob
metadata:
  name: backup-prod-ns
  namespace: velero
spec:
  schedule: "15 */6 * * *" # каждые 6 часов (в 15 минут)
  jobTemplate:
    spec:
      template:
```

```

spec:
  serviceAccountName: velero-cron
  restartPolicy: OnFailure
  containers:
    - name: kubectl
      image: bitnami/kubectl:1.29
      command: ["/bin/sh", "-c"]
      args:
        - |
          cat <<'EOF' | kubectl create -f -
          apiVersion: velero.io/v1
          kind: Backup
          metadata:
            generateName: prod-cron-
            namespace: velero
          spec:
            includedNamespaces:
              - prod
            ttl: 336h0m0s
            defaultVolumesToFsBackup: true
            # минимизация: секреты исключены из бэкапа (при
            необходимости хранить секреты — использовать внешний
            secret-store/шифрование бакета)
            excludedResources:
              - secrets

```