|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема 1 | Тема 2 | Тема 3 | Тема 4 | Всего |
| 20 | 110 | 44 | 26 | 200 |

Тема 1. Организация автоматического тестирования ПО

1. Что такое CI (Continuous Integration)?
	1. ПО для автоматической интеграции кода
	2. принцип разработки ПО
	3. система контроля версий
	4. модуль автоматического тестирования ПО
2. CI включает в себя:
	1. только автоматическую сборку, тестирование необходимо проводить вручную
	2. автоматическую сборку и тестирование
	3. только автоматическое тестирование, сборку необходимо проводить вручную
	4. ничего из перечисленного
3. CDeploy - это:
	1. подход к разработке, в рамках которого выпуск приложения надлежащего качества гарантирован в любое время
	2. подход к разработке, при котором функциональные возможности предоставляются посредством автоматизированного развёртывания
	3. подход к разработке с использованием автоматизированного тестирования для получения немедленной обратной связи
	4. ничего из перечисленного
4. Выберите основные характеристики CDelivery:
	1. разработка проводится итерациями неопределенного срока
	2. выпуск ПО надлежащего качества гарантирован в любое время
	3. передача ПО происходит автоматически
	4. все перечисленные
5. Статистический анализ кода используется для:
	1. проверки стилей форматирования
	2. определения ошибок без выполнения кода
	3. тестирования выполняемой программы
	4. ничего из перечисленного
6. Юнит-тестирование используется для
	1. тестирования работы продукта
	2. тестирования отдельных модулей продукта
	3. проверки модульной структуры продукта
	4. ничего из перечисленного
7. Полный запуск продукта и проверка общей корректности его работы используется при:
	1. динамическом анализе
	2. интеграционном тестировании
	3. юнит-тестировании
	4. всё перечисленное
8. Стресс-тестирование является одним из видов дополнительного тестирования, заключающегося в проверке
	1. соответствия продукта функциональным требованиям
	2. морального состояния пользователей продукта
	3. производительности системы под нагрузкой
	4. всё перечисленное
9. Смысл юнит-тестирования заключается в:
	1. покрытии тестами каждой нетривиальной функции и метода
	2. проверке соответствия написанного кода стандартам
	3. проверки взаимодействия частей приложения
	4. всё вышеперечисленное
10. Статическое тестирование предназначено для:
	1. проверки правильной работы каждой отдельной части приложения
	2. выявления ошибок, проявляющихся случайным образом
	3. тестирования приложения без запуска кода
	4. всё вышеперечисленное
11. Интеграционное тестирование предназначено для:
	1. покрытия тестами каждой нетривиальной функции и метода
	2. проверки работоспособности приложения в целом в условиях, приближенных к реальным
	3. выявления в коде синтаксических ошибок
	4. всё вышеперечисленное
12. Что из нижеперечисленного не является целью тестирования?
	1. проверка соответствия ПО заявленным требованиям
	2. обеспечение уверенности в качестве ПО
	3. повышение стоимости разрабатываемого ПО
	4. поиск очевидных ошибок в ПО, перед его поставкой конечному пользователю
13. Какое из следующих утверждений о тестировании является ошибочным?
	1. Если во время тестирования не было выявлено проблем - значит их нет
	2. Полное тестирование с использованием всех входных комбинаций данных, результатов и предусловий физически невыполнимо
	3. Если повторять те же тестовые сценарии снова и снова, в какой-то момент этот набор тестов перестанет выявлять новые дефекты.
	4. Тестирование только снижает вероятность наличия дефектов, которые находятся в программном обеспечении, но не гарантирует их отсутствия.
14. IaC - это?
	1. способ создания GUI приложения, в котором интерактивные элементы интерфейса создаются и располагаются посредством написания кода, а не использования графической среды
	2. подход для управления и описания инфраструктуры через конфигурационные файлы или интерактивное взаимодействие, а не через ручное редактирование конфигураций на серверах.
	3. подход для управления и описания инфраструктуры через ручное редактирование конфигураций на серверах
	4. всё вышеперечисленное
15. Continuous testing - это
	1. постоянное выполнение одного и того же теста, пока он не обнаружит дефект в ПО
	2. Выполнение тестирования приложения в течение определенного периода после выпуска его релизной версии
	3. выполнение автоматизированных тестов в рамках конвейера поставки ПО для получения немедленной обратной связи
	4. ничего из вышеперечисленного
16. Что не является частью функционального тестирования?
	1. проверка работоспособности приложения в штатных условиях
	2. проверка на соответствие приложения функциональным требованиям
	3. наиболее точная имитация поведения пользователя
	4. проверка, что приложение запускается
17. Что из нижеперечисленного соответствует определению нагрузочного тестирования?
	1. тестирование кода отдельных блоков кода
	2. проверка запуска приложения и общей корректности его работы
	3. проверка продукта на соответствие функциональным требованиям
	4. проверка метрик производительности системы под нагрузкой
18. Что из нижеперечисленного не соответствуем принципам организации автоматического тестирования?
	1. Организация процесса тестирования таким образом, что в случае возникновения ошибки информация о ней будет максимально быстро доставлена до того, кто её вызвал
	2. Тесты пишутся так, чтобы их не надо было править после выявления ошибок и запускать повторно
	3. Информация о запуске тестов максимально закрыта, доступна только информация об отработавших тестах, результате тестирования и событиях, приведших к запуску тестирования
	4. Измеряется время работы тестов, ставится ограничение на длительность тестирования
19. Что такое “дымовой тест” (smoke test)?
	1. Минимальный набор тестов на явные ошибки (ошибка сборки, соединения с БД, загрузка конфигурационных файлов,...)
	2. Тест, прерывающий цепочку успешно выполняемых автоматических тестов
	3. Набор тестов, максимально полно покрывающий все возможные способы взаимодействия пользователя с приложением
	4. Тест, не влияющий на результат проведения тестирования
20. Что из нижеперечисленного не является частью процесса Continuous Integration?
	1. Добавление нового коммита в репозиторий
	2. Сборка/компиляция проекта
	3. Тестирование
	4. Копирование собранного проекта на рабочем сервере

Тема 2. Инструменты автоматизации настройки среды

1. В чём заключается разница между виртуализацией и контейнеризацией?
	1. Виртуализация позволяет запустить несколько операционных систем на одном физическом сервере. Контейнеризация работает на одной и той же операционной системе, в которой приложения упакованы в контейнеры и запускаются на одном сервере/виртуальной машине.
	2. Контейнеризация позволяет запустить несколько операционных систем на одном физическом сервере. Виртуализация работает на одной и той же операционной системе, в которой приложения упакованы в контейнеры и запускаются на одном сервере/виртуальной машине.
	3. Это названия одного и того же явления
	4. Виртуализация - процесс имитирования роботы нескольких процессорных ядер на одном физическом. Контейнеризация - процесс замещения используемой процессом оперативной памяти на долговременную (дисковую)
2. Что не соответствует определению изоляции?
	1. общие ресурсы с ОС (ядро)
	2. своё окружение (файлы, переменные среды)
	3. общее с ОС дерево процессов
	4. ограничения на диск, ОЗУ, сеть
3. Для чего руководства по установке и настройке Docker предлагают добавить пользователя в группу docker?
	1. Для защиты системных файлов хоста от действий пользователя, совершаемых в контейнере
	2. Для предоставления управления средствами Docker непривилегированному пользователю
	3. Для предоставлению пользователю возможности создания томов и сетей, а также проброса портов из контейнера
	4. Всё вышеперечисленное
4. Выберете верное утверждение касательно запуска контейнера Docker в контейнере Docker:
	1. Подобной возможности запуска нет
	2. Такой возможности нет, но можно присоединить один контейнер к дереву процессов и ресурсом другого при запуске
	3. Такая возможность есть, однако рекомендуется делать привязку /var/run/docker.sock хоста к одноимённому файлу внешнего контейнера, для предоставления возможности запуска контейнеров на уровне хоста
	4. Такая возможность существует и приветствуется в организации инфраструктуры
5. Какие состояния из представленных ниже не являются состояниями контейнера Docker?
	1. Created
	2. Running
	3. Started
	4. Exited
	5. Paused
	6. Stop
	7. Disabled
6. Что обозначает состояние контейнера Dead?
	1. контейнер завершил работу, но таким образом, что его нельзя перезапустить (ресурсы заблокированы)
	2. контейнер завершил работу
	3. контейнер находится в процессе удаления
	4. контейнер завершил свою работу из-за ошибки в базовом процессе
7. Что обозначает состояние контейнера Paused?
	1. контейнер завершил работу, но таким образом, что его нельзя перезапустить (ресурсы заблокированы)
	2. контейнер приостановлен
	3. контейнер находится в процессе удаления
	4. контейнер завершил свою работу из-за ошибки в базовом процессе
8. Что обозначает состояние контейнера Exited?
	1. контейнер завершил работу, но таким образом, что его нельзя перезапустить (ресурсы заблокированы)
	2. контейнер завершил работу
	3. контейнер находится в процессе удаления
	4. контейнер завершил свою работу из-за ошибки в базовом процессе
9. Какое состояние контейнера свидетельствует о том, что он завершил свою работу?
	1. Stopped
	2. Paused
	3. Disabled
	4. Exited
	5. Deleted
10. Как долго живёт контейнер?
	1. До перезапуска операционной системы
	2. До выключения демона docker.service
	3. До выполнения команды docker rm <имя\_контейнера>
	4. До завершения процесса с PID=1 в контейнере
11. Потеряются ли данные при случайном завершении работы контейнера? Если да то какие?
	1. Никакие данные не потеряются
	2. Потеряются только данные основного процесса, хранимые в оперативной памяти
	3. Потеряются данные всех процессов, хранимые в оперативной памяти
	4. Потеряются все данные
12. Какой registry используется в докере по умолчанию?
	1. localhost
	2. hub.docker.com
	3. Registry
	4. github.com
13. Как использовать registry, отличный от значения по умолчанию?
	1. изменить конфигурацию Docker
	2. использовать namespace в названии образов
	3. получить токен соответствующего registry
	4. использовать ssh-соединение с хостом registry
14. За что отвечает серверная часть Docker?
	1. Управление томами, сетями, контейнерами и образами
	2. Загрузка и выгрузка образов в Docker Registry
	3. Подключение к работающему контейнеру
	4. Распределение вычислительной нагрузки между элементами сети построенной на базе docker-compose
15. Какой образ будет загружен при выполнении команды docker pull если тэг образа не был указан?
	1. Самый первый из загруженных в репозиторий
	2. Команда завершится с ошибкой «Укажите тэг»
	3. С тэгом latest
	4. Любой из доступных
16. Какую технологию использует docker для предоставления изолированных рабочих пространств?
	1. namespaces
	2. UnionFS
	3. Virtualizing
	4. CLI
17. Что такое docker registry?
	1. Система авторизации, позволяющая использовать приватные образы
	2. Система подписи контейнеров
	3. Подсистема docker, отвечающая за создание и управление контейнерами
	4. Хранилище публичных и/или приватных образов
18. Основным отличием использования тома (volume) от монтирования каталога в контейнер является:
	1. не поддерживает общий доступ к файлам из нескольких контейнеров
	2. является рекомендуемым способом хранения данных, поскольку предполагается, что другие программы не должны иметь доступа к файлам внутри тома, а также том может физически располагаться на удалённом сервере, а не на хосте
	3. не рекомендуется к использованию, из-за того что позволяет изменять пользовательские файлы пользователя на хосте
	4. том - область оперативной памяти, которой ограничены процессы запускаемые в контейнере, а примонтированный коталог - часть файловой системы хоста, в которой контейнер хранит свои файлы
19. Для чего используется Dockerfile?
	1. Для описания слоёв, из которых состоит образ
	2. Для задания параметров, с которыми запускается контейнер
	3. Для настройки демона docker
	4. Для создания контейнера без использования образа
20. Как взаимосвязаны контейнер и образ?
	1. На основе контейнера создаётся образ
	2. Контейнер состоит из описывающих этапы его сборки образов
	3. На основе образа создаётся контейнер
	4. Образ - это слепок файловой системы, с которой работал контейнер
21. В чём заключается различие между докер репозиторием и докер реестром?
	1. Репозиторий – набор образов, обладающих одинаковыми именами, но разными тэгами; реестр – сервис хранения репозиториев
	2. Это синонимы
	3. Репозиторий – сервис хранения реестров; реестр – набор образов, обладающих одинаковыми именами, но разными тэгами
	4. Репозиторий – сетевое хранилище образов; реестр – список пользователей, обладающих правами на работу с образами из конкретного репозитория
22. Что такое docker-клиент?
	1. Часть docker, управляющая образами, контейнерами и т.п.
	2. Пользователь, использующий Enterprise версию docker
	3. Часть docker, устанавливаемая на рабочий компьютер пользователя
	4. Часть docker, предоставляющая пользователю интерфейс управления образами, контейнерами и т.п.
23. Какой образ будет загружен при выполнении команды docker pull archlinux ?
	1. archlinux:base
	2. archlinux:base-devel
	3. archlinux:base-devel-20220130.0.46058
	4. archlinux:latest
24. Что обозначается термином «тэг» применительно к докер-образам?
	1. метка-идентификатор, добавляемая к скачиваемым образам и используемая при выделении разных виртуальных каналов данных в одном физическом канале
	2. Команда, описывающая шаг настройки контейнера в docker-файле
	3. метка-идентификатор, привязываемая к определённой версии образа
	4. метка-идентификатор, используемая для поиска образа по категориям
25. Какой образ будет загружен в результате выполнения команды docker pull debian –a?
	1. debian:unstable-slim
	2. debian:testing-slim
	3. debian:stretch-slim
	4. debian:stable-slim
	5. Все вышеперечисленные
	6. Ничего из вышеперечисленного
26. Какая информация будет выведена в результате выполнения команды docker pull kubuntu -q?
	1. Репозиторий, из которого выкачивается образ
	2. Время загрузки образа
	3. Ошибка
	4. Хэш образа
27. Флаг --platform <…> команды docker pull…
	1. задаёт платформу, если сервер поддерживает мультиплатформенность
	2. проверяет, поддерживается ли заданная флагом платформа образом
	3. позволяет указать архитектуру процессора для загрузки образа, оптимизированного под неё
	4. указывает, сколько слоёв из образа должно быть загружено
28. Как выглядит команда загрузки образа me/my\_image с сервера 10.183.225.205:5000?
	1. docker pull --ip 10.183.225.205 --port 5000 me/my\_image
	2. docker pull 10.183.225.205:5000/me/my\_image
	3. docker pull me/my\_image --site 10.183.225.205:5000
	4. docker pull –path 10.183.225.205:5000@me/\_my\_image
	5. docker pull 10.183.225.205:5000@me/my\_image
29. Команда docker run используется для:
	1. вывода информации об образе
	2. создания и запуска контейнера
	3. вывода списка контейнеров
	4. сборки образа контейнера
30. Какие флаги необходимо добавить к команде docker run debian sh чтобы получить доступ к консоли внутри контейнера? (подсказка: необходимо выделить псевдо-терминал и включить перенаправление потока ввода)
	1. –init
	2. –interactive
	3. –stdout
	4. –tmpfs
	5. –tty
	6. –stdin
31. Что делает флаг --name команды docker run?
	1. Запускает контейнер от имени другого пользователя
	2. Изменяет значение переменной USER в контейнере
	3. Присваивает контейнеру имя, отображаемое в выводе команды docker ps
	4. Указывает имя образа, на основе которого будет создан контейнер
32. Что делает флаг –u команды docker run?
	1. Запускает контейнер от имени другого пользователя
	2. Изменяет значение переменной USER в контейнере
	3. Присваивает контейнеру имя, отображаемое в выводе команды docker ps
	4. Указывает имя образа, на основе которого будет создан контейнер
33. Какая команда запустит контейнер в директории /etc/default/?
	1. docker run –workdir etc/default/ debian
	2. docker run –pwd etc/default/ debian
	3. docker run –workdir /etc/default/ debian
	4. docker run –cd /etc/default/ debian
34. Контейнер был создан с помощью команды docker –d –P 5000:4000 apach. Какой порт будет открыт на хост-системе?
	1. 5000
	2. 4000
	3. Будет открыт случайный порт
	4. Контейнер не может быть создан данной командой
35. С помощью какой команды можно передать внутрь контейнера переменную окружения VAR=TEST?
	1. docker run –e VAR=TEST nginx
	2. docker run –v VAR TEST nginx
	3. docker run –e $VAR=TEST nginx
	4. docker run –p VAR:TEST nginx
36. Контейнер был запущен командой docker run -d dance sleep 1000000. Завершение какого процесса приведёт к закрытию контейнера?
	1. PID=1 | python3 -c while True: print(«\033[6;34mLets dance!\033[0m»
	2. PID=19 | python3 -c while True: print(«\033[6;34mLets dance!\033[0m»)
	3. PID=1 | sleep 1000000
	4. PID=19 | sleep 1000000

Примечание: образ dance задан конфигурационным файлом



1. Какая команда запустит контейнер с примонтированной папкой /data из хост системы на /not\_access контейнера в режиме только для чтения?
	1. docker run -v /data*:*/not\_access:ro ubuntu
	2. docker run --read-only --mount /data:/not\_access ubuntu
	3. docker run --mount /data:/not\_access ubuntu
	4. docker run -v /data:not\_access:ro ubuntu
2. Вы располагаетесь в домашней директории (её содержимое представлено на изображении). Где в файловой системе хоста будет создан файл file.txt после выполнения команды docker run -v dir:/app --name ubuntu ubuntu touch /app/file.txt? (Примечание: домашняя директория пользователя располагается в /home/$USER)



* 1. /home/USER/app/file.txt
	2. /var/lib/docker/volumes/app/\*\*/file.txt
	3. /home/oblomov/app/file.txt
	4. /home/USER/dir/file.txt
	5. /var/lib/docker/volumes/dir/file.txt
	6. /home/oblomov/dir/\*\*/file.txt
	7. /app/file.txt
	8. /dir/file.txt
	9. Файл создан не будет
1. Вы располагаетесь в домашней директории (её содержимое представлено на изображении). Где в файловой системе хоста будет создан файл file.txt после выполнения команды docker run -v /home/$USER/dir:/app -name ubuntu ubuntu touch /app/file.txt?



* 1. /home/USER/app/file.txt
	2. /var/lib/docker/volumes/app/\*\*/file.txt
	3. /home/oblomov/app/file.txt
	4. /home/USER/dir/file.txt
	5. /var/lib/docker/volumes/dir/file.txt
	6. /home/oblomov/dir/\*\*/file.txt
	7. /app/file.txt
	8. /dir/file.txt
	9. Файл создан не будет
1. Укажите верное утверждение об различиях между командами docker run и docker exec:
	1. Эти команды взаимозаменяемы
	2. docker run используется для создания контейнера и последующего запуска команды из-под него, а docker exec – для запуска команды в рамках запущенного контейнера
	3. docker exec используется для создания контейнера и последующего запуска команды из-под него, а docker run – для запуска команды в рамках запущенного контейнера
	4. Это принципиально различные команды – одна используется для создания контейнера, а другая – для работы с ним
2. Последовательностью каких команд может быть заменена команда docker run?
	1. docker pull … && docker create … && docker start …
	2. docker pull … && docker start … && docker exec …
	3. docker start … && docker exec …
	4. Нет такой последовательности
3. Дан вывод команды docker ps. Что будет выведено на экран в результате выполнения команды docker exec dazzling\_mendeleev ls –lA /home



* 1. Ничего
	2. total 0
	3. Сообщение об ошибке
	4. Список пользователей хост-системы
1. С помощью какой команды можно запустить команду внутри запущенного контейнера my\_nginx, созданного на основе образа nginx, с переменной окружения VAR=TEST?
	1. docker exec –e VAR=TEST my\_nginx command
	2. docker exec –e VAR=TEST nginx command
	3. docker run –e $VAR=TEST nginx command
	4. docker run -e VAR=TEST my\_nginx command
2. Какая команда выполнит команду ln -s d внутри запущенного контейнера my\_deb, созданного на основе образа debian, в директории /etc/default/?
	1. docker exec --workdir /etc/default/ debian
	2. docker exec -e PWD=/etc/default/ my\_deb
	3. docker exec --workdir /etc/default/ my\_deb
	4. docker exec /etc/default/ my\_deb
3. Что делает флаг –u команды docker exec?
	1. Задаёт пользователя, от имени которого будет выполняться команда
	2. Позволяет выбрать контейнер, в котором будет запускаться команда, по имени, а не по id
	3. Заменяет отображаемое в выводе команды docker ps имя контейнера
	4. Сбрасывает все изменения, произошедшие в контейнере после его запуска.
4. Что делает связка параметров -it команды docker exec?
	1. Позволяет выбрать контейнер, в котором будет запускаться команда, по имени, а не по id
	2. Запускает команду в интерактивном режиме
	3. Заменяет отображаемое в выводе команды docker ps имя контейнера
	4. Сбрасывает все изменения, произошедшие в контейнере после его запуска
5. За что отвечает флаг --detach-keys команды docker exec?
	1. Запрещает контейнеру воспринимать определённые символы из потока ввода
	2. Заменяет определённую последовательность символов в потоке ввода на другую
	3. Позволяет переопределить последовательность символов, необходимых для отсоединения от контейнера без остановки его главного процесса (процесса с PID=1)
	4. Позволяет перехватывать определённые сочетания клавиш и передавать их непосредственно в контейнер, а не в командную оболочку, в которой была запущена команда
6. Что делает команда docker ps –a?
	1. Показывает список всех контейнеров
	2. Показывает список всех запущенных контейнеров
	3. Показывает список всех образов
	4. Показывает список процессов, запущенных во всех контейнерах
7. Необходимо получить список container\_id запущенных контейнеров. Выберите команду, с помощью которой это можно сделать:
	1. docker ps -a
	2. docker ps -q
	3. docker ps ids
	4. ничего из перечисленного
8. Дан вывод команды docker ps. Какой порт открыт у контейнера reverent\_bardeen?



* 1. 2556
	2. 49156
	3. Никакой
	4. 80
1. Какая команда должна подставлена вместо !! (двух восклицательных знаков), в команду docker rm -f $(!!) чтобы удалить все контейнеры без получения ошибок?
	1. docker ls -lA
	2. docker ps -a | cut -d " " -f1
	3. docker ps -q
	4. docker ps -aq
2. Какая команда выведет данные в таком виде?



* 1. docker ps -a --no-trunc
	2. docker ps --full
	3. docker ps -a --format="{{text .}}"
	4. docker ps --no-trunc
1. Сочетание каких флагов команды docker ps оставит в выводе только те контейнеры которые были закрыты?
	1. --closed -q
	2. --a --filter status=exited
	3. -q --filter status=closed
	4. --closed -a
2. Какая команда выведет информацию о последнем созданном контейнере?
	1. docker ps -n1
	2. docker ps --last-created
	3. docker ps -l 1
	4. docker ps --latest 1
3. Как удалить контейнер, не дожидаясь завершения основного процесса, запущенного в нём?
	1. docker rm –f name
	2. sudo systemctl restart docker.service
	3. shutdown –r now
	4. docker kill name
4. Где располагаются все контейнеры в хост-системе?
	1. В оперативной памяти
	2. /var/lib/docker/containers/
	3. Вопрос некорректен, поскольку контейнер является процессом, запущенным в рамках демона docker.service
	4. /proc/docker/containers/
5. Необходимо присвоить образу image:latest тег *v1.0*. Укажите команду, с помощью которой это можно сделать
	1. docker name image:latest image:v1.0
	2. docker tag image v1.0
	3. docker tag image:latest image:v1.0
	4. ничего из перечисленного
6. С помощью какой команды можно удалить образ legacy\_image?
	1. docker rm legacy\_image
	2. docker rmi legacy\_image
	3. docker system prune -a
	4. sudo rm -rf /
7. Что делает команда docker image rm myimage
	1. Удаляет все образы с именем myimage
	2. Пересоздает образ myimage
	3. Удаляет образ myimage:latest
	4. Ничего из перечисленного
8. Какой флаг удалит контейнер container вместе с ассоциированными с ним анонимными томами?
	1. docker rmi container
	2. docker rm -f container
	3. docker rm container
	4. docker rm -v container
9. Какой командой следует воспользоваться чтобы удалить все контейнеры?
	1. docker rmi -a
	2. docker images -q | docker rmi -f
	3. docker ps -q
	4. docker rmi -f $(docker images -q)
10. Что делает команда docker rmi --no-prune test\_image?
	1. Удаляет образ вместе с образами, на которых он основывался
	2. Удаляет образ вместе с родительскими образами, у которых нет тегов
	3. Удаляет образ не затрагивая родительские образы без тегов
	4. Удаляет образ и все образы, которые на нём основываются
11. Какой формат данных по-умолчанию использует docker inspect для вывода информации?
	1. CSV
	2. XML
	3. JSON
	4. HTML
12. Когда был создан образ gentoo/stage3 (тэг: amd64-openrc-20220213)?
	1. 2022-02-13T02:00:01.392810345Z
	2. 2022-02-13T02:00:01.373708401Z
	3. 2022-01-13T02:00:01.102597108Z
	4. 2022-02-13T02:00:07.610294863Z
13. Какой RepoDigest у тега 20.10.12-alpine3.15 образа docker?
	1. sha256:1f4a556c08bc2f54c0337847e0ffd964ad39da892c5d36fc851728d60645977f
	2. sha256:90c29958713df30b15a09c261d26cafc66bab0b23247300298267ef7f066fa67
	3. docker@sha256:a729cce205a05b0b86dc8dca87823efaffc3f74979fe7dc86a707c2fbf631b61
	4. docker@sha256:d848d730231a7e8303e538515251f95cfb6ce86ea7385c9d1bfae7cfd7eff179
14. Как выглядит полное имя флага -s команды docker inspect?
	1. --size
	2. --sort
	3. --split
	4. --search
15. Где по умолчанию располагается домашняя директория пользователя в контейнере, собранном из образа rabbitmq:3.9.0-management-alpine?
	1. /root
	2. /home/rabbitmq
	3. /
	4. /var/lib/rabbitmq
16. Какие порты открыты по умолчанию (ExposedPorts) в контейнере, запущенном из образа docker pull nginx:1.21?
	1. 20
	2. 21
	3. 25
	4. 80
	5. 3000
	6. 16535
17. Какой командой предпочтительней пользоваться для завершения работы с контейнером и почему?
	1. docker kill | её использование освобождает дисковое пространство, занятое контейнером
	2. docker stop | её использование останавливает контейнер, но сохраняет его в оперативной памяти, что позволит впоследствии запустить его заново
	3. docker kill | её использование гарантированно закроет контейнер (убьёт основной процесс контейнера)
	4. docker stop | эта команда вначале посылает основному процессу контейнера сигнал 15, что позволяет ему освободить занятые ресурсы, завершить собственные подпроцессы и сохранить необходимые данные
18. Выберете все команды, позволяющие контейнеру name продолжить свою работу после остановки командой docker pause name
	1. docker kill --signal SIGCONT name
	2. docker pause --undo name
	3. docker unpause name
	4. docker kill --signal 25 name
	5. docker stop --signal 18 name
	6. docker run name
	7. docker kill --signal SIGPLAY name
	8. docker kill --signal 9 name
	9. docker exec name init 6
19. Что делает команда docker stop -t 5 $(docker ps -q)?
	1. останавливает все запущенные контейнеры на 5 секунд
	2. останавливает выполнение последних 5 запущенных контейнеров
	3. всем работающим контейнерам предлагает завершить свою работу и, если какой-либо контейнер не выключиться спустя 5 секунд - закроет его принудительно
	4. Такой команды не существует
20. Можно ли использовать один Docker-контейнер в качестве шаблона для создания другого?
	1. Нет, нельзя
	2. Можно: для этого необходимо вручную скопировать файлы контейнера-шаблона в новый контейнер
	3. Можно: для этого необходимо сделать копию контейнера с использованием команды docker container copy
	4. Можно: для этого необходимо на основе контейнера-шаблона создать новый образ
21. Выберите верные утверждения: инструкция ADD need\_to\_copy . :
	1. Если need\_to\_copy - файл, то скопирует его в корневую директорию
	2. Если need\_to\_copy - файл, то скопирует его в рабочую директорию
	3. Если need\_to\_copy - директория, то скопирует эту директорию в рабочую директорию
	4. Если need\_to\_copy - директория, то скопирует содержимое данной директории в рабочую директорию
22. Укажите верные различия между инструкциями ADD и COPY:
	1. Инструкция ADD выполняется при запуске контейнера, а COPY при сборке образа
	2. Инструкция COPY позволяет добавлять файлы из удаленных источников по URL
	3. Инструкция COPY выполняется при запуске контейнера, а ADD при сборке образа
	4. Инструкция ADD позволяет распаковывать tar-архивы
23. Необходимо добавить в докер-образ следующие файлы: *test1.txt, test2.txt, test3.txt, test4.txt*; других файлов в директории сборки нет. С помощью какой(их) инструкций это можно сделать?
	1. ADD test\*.txt .
	2. COPY test??.txt .
	3. ADD test1.txt .
	ADD test2.txt .
	ADD test3.txt .
	ADD test4.txt .
	4. COPY test?.txt .
	5. Всеми перечисленными
24. При использовании инструкций ADD/COPY можно использовать в качестве параметра:
	1. Только относительные пути для source и destination
	2. Абсолютные пути для source и destination
	3. Только относительные пути для source и любые для destination
	4. Любые пути для source и только относительные для destination
	5. Только относительные пути для source и только абсолютные для destination
25. Файл .dockerignore содержит строку “dont\_copy\_folder” при выполнении команды docker build .
	1. Инструкция ADD dont\_copy\_folder . скопирует содержимое папки *dont\_copy\_folder*, а инструкция COPY dont\_copy\_folder . будет проигнорирована
	2. Инструкция COPY dont\_copy\_folder . скопирует содержимое папки *dont\_copy\_folder*, а инструкция ADD dont\_copy\_folder . будет проигнорирована
	3. Обе инструкции ADD dont\_copy\_folder . и COPY dont\_copy\_folder . будут проигнорированы
	4. Обе инструкции ADD dont\_copy\_folder . и COPY dont\_copy\_folder . скопируют содержимое папки
26. Кэш для RUN-инструкций во время сборки докер-образа может быть аннулирован с помощью:
	1. инструкции ADD, поскольку она добавляет файлы в образ
	2. инструкций ADD и COPY
	3. инструкции COPY, поскольку она копирует файлы в образ
	4. инструкции ADD и COPY не сбрасывают кэш RUN-инструкций
27. Выберите все верные утверждения, касающиеся инструкции LABEL:
	1. используется указания тега собираемого образа
	2. используется для указания метаданных образа
	3. указывается как пары “ключ=значение”
	4. всё перечисленное
28. Инструкция HEALTHCHECK используется для:
	1. проверки корректности сборки образа
	2. проверки работы контейнера
	3. получения кода завершения контейнера
	4. ничего из перечисленного
29. В docker-файле указано две инструкции HEALTHCHECK: первая проверяет наличие в контейнере которого файла, вторая - доступность некоторого URL. Какие из указанных инструкций будут выполняться?
	1. инструкции будут выполнены последовательно
	2. первая инструкция, т.к. она объявлена первой
	3. вторая инструкция, т.к. объявлена последней
	4. ни одна из указанных инструкций
30. Необходимо с помощью инструкции HEALTHCHECK проверять каждые 5 минут работоспособность контейнера с помощью некоторой команды, в случае 10 неудачных попыток контейнер должен считаться неработоспособным. Какие параметры необходимо указать в docker-файле для данной инструкции?
	1. --timeout=5m --tries=10
	2. --interval=5m --retries=10
	3. --period=5m --retries=10
	4. такие опции недоступны для инструкции HEALTHCHECK
31. Инструкция ARG задает переменные, которые
	1. доступны после запуска контейнера
	2. доступны во время сборки образа
	3. доступны во время запуска контейнера
	4. доступны после сборки образа
32. Выберите верные утверждения. Инструкция ENV задает переменные среды, которые:
	1. доступны только после запуска контейнера
	2. доступны во время сборки образа
	3. доступны после запуска контейнера
	4. ничего из перечисленного
33. Во время сборки образа выполняются следующие инструкции:

 RUN export MYVAR=5

 RUN echo $MYVAR > tmp

 Выберите верное утверждение:

* 1. В файл tmp запишется значение переменной MYVAR, равное 5
	2. Сборка завершится с ошибкой, поскольку переменная MYVAR не определена во второй инструкции
	3. В файл tmp запишется пустая строка, поскольку переменная MYVAR неопределена
1. Инструкция CMD используется для указания:
	1. кода завершения процесса сборки
	2. команды, выполняемой при запуске контейнера
	3. возможности подключения к контейнеру через терминал
	4. ничего из перечисленного
2. Выберите верные утверждения
	1. Инструкция RUN выполняет переданную команду и завершает сборку образа
	2. Инструкция RUN задает команду, которая будет выполнена при запуске контейнера
	3. При использовании нескольких инструкций CMD, они будут выполнены по очереди
	4. Ничего из перечисленного
3. Какую инструкцию необходимо добавить в Dockerfile перед инструкцией RUN apt install some\_program, чтобы обновить информацию о deb-пакетах и зависимостях:
	1. RUN apt install
	2. CMD apt update
	3. RUN apt update
	4. CMD apt upgrade
4. Выберите варианты корректного использования команды RUN для создания директории *my\_dir* в рабочей директории:
	1. RUN mkdir /my\_dir
	2. RUN [“mkdir”, “/my\_dir”]
	3. RUN [‘mkdir’, ‘my\_dir’]
	4. все перечисленные
	5. ни один из перечисленных
5. Команда docker build используется для:
	1. вывода информации об образе
	2. запуска контейнера
	3. вывода списка контейнеров
	4. сборки образа контейнера
6. При использовании команды docker build ./src
	1. docker-файл должен находиться в родительской директории (..)
	2. docker-файл должен находиться в директории ./src/build
	3. docker-файл должен находиться в текущей директории (.)
	4. docker-файл должен находиться в директории ./src
7. По умолчанию названием docker-файла, используемого при выполнении команды docker build, является:
	1. dockerfile
	2. DockerFile
	3. Dockerfile
	4. ищутся все перечисленные варианты
8. Флаг --no-cache используется в команде docker build для:
	1. удаления ранее собранного образа
	2. удаления кэша сборки
	3. игнорирования кэша при сборке
	4. удаления кэша после окончания сборки
9. Выберите корректную команду docker build, указывающую путь к docker-файлу “./build/dockerfile”
	1. docker build -f ./build/dockerfile
	2. docker build ./build/dockerfile
	3. docker build -f ./build/dockerfile .
	4. docker build ./build/dockerfile ./build/dockerfile
10. Контекстом сборки для команды docker build может служить:
	1. локальная директория
	2. tar-архив
	3. git-репозиторий
	4. все перечисленные
11. Контекст сборки - это
	1. набор файлов и директорий, передаваемый docker-процессу при запуске контейнера
	2. tar-архив, содержащий собранный docker-образ
	3. набор файлов и директорий передаваемый docker-процессу при сборке образа
	4. отладочная информация сборки, запущенной с помощью команды docker build
12. Какой тег будет назначен образу после успешного выполнения команды docker build -t my\_image .?
	1. <none>
	2. latest
	3. 0
	4. 1.0
13. Выберите результат добавления аргумента --quiet (-q) команды docker build:
	1. сборка производится без использования промежуточных контейнеров
	2. промежуточные контейнеры удаляются после сборки
	3. логи сборки не печатаются в stdout
	4. такого аргумента нет
14. Необходимо использовать наиболее новую версию базового образа при сборке. С помощью какого аргумента команды docker build это можно сделать?
	1. --update
	2. --check
	3. --newest
	4. --pull
15. Выберите аргумент команды docker build, ограничивающий размер оперативной памяти, выделяемой контейнеру сборки, 2 гигабайтами:
	1. --max-ram 2
	2. --max-size 2GB
	3. --ram 2048
	4. --memory 2g
16. С помощью какого аргумента можно установить аргументы сборки для команды docker build:
	1. --set-args VAR1=5;VAR2=10
	2. --build-args VAR1=5;VAR2=10
	3. --build-arg VAR1=5 --build-arg VAR2=10
	4. --args VAR1=5 --args VAR2=10
	5. установить аргументы сборки с помощью команды docker build нельзя
17. Выберите корректную команду для сборки образа с именем myimage:v1.0 и контекстом сборки, расположенным в директории ./src:
	1. docker build -t myimage:v1.0 --context ./src
	2. docker build -t=myimage:v1.0 ./src
	3. docker build -t myimage:v1.0 src
	4. docker build -f ./src/Dockerfile -t myimage:v1.0
18. Где обычно следует располагать команду обновления зависимостей и программ в Dockerfile?
	1. Как можно ближе к началу
	2. Как можно ближе к концу
	3. В середине
	4. Расположение данной команды не несёт никакой роли
19. Какие из приведённых ниже утверждений верны касательно наполнения Dockerfile?
	1. Dockerfile должен содержать директиву FROM
	2. Для получения доступа к портам контейнера необходимо в Dockerfile указать директиву EXPOSE
	3. За создание новых слоёв отвечают только директивы FROM и RUN
	4. Директива CMD не является обязательной
20. Влияет ли количество слоёв образа на скорость работы контейнера?
	1. Да, потому что каждый слой порождает отдельный контейнер, общающиеся между другом посредством специальных запросов.
	2. Нет, потому что каждый слой порождает отдельный контейнер, но они работают в рамках одного дискового пространства
	3. Да, потому что при запуске контейнера каждый слой вносит определённые изменения в контейнер, создаваемый из базового образа
	4. Нет, потому что слой - образ, используемый для создания другого образа посредством внесения изменений в свою копию. При этом контейнер создаётся на базе одного образа, уже содержащего все необходимые файлы и приложения
21. Выберете верные утверждения касательно директивы FROM в Dockerfile
	1. Это обязательная команда
	2. Задаёт базовый образ
	3. Невозможно использовать несколько FROM в одном файле
	4. Эта команда должна быть размещена на самой первой строчке файла
22. В каких случаях используется ENTRYPOINT?
	1. это синоним CMD
	2. когда необходимо запретить изменять корень дерева процессов контейнера
	3. такой директивы нет
	4. когда необходимо сделать корнем дерева процесса базовый процесс базового образа
23. Какие утверждения об сборке образов неверны?
	1. Сборка происходит до запуска контейнера
	2. Сборка происходит непосредственно перед каждым запуском контейнера
	3. Сборка инкрементальна,т.е. к каждый новый слой создаётся на основе старого посредством добавления в него изменений
	4. Сборка нового образа всегда сопровождается созданием всех промежуточных слоёв с нуля
24. Для чего используется директива RUN?
	1. Для запуска команды из оболочки используемого образа
	2. Для задания новых переменных среды
	3. Для копирования файлов в процессе сборки образа
	4. Для определения базового процесса контейнера на этапе сборки образа
25. Можно ли в рамках одного RUN запустить несколько команд?
	1. Можно, но только если вывод одной используется как входные данные другой (cat file.txt | sed ‘s/\s\+//’ )
	2. Можно, но разделяя команды символами “&&” и “||”
	3. Можно, но только разделяя команды символом “;”
	4. Всё вышеперечисленное верно
	5. Нельзя

Тема 3. Инструменты автоматизации развертывания и поставки ПО

1. Что такое docker-compose?
	1. Инструмент, позволяющий быстро разворачивать многоконтейнерные приложения
	2. Утилита, предназначенная для составления Dockerfile через CLI
	3. Утилита, предназначенная для составления Dockerfile через GUI
	4. Часть докер, отвечающая за создание и управление сетями
2. Какие форматы файлов могут быть использованы для создания compose-конфигурации?
	1. yaml
	2. json
	3. csv
	4. docx
3. Что делает аргумент --no-cache команды docker-compose build?
	1. Запрещает docker-процессу использовать кэш-память при сборке
	2. Не кэширует сборку образа
	3. Игнорирует существующий кэш сборки
	4. Позволяет установить местоположение кэша для использования
4. Выберите аргумент команды docker-compose build, проверяющий наличие более новой версии образа и скачивающий его для использования в сборке:
	1. --newest
	2. --pull
	3. --update
	4. --check
5. Выберите верную команду docker-compose build, устанавливающую следующие аргументы сборки: VAR - число 0, VAR1 - строка ‘build’
	1. docker-compose build --build-args VAR=0;VAR1=build
	2. docker-compose build --build-arg VAR 0 --build-arg VAR1 build
	3. docker-compose up --build-arg VAR=0 --build-arg VAR1=build
	4. docker-compose build --build-arg VAR=0 --build-arg VAR1=’build’
6. Выберите аргумент команды docker-compose build, ограничивающий размер оперативной памяти, выделяемой контейнеру сборки, 5 гигабайтами:
	1. --max-size 5GB
	2. --memory 5g
	3. --max-ram 5
	4. --ram 5120
7. С помощью какого аргумента команды docker-compose build можно скрыть логи сборки?
	1. --no-logs
	2. --quiet
	3. --logs None
	4. скрыть логи сборки невозможно
8. Что происходит с контейнерами, запущенными с помощью команды docker-compose up, после завершения команды?
	1. Контейнеры продолжают работать
	2. Контейнеры останавливаются
	3. Зависит от кода завершения команды
	4. Ничего из перечисленного
9. В compose-конфигурации используются локальные образы, которые необходимо предварительно собрать. Можно ли сделать это без использования команды docker-compose build?
	1. нет, сборка образов возможна только с помощью команды docker-compose build
	2. да, с помощью аргумента --build-missing команды docker-compose up
	3. да, с помощью аргумента --build команды docker-compose up
	4. ничего из перечисленного
10. Необходимо запустить контейнеры compose-конфигурации в фоновом режиме. Выберите команду, с помощью которой это можно сделать:
	1. docker-compose up
	2. docker-compose up --daemon
	3. docker-compose up -d
	4. docker-compose up --background
11. Каким образом можно удалить контейнеры compose-сервисов, которые не определены в compose-файле, например, после его изменения?
	1. вручную, с помощью команды docker rmi
	2. автоматически, с помощью аргумента --remove-orphans команды docker-compose up
	3. вручную, с помощью команды docker rm
	4. ничего из перечисленного
12. Необходимо удалить и пересоздать все контейнеры, определенные в compose-конфигурации. С помощью какого аргумента команды docker-compose up это можно сделать?
	1. --remove
	2. --recreate
	3. --force-recreate
	4. удаление с помощью команды docker-compose up невозможно
13. Сервис в compose-конфигурации представляет из себя:
	1. описание сборки docker-образа
	2. инструкции по запуску compose-конфигурации
	3. конфигурацию запускаемого контейнера
	4. настройки взаимодействия между контейнерами
14. Compose-конфигурация содержит некоторое количество сервисов. Выберите команду, с помощью которой можно запустить сервис web, не запуская другие сервисы:
	1. docker-compose up
	2. docker-compose run web
	3. docker-compose up --service web
	4. docker-compose start web
15. В compose-конфигурации с сервисами web и db используется сеть по умолчанию. Как можно получить доступ к контейнеру db из контейнера web с помощью данной сети?
	1. получение доступа невозможно в сети по умолчанию
	2. доступ возможен по имени хоста web
	3. доступ возможен по localhost
	4. ничего из перечисленного
16. С помощью какой команды необходимо создать external сеть, чтобы использовать её в compose-конфигурации:
	1. docker-compose create network
	2. docker network create
	3. docker create network
	4. docker-compose network
17. По умолчанию команды docker-compose используют переменные среды, указанные в файле:
	1. .env в директории environment директории проекта
	2. environment.txt
	3. .env в корне директории проекта
	4. docker-compose не устанавливает переменные среды без команды пользователя
18. Установка переменных среды в контейнере возможна с помощью:
	1. опции *environment* в compose-файле
	2. опции *env\_file* в compose-файле
	3. с помощью флага -e команды *docker-compose run*
	4. всё перечисленное
19. Для одного из сервисов compose-конфигурации используется сторонний образ, для работы которого необходимо передать файл настроек *container/settings.yaml*, хранящийся в директории проекта. Каким способом это можно сделать через изменение compose-конфигурации?
	1. модифицировать Dockerfile используемого образа, добавив инструкцию COPY container/settings.yaml .
	2. с помощью опции volumes, путем монтирования файла настроек с хоста в контейнер
	3. ничего из перечисленного
20. Чем может быть чревато отсутствие ограничения объема оперативной памяти, используемой докер-контейнерами?
	1. контейнеру может быть выделено памяти больше, чем доступно на хост-машине
	2. чрезмерное потребление памяти хост-машины
	3. контейнер сможет использовать только дисковое пространство
	4. ничего из перечисленного
21. При использовании параметров, контролирующих объем доступа контейнера к ресурсам ЦП, docker:
	1. изменяет настройки ЦП хост-машины
	2. изменяет параметры контрольной группы контейнера
	3. эмулирует ЦП с заданными параметрами для контейнера
	4. ничего из перечисленного
22. Выберите команду для запуска контейнера, которому будет доступно не более 2.5 процессоров:
	1. docker run --set-cpus="2.5" ubuntu /bin/bash
	2. docker run --cpus="2.5" ubuntu /bin/bash
	3. docker run --limit-cpus="2.5" ubuntu /bin/bash
	4. ничего из перечисленного
23. На хост-машине доступно 100 процессоров, выберите две команды, запускающие контейнеры, которые будут иметь доступ к 50 разным процессорам, т.е. не будут иметь доступа к одним и тем же процессорам:
	1. docker run --cpus="50" ubuntu /bin/bash и docker run --cpus="50" ubuntu /bin/bash
	2. docker run --cpuset-cpus="0,49" ubuntu /bin/bash и docker run --cpuset-cpus="50,99" ubuntu /bin/bash
	3. docker run --cpuset-cpus="0-49" ubuntu /bin/bash и docker run --cpuset-cpus="50-99" ubuntu /bin/bash
	4. нельзя ограничить или предоставить доступ к конкретным процессорам
24. Необходимо получить подробную информацию об использовании дискового пространства docker-процессом. С помощью какой команды можно это сделать?
	1. docker info
	2. docker system df -v
	3. docker ps
	4. docker system info
25. Какие докер-объекты можно удалить с помощью команды *docker … prune*?
	1. контейнеры
	2. тома
	3. сети
	4. образы
	5. всё перечисленное
26. Dangling-образы, удаляемые командой *docker images prune* - это:
	1. образы остановленных контейнеров
	2. образы без тега и контейнеров, использующих данные образы
	3. образы, используемые в активных контейнерах
	4. ничего из перечисленного
27. В чём состоит потенциальная опасность большого количества dangling-образов?
	1. переполнение базы данных, хранящей данные об образах
	2. большое потребление дискового пространства
	3. чрезмерное количество слоев образа, создаваемых при сборке
	4. ничего из перечисленного
28. Выберите команду, которая удалит образы, контейнеры, сети и тома:
	1. docker container prune
	2. docker system prune
	3. docker system prune --volumes
	4. docker volume prune --all
29. Выберите основные характеристики томов:
	1. не увеличивают размер контейнеров
	2. цикл жизни зависит от цикла жизни контейнера
	3. существуют вне жизненного цикла контейнера
	4. ничего из перечисленного
30. Выберите команду для создания тома:
	1. docker create volume hello
	2. docker volume create hello
	3. том можно создать только с помощью compose-конфигурации
	4. ничего из перечисленного
31. Необходимо получить список томов на хост-машине, с помощью какой команды это можно сделать?
	1. docker volume ps
	2. docker volume ls
	3. docker volume list
	4. ничего из перечисленного
32. Выберите команду для удаления всех неиспользуемых томов:
	1. docker volume rm
	2. docker volume prune
	3. docker volumes prune
	4. ничего из перечисленного
33. Выберите потоки вывода, в которые выводит данные команда docker logs
	1. stderr
	2. stdin
	3. stdlog
	4. stdout
	5. syslog
	6. logger
34. Какую функцию выполняет флаг - - follow команды docker logs
	1. Выводит справку о работе
	2. Показывает дополнительную информацию
	3. Выводит метки времени построчно
	4. Выводит записи начиная с заданной метки времени
	5. Выводит записи начиная до заданной метки времени
	6. Выводит N последних строк лога
	7. Включает интерактивное отображение лога в реальном времени работы контейнера
35. Какую функцию выполняет флаг --since команды docker logs
	1. Выводит справку о работе
	2. Показывает дополнительную информацию
	3. Выводит метки времени построчно
	4. Выводит записи начиная с заданной метки времени
	5. Выводит записи начиная до заданной метки времени
	6. Выводит N последних строк лога
	7. Включает интерактивное отображение лога в реальном времени работы контейнера
36. Какую функцию выполняет флаг --until команды docker logs
	1. Выводит справку о работе
	2. Показывает дополнительную информацию
	3. Выводит метки времени построчно
	4. Выводит записи начиная с заданной метки времени
	5. Выводит записи начиная до заданной метки времени
	6. Выводит N последних строк лога
	7. Включает интерактивное отображение лога в реальном времени работы контейнера
37. Какую функцию выполняет флаг --tail команды docker logs
	1. Выводит справку о работе
	2. Показывает дополнительную информацию
	3. Выводит метки времени построчно
	4. Выводит записи начиная с заданной метки времени
	5. Выводит записи начиная до заданной метки времени
	6. Выводит N последних строк лога
	7. Включает интерактивное отображение лога в реальном времени работы контейнера
38. Какие типы сетей (driver) поддерживаются docker?
	1. bridge
	2. host
	3. none
	4. Ничего из вышеперечисленного
39. Что такое overlay-network в Docker?
	1. изолированная сеть контейнера, однако есть возможность проброса портов в хост-систему
	2. общая сеть между несколькими docker-демонами (хостами)
	3. отсутствие изоляции - контейнеры работают на сетевом адаптере хоста
	4. без сети
40. Что такое bridge-network в Docker?
	1. изолированная сеть контейнера, однако есть возможность проброса портов в хост-систему
	2. общая сеть между несколькими docker-демонами (хостами)
	3. отсутствие изоляции - контейнеры работают на сетевом адаптере хоста
	4. без сети
41. Что такое host-network в Docker?
	1. изолированная сеть контейнера, однако есть возможность проброса портов в хост-систему
	2. общая сеть между несколькими docker-демонами (хостами)
	3. отсутствие изоляции - контейнеры работают на сетевом адаптере хоста
	4. без сети
42. Что такое links в docker-compose?
	1. дополнительные алиасы для сервисов (контейнеров)
	2. адреса, на которых будут развернуты контейнеры
	3. ссылки на Docker-registry, в которые будут поставляться собранные образы
	4. ссылки на контейнеры из-вне docker-compose
43. Что такое external\_links в docker-compose?
	1. дополнительные алиасы для сервисов (контейнеров)
	2. адреса, на которых будут развернуты контейнеры
	3. ссылки на Docker-registry, в которые будут поставляться собранные образы
	4. ссылки на контейнеры из-вне docker-compose
44. В чём заключается различие между Docker и Docker-compose?
	1. Docker - используется для управления отдельными контейнерами; Docker-compose - для одновременного управления несколькими контейнерами
	2. Docker - система изоляции процессов посредством контейнеризации; Docker-compose - система изоляции процессов посредством виртуализации
	3. Docker-compose - часть Docker, отвечающая за работу сетей
	4. Docker - часть Docker-compose, отвечающая за создание контейнеров из образов

Тема 4. Серверы непрерывной интеграции

1. Что является главным общим свойством инструментов Jenkins и Github Actions?
	1. Это инструменты CI/CD
	2. Они позволяют настроить автоматический запуск тестирования приложений
	3. Они позволяют хранить и обрабатывать информацию об изменениях в коде проекта
	4. У данных приложений нет ничего общего
2. Что такое Jenkins?
	1. средство автоматизации, используемое для создания, тестирования и развертывания ПО, упрощая непрерывную интеграцию и непрерывное развертывание для пользователей.
	2. Система управления Docker-контейнерами
	3. программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации
	4. Сервис хранения и распространения кода, аналог github
3. Есть ли возможность в рамках Jenkins запускать docker-контейнеры?
	1. Есть, но только если Jenkins-сервер сам не является docker-контейнером
	2. Есть, но только если на хосте, на котором установлен Jenkins, есть Docker
	3. Есть, поскольку Jenkins поставляется со своей версией Docker
	4. Такой возможности нет

3.5.1. Project

1. Project в Jenkins – это:
	1. Настраиваемое пользователем описание работы, которую должен выполнять Jenkins, например, создание части программного обеспечения.
	2. Неизменяемый файл, созданный во время выполнения Build или Pipeline, который архивируется на Jenkins Controller для последующего использования пользователями.
	3. Машина или контейнер, подключающиеся к Jenkins Controller и выполняющие задачи по его указанию.
	4. Временный каталог в файловой системе Node, где может выполняться работа.
2. Какие типы проектов существуют в Jenkins?
	1. FreeStyle Project
	2. CI/CD project
	3. Jenkins Project
	4. Single-node Project
	5. Pipeline Project
	6. Cloud Project
3. Каких видов бывает Pipeline Project?
	1. Declarative Pipeline
	2. FreeStyle Pipeline
	3. Scripted Pipeline
	4. CI/CD Pipeline
	5. Inline Pipeline
4. FreeStyle Project создается:
	1. добавлением шагов в сборку в графическом интерфейсе
	2. заданием набора инструкций с помощью кода
	3. запуском скрипта из командой строки
	4. установкой специального плагина
5. Pipeline Project создается:
	1. добавлением шагов в сборку в графическом интерфейсе
	2. заданием набора инструкций с помощью кода
	3. запуском скрипта из командой строки
	4. установкой специального плагина
6. В Pipeline Project выделяют
	1. steps и stages
	2. steps и sub-jobs
	3. stages и sub-jobs
	4. CI и CD компоненты
7. FreeStyle Project
	1. не требует написания кода
	2. подходит для простых проектов
	3. подходит для сложных проектов
	4. требует написания кода
8. Pipeline Project
	1. А) относительно просто создается
	2. Б) подходит для простых проектов
	3. В) подходит для сложных проектов
	4. Г) требует написания кода
9. Где по умолчанию хранятся артефакты работы Freestyle Project под названием first\_test?
	1. /var/run/jenkins/projects/first\_test/
	2. /var/lib/jenkins/jobs/first\_test/
	3. /var/lib/jenkins/jobs/first\_test/builds/
	4. /var/lib/jenkins/projects/first\_test/builds/
10. Какие автоматические способы запуска project доступны в Jenkins по умолчанию (без дополнений)?
	1. В Jenkins без дополнений нет такой возможности
	2. Раз в определённый промежуток времени
	3. При событии, возникающем в github (merge, pull-request, коментарий) по webhook
	4. После завершения работы другого project

3.5.2. Build

1. Build в Jenkins – это:
	1. Результат одного выполнения Project
	2. Центральный координирующий процесс, который хранит конфигурацию, загружает плагины и отображает различные пользовательские интерфейсы для Jenkins.
	3. Настраиваемый Pipeline или Project, который запускается как часть выполнения другого Pipeline или Project.
	4. Слот для выполнения работы, определенной Pipeline или Project на Node.
2. Какие статусы могут быть у build?
	1. Aborted
	2. Finished
	3. Failed
	4. Started
	5. Stable
	6. In progress
	7. Successful
	8. Pending
	9. Unstable
3. Какой статус означает, что в build была фатальная ошибка?
	1. Aborted
	2. Failed
	3. Unstable
	4. Finished
4. Какой статус означает, что в build не было ошибок компиляции?
	1. Finished
	2. Started
	3. Stable
	4. Successful
5. Какой статус означает, что в build были ошибки компиляции, но не фатальные?
	1. Aborted
	2. Finished
	3. Failed
	4. Unstable
6. Какой статус означает, что build был прерван до ожидаемого окончания?
	1. Aborted
	2. Failed
	3. Pending
	4. Unstable
7. Как можно изменить номер, который присвоится следующему build?
	1. Отредактировать файл /var/lib/jenkins/jobs/<project\_name>/nextBuildNumber
	2. Удалить информацию о существующих build через графический интерфейс
	3. Возможность изменения номера не предусмотрена
	4. Нет правильного ответа

3.5.3. Node

1. Node в Jenkins – это:
	1. Настраиваемое пользователем описание работы, которую должен выполнять Jenkins, например, создание части программного обеспечения.
	2. Машина, являющаяся частью среды Jenkins и способная выполнять Pipeline или Project.
	3. Основное приложение Jenkins, которое предоставляет базовый веб-интерфейс, конфигурацию и основу для создания плагинов.
	4. Расширение функциональных возможностей Jenkins, предоставляемое отдельно от Jenkins Core.
2. Что считается node?
	1. controller
	2. agent
	3. project
	4. build
	5. item
3. Controller - это:
	1. Центральный координирующий процесс, который хранит конфигурацию, загружает плагины и отображает различные пользовательские интерфейсы для Jenkins.
	2. Расширение функциональных возможностей Jenkins, предоставляемое отдельно от Jenkins Core.
	3. Основное приложение Jenkins, которое предоставляет базовый веб-интерфейс, конфигурацию и основу для создания плагинов.
	4. Слот для выполнения работы, определенной Pipeline или Project на Node.
4. Executor - это:
	1. Центральный координирующий процесс, который хранит конфигурацию, загружает плагины и отображает различные пользовательские интерфейсы для Jenkins.
	2. Машина или контейнер, подключающиеся к Jenkins Controller и выполняющие задачи по его указанию.
	3. Машина, являющаяся частью среды Jenkins и способная выполнять Pipeline или Project.
	4. Слот для выполнения работы, определенной Pipeline или Project.
5. Agent - это:
	1. Центральный координирующий процесс, который хранит конфигурацию, загружает плагины и отображает различные пользовательские интерфейсы для Jenkins.
	2. Машина, являющаяся частью среды Jenkins и способная выполнять Pipeline или Project.
	3. Машина или контейнер, подключающиеся к Jenkins Controller и выполняющие задачи по его указанию.
	4. Основное приложение Jenkins, которое предоставляет базовый веб-интерфейс, конфигурацию и основу для создания плагинов
6. Сколько executor может быть на node?
	1. 0 и больше
	2. 2 и больше
	3. строго 1
	4. строго 2