|  |  |
| --- | --- |
|  |  |

КОНКУРСНОЕ ЗАДАНИЕ

КОМПЕТЕНЦИИ

«Облачные технологии»

Региональный этап чемпионата по профессиональному мастерству «Профессионалы» в 2026г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(субъект РФ)

2026г.

Конкурсное задание разработано экспертным сообществом и утверждено Менеджером компетенции, в котором установлены нижеследующие правила и необходимые требования владения профессиональными навыками для участия в соревнованиях по профессиональному мастерству.

**Модуль А. Развертывание пула серверов для организации сетевого взаимодействия (инвариант)**

*Время на выполнение модуля: 5 часов*

**Задания:**

1. Разверните четыре виртуальные машины на Linux: **vm-nexus01**, **vm-runner01**, **vm-dev01**, **vm-gitlab** со следующими характеристиками:
   1. **vm-nexus01**:
      1. vCPU — 4;
      2. RAM — 8 ГБ;
      3. Disk — 60 ГБ;
      4. Статическая конфигурация сетевых параметров.
   2. **vm-dev01**:
      1. vCPU — 2;
      2. RAM — 4 ГБ;
      3. Disk — 30 ГБ;
      4. Статическая конфигурация сетевых параметров.
   3. **vm-runner01**:
      1. vCPU — 4;
      2. RAM — 4 ГБ;
      3. Disk — 30 ГБ;
      4. Статическая конфигурация сетевых параметров.
   4. **vm-gitlab**:
      1. vCPU — 2;
      2. RAM — 8 ГБ;
      3. Disk — 50 ГБ;
      4. Статическая конфигурация сетевых параметров.
   5. Для развёртывания виртуальных машин используйте шаблон **alt-p11-cloud-template**, подготовленный на базе образа [alt-p11-cloud-x86\_64.qcow2](http://ftp.altlinux.org/pub/distributions/ALTLinux/p11/images/cloud/x86_64/).
   6. Обеспечьте доступ до каждой ВМ из-под пользователя **altlinux** с рабочего места по SSH.
   7. Обеспечьте консольный доступ до каждой ВМ из-под пользователя **altlinux** с паролем **P@ssw0rd**.
   8. Все созданные ВМ должны быть доступны с рабочего места по именам.
      1. Имя должно быть в формате **fqdn**
         1. В качестве доменной области используйте **cloud2026.region**.
2. Установите Docker на виртуальную машину **vm-nexus01**.
   1. Напишите файл **compose.yml** для развёртывания настроенного и готового к работе **Nexus**:
      1. Настройте volume для сохранения данных сервиса Nexus с именем **nexus-data**;
      2. В качестве СУБД используйте **PostgreSQL**;
      3. Предусмотрите **healthcheck** для контейнера СУБД и **очередность запуска**.
      4. С рабочего места доступ до веб-интерфейса Nexus должен быть по **http://vm-nexus01.cloud2026.region:8081**;
      5. С рабочего места доступ до веб-интерфейса Nexus должен быть по **https://nexus.cloud2026.region**;
         1. Допустимо использовать самоподписанные сертификаты;
         2. Допустимо (но не обязательно) использовать обратное проксирование для достижения необходимого результата.
3. Настройте менеджер репозиториев Nexus:
   1. Создайте учётную запись в Nexus:
      1. Имя учётной записи: **devops**, пароль: **P@ssw0rd**;
      2. Пользователь **devops** должен обладать правами администратора.
   2. Создайте в Nexus следующие репозитории:
      1. **Docker hosted** репозиторий для хранения Docker-образов доступный на порту **8083**;
      2. **Docker proxy** репозиторий для Docker Hub доступный на порту **8084**;
4. Установите Docker на **vm-dev01**:
   1. Напишите **Dockerfile** для приложения [django-girls-wo-docker](https://disk.yandex.ru/d/OCdg2_2Q52VYaQ) (https://disk.yandex.ru/d/OCdg2\_2Q52VYaQ):
      1. В качестве базового образа используйте **python:3.9-alpine**;
      2. В качестве рабочей директории используйте **/app**;
      3. Для связи используйте порт **8000**.
   2. Соберите образ и запушьте его в **Nexus**.
   3. Запустите приложение на **vm-dev01**, используя образ из Nexus.
      1. С рабочего места доступ до веб-интерфейса приложения должен быть по **http://vm-dev01.cloud2026.region:8000**;
5. Установите Docker на виртуальную машину **vm-gitlab**.
   1. Напишите **compose.yml** для развёртывания настроенного и готового к работе **GitLab CE**:
      1. GitLab должен быть доступен с рабочего места по **https://gitlab.cloud2026.region**;
         1. Допустимо использовать самоподписанные сертификаты;
         2. Допустимо (но не обязательно) использовать обратное проксирование для достижения необходимого результата.
6. Настройте GitLab CE:
   1. Создайте учётную запись в GitLab:
      1. Имя учётной записи: **devops**, пароль: **P@ssw0rd**;
   2. В GitLab создайте репозиторий с именем **django-girls-wo-docker**;
      1. Репозиторий должен принадлежать пользователю devops;
      2. Запушьте **исходный код** приложения с **vm-dev01** в созданный репозиторий.
7. Установите Docker на виртуальную машину **vm-runner01**:
   1. Создайте собственный **GitLab Runner** с **Docker-исполнителем** (Docker executor) и зарегистрируйте его в GitLab для проекта с приложением.
8. Разработайте **pipeline** в GitLab CI:
   1. С этапом **сборки** Docker-образа;
   2. С этапом **загрузки** Docker-образа в GitLab Container Registry;
   3. С этапом **перепубликации** образа в Nexus;
   4. Добавьте в pipeline **job** деплоя приложения на **vm-dev01**.
      1. С рабочего места доступ до веб-интерфейса приложения должен быть по **http://vm-dev01.cloud2026.region:8080**
9. Доработайте **pipeline** в GitLab CI:
   1. При успешном выполнении job деплоя приложения на **vm-dev01**, должно происходить:
      1. Автоматическое разверёртывание виртуальных машин **prod-app01** и **prod-app02** с характеристиками, аналогичными **vm-dev01**;
      2. Автоматическое разверёртывание приложения **django-girls-wo-docker** на виртуальных машинах **prod-app01** и **prod-app02**.
         1. С рабочего места доступ до веб-интерфейса приложения должен быть по **http://prod-app01.cloud2026.region:8000** и **http://prod-app02.cloud2026.region:8000**.
   2. Настройте зависимости между job'ами таким образом, чтобы:
      1. Job'ы развертывания **prod-app01** и **prod-app02** с последующим развёртыванием приложения выполнялись ТОЛЬКО при успешном завершении job'а деплоя на **vm-dev01**
      2. При неудаче деплоя на **vm-dev01** pipeline немедленно прекращал работу.

**Модуль Б. Развертывание приложений в отказоустойчивой масштабируемой инфраструктуре (инвариант)**

*Время на выполнение модуля: 5 часов*

**Задания:**

**1) Подготовка машины Cloud-ADM.**

1. Создайте виртуальный инстанс с именем **Cloud-ADM** и подключите его к необходимым сетям (см. Топология L-3), а также ассоциируйте с ним плавающий IP-адрес.

2. Установите следующие параметры для создаваемого инстанса:

i. Тип виртуальной машины: **2 vCPU, 4 ГБ RAM**;

ii. Размер диска: **30 ГБ**.

3. В качестве операционной системы используйте Альт Рабочая станция, шаблон **alt-workstation-10.4-p10-cloud**.

4. Настройте инстанс для разрешения внешних подключений по протоколу SSH и RDP.

5. Настройте внешнее подключение по SSH сохранив ключевую пару для доступа с вашего локального ПК на рабочем столе с расширением **.pem**:

i. Создайте в PuTTY профиль с именем **Cloud-ADM**;

ii. Реализуйте возможность установления соединения с инстансом **Cloud-ADM** с локального ПК через PuTTY, без необходимости ввода дополнительных параметров;

iii. Для подключения используйте имя пользователя **altlinux** и ранее сохранённую ключевую пару.

6. Настройте внешнее подключение по RDP сохранив на рабочем столе профиль с именем **Cloud-ADM**:

i. Для подключения используйте имя пользователя **altlinux** и пароль **P@ssw0rd**.

**2) Развёртывание облачной инфраструктуры.**

1. Подготовьте сценарий автоматизации развёртывания облачной инфраструктуры:

a) Создайте все необходимые инстансы:

i. Cloud-HA01, Cloud-HA02: **1vCPU, 512 МБ ОЗУ, 10 ГБ размер диска;**

ii. Cloud-DB01, Cloud-DB02 , Cloud-WEB01, Cloud-WEB02: **1vCPU, 1 ГБ ОЗУ, 10 ГБ размер диска;**

iii. В качестве операционной системы используйте Альт Starterkit (оптимизированная сборка под облака), шаблон **alt-p10-cloud-x86\_64**.

b) Создайте все необходимые виртуальные сети:

i. Обеспечьте правильное подключение создаваемых инстансов к соответствующим виртуальным сетям в рамках заданной топологии (см. Топология L-3).

с) Реализуйте безопасный доступ:

i. Разрешите трафик по протоколу **ICMP** для всех инстансов;

ii. Доступ ко всем инстансам должен быть реализован по протоколу SSH только с виртуальной машины **Cloud-ADM** на основе открытых ключей;

d) Создайте балансировщик нагрузки и распределите трафик между инстансами **Cloud-HA01** и **Cloud-HA02**.

i. Ограничьте внешний доступ к балансировщику только протоколами **HTTP** и **HTTPS**.

2. Имена инстансов, виртуальных сетей, и балансировщика нагрузки должны соответствовать именованиям, указанным в Топологии (см. Общая топология и Топология L-3).

3. Реализация скрипта автоматизации:

a) На виртуальной машине **Cloud-ADM** создайте скрипт **deploy-cloudinfra.sh**:

i. В качестве рабочей директории используйте путь **/home/altlinux/bin**;

ii. Скрипт должен использовать файл конфигурации **/home/altlinux/bin/cloudinit.conf** для настройки подключения к облачному провайдеру;

iii. В файле **cloudinit.conf** допускается использование комментариев, поясняющих назначение параметров;

iv. При проверке задания, эксперты могут изменить настройки только в файле **cloudinit.conf**. Другие файлы редактироваться не будут.

b) Скрипт должен быть разработан таким образом, чтобы его можно было выполнять из любой директории без необходимости указания полного пути к исполняемому файлу.

c) Для выполнения задания используйте инструменты для автоматизации развёртывания инфраструктуры **Terraform** или (и) **OpenStack CLI**.

**3) Настройка облачной инфраструктуры.**

1. Подготовьте сценарий автоматизации настройки облачной инфраструктуры:

a) Реализуйте следующий функционал на инстансах **Cloud-HA01** и **Cloud-HA02**:

i. Установить и настроить **HAProxy**:

1. Настроить frontend для прослушивания портов **80** (HTTP) и **443** (HTTPS);

2. Настроить backend с балансировкой между веб-серверами **Cloud-WEB01** и **Cloud-WEB02**, с использованием сети **Internal-Net**;

3. Использовать алгоритм балансировки **round-robin**;

4. При обращении на IP-адреса **Cloud-HA01** и **Cloud-HA02** из сети **Management-Net** с инстанса **Cloud-ADM**, как на порт **80** так и на порт **443**, должно выводиться содержимое веб-приложения развёрнутого на **Cloud-WEB01** и **Cloud-WEB2**, в случае обращения на порт **443** у **Cloud-ADM** не должно возникать никаких проблем с сертификатами (не допускается использование **«Добавления исключений**»).

ii. Установить и настроить **Keepalived**:

1. Режим работы: **Active-Backup**;

2. **Cloud-HA01**: Активный узел (приоритет **100**);

3. **Cloud-HA02**: Резервный узел (приоритет **90**);

4. Настроить **виртуальный IP** (VIP) для отказоустойчивости, используйте первый не занятый IP-адрес из подсети **Internal-Net**;

5. Настроить **аутентификацию** между узлами с использованием общего секретного ключа (в текущей вариации задания VIP — должен просто быть, и корректно переезжать между Активным и Резервным узлом, использовать VIP — для настройки какого-либо иного функционала не требуется).

b) Реализуйте следующий функционал на инстансах **Cloud-WEB01** и **Cloud-WEB02**:

i. Установить **Apache2**;

ii. Настроить **веб-сервера** для обслуживания веб-приложения, обеспечить поддержку **PHP**;

iii. Настроить взаимодействие с базой данных;

iv. Код простого (тестового) веб приложения и подключения к базе данных расположен в Приложении (допускается вносить изменения в исходных код приложения — исключительно только в **$hosts = ['Cloud-DB01', 'Cloud-DB02']** для указания необходимых IP-адресов или доменных имён, в зависимости от способа реализации).

v. В рамках подсети управления (**Management-Net**) при обращении в веб-браузере с инстанса **Cloud-ADM** по доменным именам **cloud-web01.au.team** и **cloud-web02.au.team** должно отрабатывать веб-приложение.

vi. Веб-приложение должно быть доступно как по **HTTP** так и по **HTTPS**, в случае с **HTTPS** используйте самоподписанные сертификаты или иные другие, но с инстанса **Cloud-ADM** должно быть корректное доверие сертификату (не допускается использование **«Добавления исключений**»).

c) Реализуйте следующий функционал на инстансах **Cloud-DB01** и **Cloud-DB02**:

i. Установить **PostgreSQL**;

1. Настроить репликацию между **Cloud-DB01** (Master) и **Cloud-DB02** (Replica);

2. Обеспечить отказоустойчивость: в случае сбоя мастера, реплика должна автоматически становиться новым мастером (учтите, что приложение не вносит никаких данных в базу данных, а только получает их, запрещается изменять учётные данные: Имя БД, пользователя и его пароль — в исходном коде веб-приложения).

ii. Создать тестовую базу данных и пользователя:

1. Создать базу данных с именем **testdb**;

2. Создать пользователя **testuser** с паролем **testpassword**;

3. Назначить пользователю **testuser** права на доступ к базе данных **testdb**.

2. Реализация скрипта автоматизации:

a) На виртуальной машине **Cloud-ADM** создайте скрипт **configure-cloudinfra.sh**:

i. В качестве рабочей директории используйте путь **/home/altlinux/bin**.

b) Скрипт должен быть разработан таким образом, чтобы его можно было выполнять из любой директории без необходимости указания полного пути к исполняемому файлу.

c) Для выполнения задания используйте инструменты автоматизации конфигурации инфраструктуры **Ansible**.

**4) Настройка мониторинга.**

1. Создайте инстанс с именем **Cloud-MON**.

2. Установите следующие параметры для создаваемого инстанса:

i. Тип виртуальной машины: **1 vCPU, 1 ГБ RAM**.

ii. Размер диска: **10 ГБ**.

iii. В качестве операционной системы используйте Альт Starterkit (оптимизированная сборка под облака), шаблон **alt-p10-cloud-x86\_64**

3. Организуйте доступ по SSH до инстанса **Cloud-MON** с инстанса **Cloud-ADM**:

i. Подключение на основе ключевой пары;

ii. Возможность подключения по имени **cloud-mon** из под пользователя **altlinux** без указания дополнительных параметров.

4. Настройте мониторинг с помощью **NodeExporter**, **Prometheus** и **Grafana** в **Docker**:

i. Создайте в домашней директории пользователя **altlinux** файл **monitoring.yml** для Docker Compose;

ii. Используйте контейнеры **NodeExporter**, **Prometheus** и **Grafana** для сбора, обработки и отображения метрик;

iii. Настройте **Dashboard** в **Grafana**, в котором будет отображаться загрузка CPU, объём свободной оперативной памяти и места на диске.

iv. Интерфейс Grafana должен быть доступен по имени **grafana.au.team** на порту **3000** с инстанса **Cloud-ADM,** из под пользователя **admin** с паролем **P@ssw0rd**;

v. Добавьте инстансы **Cloud-ADM,** **Cloud-BACKUP01** и **Cloud-BACKUP02** в **Dashboard** для мониторинга.

**5) Настройка резервного копирования.**

1. Создайте инстансы с именами **Cloud-BACKUP01** и **Cloud-BACKUP02** .

2. Установите следующие параметры для создаваемых инстансов:

i. Тип виртуальной машины: **1 vCPU, 1 ГБ RAM**.

ii. Размер диска: **10 ГБ**.

iii. Дополнительный том: **5 ГБ**.

iv. В качестве операционной системы используйте Альт Starterkit (оптимизированная сборка под облака), шаблон **alt-p10-cloud-x86\_64**.

3. На инстансы **Cloud-ADM, Cloud-BACKUP01** и **Cloud-BACKUP02** установите Кибер Бекап 17 версии:

i. Сервер управления необходимо развернуть на инстансе **Cloud-ADM**;

ii. Настроить организацию (отдел) **AU\_Team**;

iii. Настройте пользователя **au-admin** с паролем **P@ssw0rd** и правами администратора на сервере управления Кибер Бекап

iv. В качестве узлов хранилища используйте инстансы **Cloud-BACKUP01** и **Cloud-BACKUP02**.

4. Настройте устройства хранения на базе **Cloud-BACKUP01** и **Cloud-BACKUP02**:

i. Подключите в качестве устройства хранения блочное устройство **sdb** в формате **xfs**;

ii. Устройство должно быть при монтировано в папку **/backups**;

iii. Наименование хранилищ **BACKUP01** и **BACKUP02** в зависимости от размещения на соответствующем инстансе.

5. Создайте план резервного копирования для инстанса **Cloud-ADM**, а именно для домашней директории пользователя **altlinux.**

i. В качестве узла хранения используйте **BACKUP01**;

ii. Выполните резервное копирование для домашней директории пользователя **altlinux.**

6. Создайте план репликации резервной копии домашней директории пользователя **altlinux.**

i. В качестве узла хранения репликации используйте **BACKUP02;**

ii. Выполните репликацию созданной резервной копии.

**6) Завершение работы**

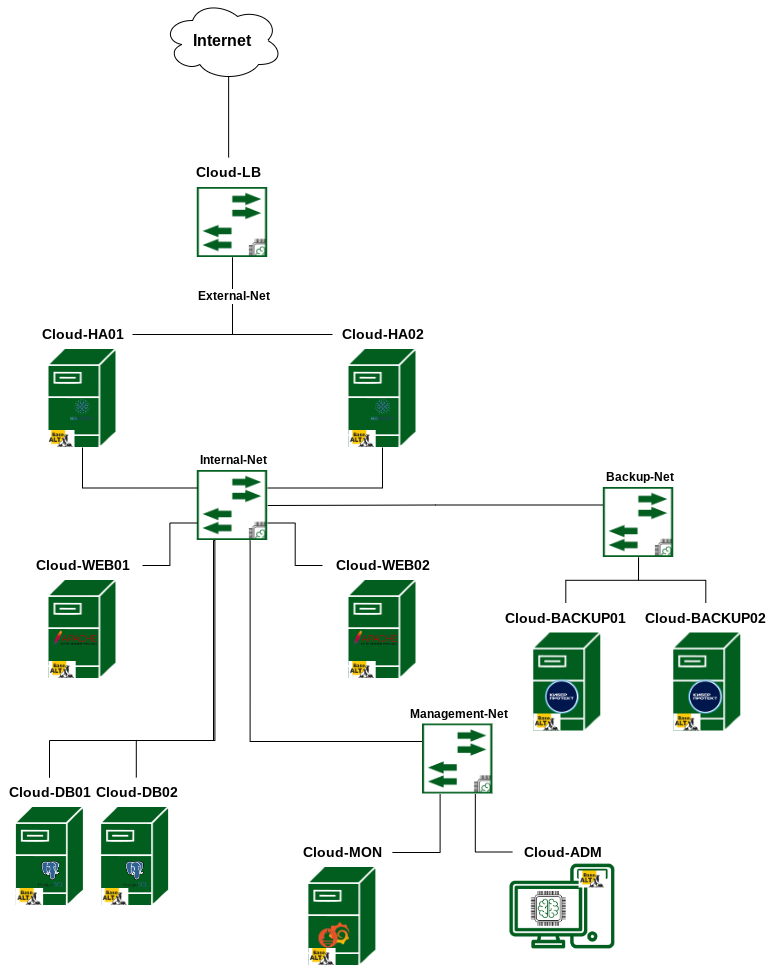
1. По окончании рабочего времени освободите ресурсы облачного провайдера, использованные для автоматически созданных объектов.

2. Удалите все автоматически созданные виртуальные машины, сети, объекты и другие ресурсы.

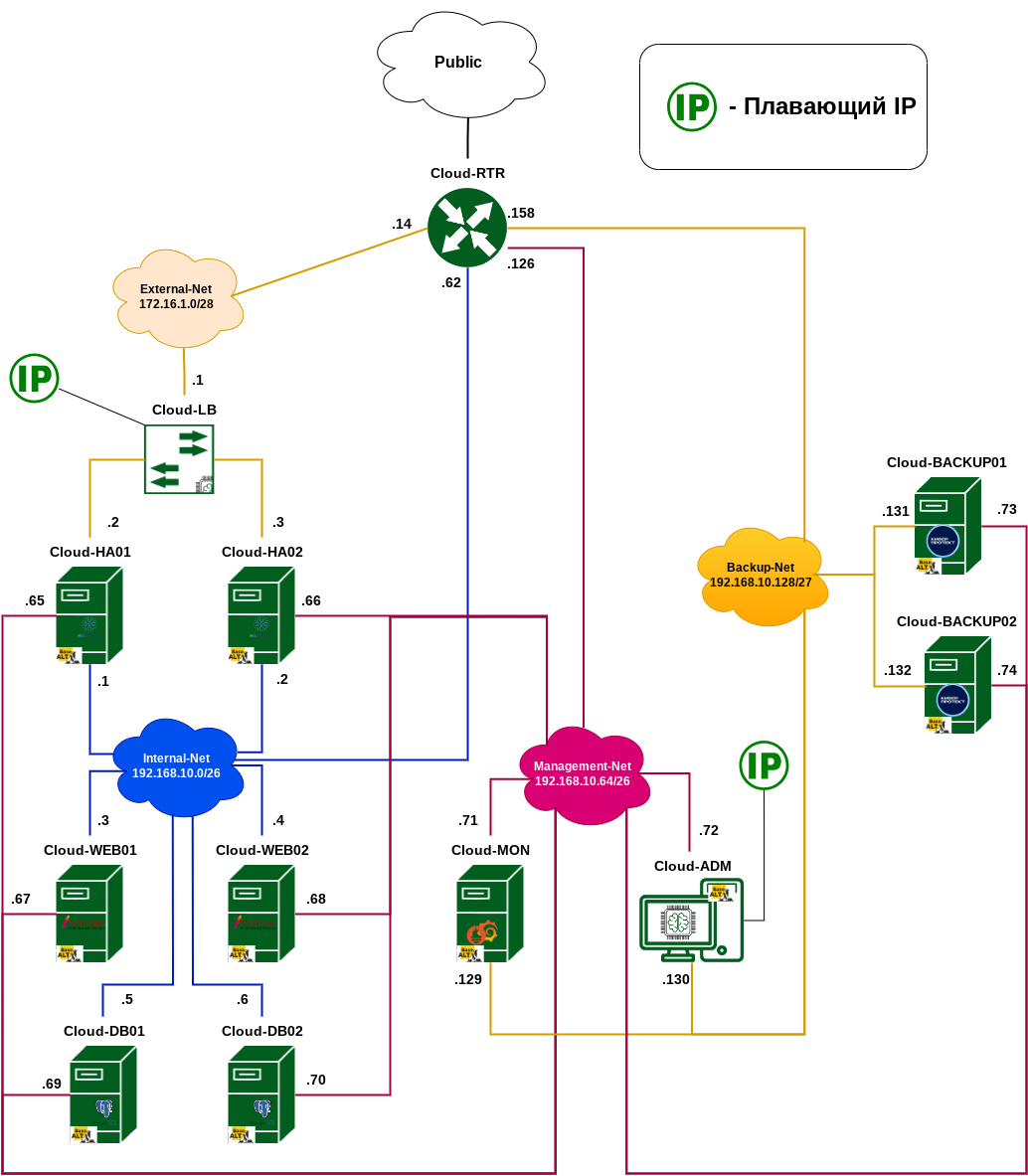
3. **Внимание:** НЕ удаляйте инстансы: **Cloud-ADM, Cloud-MON, Cloud-BACKUP01, Cloud-BACKUP02**, а также ресурсы, необходимые для их функционирования: виртуальный маршрутизатор **Cloud-RTR**, виртуальные сети: **Backup-Net, Management-Net**, плавающий IP-адрес ассоциируемый с инстансом **Cloud-ADM**

4. **Важно:** если в облачной инфраструктуре останутся объекты, кроме тех, которые **перечислены** или создаются по умолчанию, проверка выполнения задания не будет проводиться.

**Общая топология:**



**Топология L-3:**



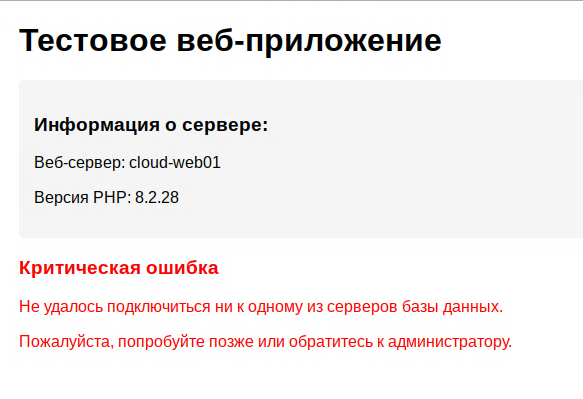
**Приложение:**

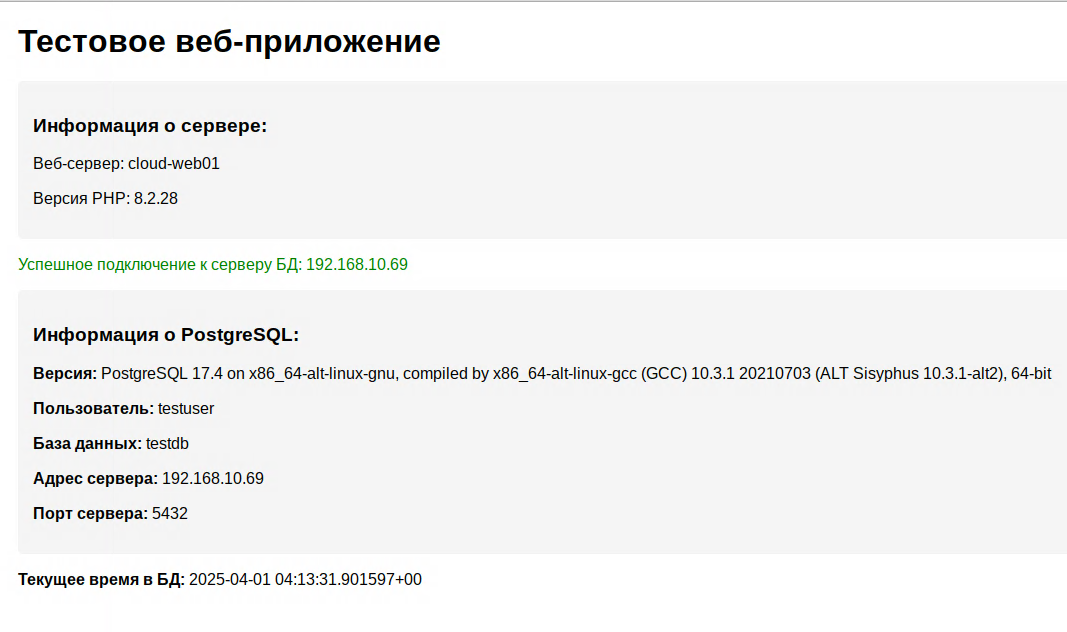
|  |
| --- |
| <?php  // Заголовок HTML-страницы  echo "<!DOCTYPE html>  <html lang='ru'>  <head>  <meta charset='UTF-8'>  <title>Тестовое приложение</title>  <style>  body { font-family: Arial, sans-serif; margin: 20px; }  .success { color: green; }  .error { color: red; }  .info { background-color: #f5f5f5; padding: 15px; border-radius: 5px; }  </style>  </head>  <body>  <h1>Тестовое веб-приложение</h1>";  // Вывод имени веб-сервера  echo "<div class='info'>  <h3>Информация о сервере:</h3>  <p>Веб-сервер: " . gethostname() . "</p>  <p>Версия PHP: " . phpversion() . "</p>  </div>";  // Параметры подключения к базе данных  $hosts = ['Cloud-DB01', 'Cloud-DB02']; // Основной и резервный серверы  $port = '5432';  $dbname = 'testdb';  $user = 'testuser';  $password = 'testpassword';  // Попытка подключения к каждому серверу БД по очереди  $connected = false;  foreach ($hosts as $host) {  $conn\_string = "host=$host port=$port dbname=$dbname user=$user password=$password";  $conn = pg\_connect($conn\_string);    if (!$conn) {  echo "<div class='error'><p>Не удалось подключиться к серверу БД $host: " . pg\_last\_error() . "</p></div>";  continue;  }    $connected = true;  echo "<div class='success'><p>Успешное подключение к серверу БД: $host</p></div>";    // Получение информации о сервере PostgreSQL  $result = pg\_query($conn, "SELECT version(), current\_user, current\_database(), inet\_server\_addr(), inet\_server\_port()");  if (!$result) {  echo "<div class='error'><p>Ошибка при выполнении запроса: " . pg\_last\_error($conn) . "</p></div>";  } else {  $row = pg\_fetch\_assoc($result);  echo "<div class='info'>  <h3>Информация о PostgreSQL:</h3>  <p><strong>Версия:</strong> " . htmlspecialchars($row['version']) . "</p>  <p><strong>Пользователь:</strong> " . htmlspecialchars($row['current\_user']) . "</p>  <p><strong>База данных:</strong> " . htmlspecialchars($row['current\_database']) . "</p>  <p><strong>Адрес сервера:</strong> " . htmlspecialchars($row['inet\_server\_addr']) . "</p>  <p><strong>Порт сервера:</strong> " . htmlspecialchars($row['inet\_server\_port']) . "</p>  </div>";  pg\_free\_result($result);  }    // Пример выполнения простого запроса  $result = pg\_query($conn, "SELECT NOW() AS current\_time");  if ($result) {  $row = pg\_fetch\_assoc($result);  echo "<p><strong>Текущее время в БД:</strong> " . htmlspecialchars($row['current\_time']) . "</p>";  pg\_free\_result($result);  }    // Закрытие соединения  pg\_close($conn);  break; // Прерываем цикл после успешного подключения  }  if (!$connected) {  echo "<div class='error'><h3>Критическая ошибка</h3>  <p>Не удалось подключиться ни к одному из серверов базы данных.</p>  <p>Пожалуйста, попробуйте позже или обратитесь к администратору.</p></div>";  }  // Завершение HTML-страницы  echo "</body></html>";  ?> |

**Пример работоспособности веб-приложения:**

* + Демонстрация работы кода, если нет поддержки PHP:

****

* + Демонстрация работы кода, если есть поддержка PHP, но нет подключения к БД:
  + Демонстрация работы кода, если есть поддержка и подключение к БД:



**Модуль В. Настройка объектного хранилища для централизованного управления данными и обеспечения отказоустойчивости хранения (Вариатив)**

*Время на выполнение модуля:3 часа*

**Задания:**

**1. Разверните распределенное объектное хранилище MinIO на четырех серверах: srv1-cod, srv2-cod, srv3-cod и srv4-cod.**

a. Используйте виртуальные машины srv1, srv2, srv3 и srv4 в качестве хостов кластера.

b. Установите и настройте сервер MinIO в distributed mode.

c. В качестве бэкенда хранения на каждом сервере используйте несколько отдельных дисков:

i. На srv1-cod: диски /dev/sdb, /dev/sdc, /dev/sdd.

ii. На srv2-cod: диски /dev/sdb, /dev/sdc, /dev/sdd.

iii. На srv3-cod: диски /dev/sdb, /dev/sdc, /dev/sdd.

iv. На srv4-cod: диски /dev/sdb, /dev/sdc, /dev/sdd.

v. Создайте на каждом диске файловую систему xfs с оптимизацией для больших файлов, используйте опции -i size=512 -n size=8192.

vi. Смонтируйте файловые системы:

1. На srv1-cod: в /mnt/minio-data-node1-1, /mnt/minio-data-node1-2, /mnt/minio-data-node1-3.
2. Аналогично для других серверов (/mnt/minio-data-nodeX-Y).

vii. Настройте автоматическое монтирование при загрузке системы на всех серверах с использованием UUID в /etc/fstab.

**2. Выполните конфигурацию распределенного кластера MinIO.**

a. Запустите службу MinIO на всех четырех серверах как часть единого кластера.

b. Используйте стандартный порт 9000 для API на каждом узле.

c. Используйте стандартный порт 9001 для веб-интерфейса Console на каждом узле.

d. Установите общие учетные данные root пользователя для всего кластера:

i. Имя пользователя: minioadmincluster

ii. Пароль: P@ssw0rd2025

e. Настройте MinIO на каждом узле для использования соответствующих каталогов в качестве хранилища /mnt/minio-data-node1-{1,2,3} на srv1-cod.

f. Обеспечьте повышенную отказоустойчивость кластера:

i. Потеря до двух узлов не должна приводить к потере данных и должна позволять продолжить работу в degraded mode.

ii. Настройте автоматическое восстановление данных при возврате узла.

iii. Внедрите bucket replication между двумя подкластерами, srv1+srv2 и srv3+srv4 для гео-распределения.

**3. Настройте доступ к распределенному MinIO.**

a. Кластер MinIO должен быть доступен по внутренним именам хостов:

i. srv1-cod.cod.ssa2025

ii. srv2-cod.cod.ssa2025

iii. srv3-cod.cod.ssa2025

iv. srv4-cod.cod.ssa2025

b. Веб-интерфейс Console должен быть доступен на каждом узле:

i. srv1-cod.cod.ssa2025:9001

ii. srv2-cod.cod.ssa2025:9001

iii. srv3-cod.cod.ssa2025:9001

iv. srv4-cod.cod.ssa2025:9001

c. API должно быть доступно на каждом узле:

i. srv1-cod.cod.ssa2025:9000

ii. srv2-cod.cod.ssa2025:9000

iii. srv3-cod.cod.ssa2025:9000

iv. srv4-cod.cod.ssa2025:9000

**4. Интеграция с Центром Сертификации и обеспечение безопасности.**

a. Выпустите отдельные сертификаты для каждого узла кластера с использованием центра сертификации на `srv1-cod` для:

i. srv1-cod.cod.ssa2025

ii. srv2-cod.cod.ssa2025

iii. srv3-cod.cod.ssa2025

iv. srv4-cod.cod.ssa2025

b. Настройте MinIO на всех узлах для использования соответствующих выпущенных сертификатов для шифрования всех соединений (TLS).

c. Настройте MinIO для использования только TLS-соединений для API и Console.

**5. Проверка функциональности.**

a. Создайте тестовый бакет с именем test-cluster-bucket с включенной версионностью и политикой retention (7 дней).

b. Загрузите несколько тестовых объектов минимум 10 в созданный бакет.

c. Убедитесь, что объект доступен через любой узел кластера.

d. Проверьте доступность веб-интерфейса Console через TLS.

e. Проверьте доступность API через TLS.

2. СПЕЦИАЛЬНЫЕ ПРАВИЛА КОМПЕТЕНЦИИ*[[1]](#footnote-1)*

1. Конкурсантам при выполнении всех модулей можно использовать интернет-ресурсы, за исключением:

– Систем контроля версий

– Общения посредством форумов/мессенджеров/иных средств коммуникации – видеохостингов

Средства, требующие авторизацию любой формы

1. Конкурсанты имеют право задавать уточняющие вопросы экспертам (кроме эксперта наставника) и вправе получить ответ, если вопрос не предполагает получения информации о реализации конкретной технологии

2.1. Личный инструмент конкурсанта

* Клавиатура, мышь не программируемые
* Средства защиты слуха и глаз

2.2. Материалы, оборудование и инструменты, запрещенные на площадке

Мобильные устройства, устройства фото-видео фиксации, носители информации.

1. *Указываются особенности компетенции, которые относятся ко всем возрастным категориям и чемпионатным линейкам без исключения.* [↑](#footnote-ref-1)