Общество с ограниченной ответственностью «ОНЛАНТА КОД ИТ»

Программное обеспечение «ONPLATFORM»

Руководство пользователя

Москва 2023 г.

Аннотация

Настоящий документ содержит указания и рекомендации необходимые для правильной работы с программным обеспечением «ONPLATFORM» – программной платформой, разработанной ООО «ОНЛАНТА КОД ИТ» (далее – Платформа).

Содержание

1 Общие сведения

1.1 Наименование

Полное наименование: Программное обеспечение «ONPLATFORM».

Условное обозначение: Платформа.

1.2 Разработчик

Общество с ограниченной ответственностью «ОНЛАНТА КОД ИТ»
(ООО «ОНЛАНТА КОД ИТ»)

Фактический адрес: 129075, г. Москва, Мурманский проезд, д. 14, стр. 5

kodit@onlanta.ru; https://onkodit.ru

Тел./факс: +7 (495) 258 89 86

1.3 Назначение Платформы

Платформа предназначена для автоматизации процессов разработки ПО путем развертывания и последующего обслуживания рабочих кластеров Kubernetes с интегрированными дополнительными модулями.

Платформа предлагает готовые решения для типовых задач в области автоматизации сборки, поставки приложений в целевые окружения, управления конфигурациями, обеспечения информационной безопасности, мониторинга и диагностики, снижая, таким образом, затраты, связанные с внедрением систем для решения подобных задач и позволяя компаниям сосредоточиться на развитии собственных программных продуктов.

1.4 Перечень терминов, определений и сокращений

Термин	Описание
Деплой	Процесс установки/обновления программного обеспечения в среде Kubernetes
Нода (узел)	Точка в сети, которая либо распределяет данные между другими узлами (нодами) сети, либо является конечной точкой сети
ОС	Операционная система
ПО	Программное обеспечение
ППО	Прикладное программное обеспечение
СПО	Системное программное обеспечение

Термин	Описание
СВАС	Система виртуализации аппаратных средств - любая система виртуализации, в том числе могут использоваться системы из Реестра отечественного программного обеспечения, такие как Скала-Р, Астра Брест и т.д.
Виртуализация	Предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию друг от друга вычислительных процессов, выполняемых на одном физическом ресурсе.
BGP	Протокол динамической маршрутизации, который относится к классу протоколов маршрутизации внешнего шлюза EGP
CSI-драйвер	Container Storage Interface Компонент, обеспечивающий управление выделением томов (PV) оркестратором Kubernetes
DNS	Domain Name System Сервис, обеспечивающий возможность использования доменных имён вместо IP-адресов, а также корректную работу служб Платформы в условиях динамического выделения IP-адресов
GitOps	Подход, при котором состояние целевой системы (в данном случае, кластера Kubernetes) хранится в репозитории Git и обновляется автоматически при появлении изменений в этом репозитории
IDM	IDentity Manager Сервис, обеспечивающий централизованное хранение информации об учётных записях пользователей и их правах на пользование службами Платформы, а также удостоверяющий пользователей при входе в соответствующие службы
Ingress-контроллер	Отвечает за создание и изменение ресурсов Application Load Balancer
Kea DHCP-сервер	Включает в себя полнофункциональную реализацию сервера с поддержкой протоколов DHCPv4 и DHCPv6. Основан на технологиях BIND 10 и построен с использованием модульной архитектуры
Kubernetes	Программный комплекс, обеспечивающий функционал управления контейнерными средами на одной или

Термин	Описание
	нескольких нодах
LINSTOR-контроллер	Основной контроллер, который предоставляет API для создания и управления ресурсами
Mozilla SOPS	Инструмент управления секретами с открытым исходным кодом
Pod	Набор из одного или более контейнеров для совместного развертывания на ноде, реализующий какую-либо функцию
PV	Persistent Volume Логический том в кластере Kubernetes для хранения данных между перезапусками контейнеров
PVC	Persistent Volume Claim Привязка логического тома (PV) к конкретному приложению в кластере Kubernetes для хранения данных между перезапусками контейнеров
Service Mesh	Компонент, обеспечивающий виртуальную сеть внутри кластера Kubernetes для обеспечения безопасности передаваемых данных, мониторинга обмена данными
SNAT	Замена адреса источника при прохождении пакета в одну сторону и обратной замене адреса назначения в ответном пакете
SSL	Secure Sockets Layer Протокол, обеспечивающий безопасность соединений и обеспечивающий гарантированную сохранность информации, передаваемой по сетям связи, от считывания и модификации путём шифрования передаваемых данных
VPN	Virtual Private Network защищённый канал связи, обеспечивающий безопасный обмен данными между клиентом и Платформой

2 Подготовка к работе

2.1 Состав дистрибутива

Программное обеспечение, используемое Платформой, имеет открытый исходный код. Перечень ПО, используемого Платформой и включенного в состав дистрибутива:

Модуль	ПО

Модуль	ПО
Kubernetes	• Calico. Плагин для Kubernetes, реализующий виртуальную оверлейную сеть и распределенный фаервол для организации и контроля сетевого взаимодействия между контейнеризованными приложениями, запущенными в Kubernetes; • Keda. ПО, реализующее декларативный программный интерфейс для управления автоматическим масштабированием ППО, запущенном в кластерах Kubernetes; • Kubernetes. ПО для оркестрации контейнеризованных приложений: автоматизации их развёртывания, масштабирования и координации; • LINSTOR. Программно-определяемое распределенное хранилище данных, реализующее поддержку персистентного хранения данных СПО и ППО, запущенным в кластерах Kubernetes; • Linkerd. ПО, реализующее сервисную сеть (service mesh) для управления взаимодействием между сервисами (приложениями) в распределенной системе; • MetalLB. ПО, реализующее декларативный программный интерфейс для управления сетевым трафиком, входящим в кластеры Kubernetes; • cert-manager. ПО, реализующее декларативный программный интерфейс для управления выпуском TLS-сертификатов для использования в различном СПО и ППО, запущенном в кластерах Kubernetes; • containerd. Контейнерная среда, используемая Kubernetes для запуска контейнеров; • etcd. Распределенное хранилище конфигурационных данных для Kubernetes и прочих систем; • ingress-nginx. ПО, реализующее декларативный программный интерфейс для управления обработкой входящих запросов к СПО и ППО, развернутому в кластерах Kubernetes; • Docker. ПО для контейнеризации, используемое для запуска отдельных инфраструктурных компонентов Платформы; • Kea. ПО, используемое для организации DHCP-сервера для нужд Платформы. • Nginx. Веб-сервер и прокси-сервер для Unix-подобных систем.
Безопасность	• Dex IDP. ПО, позволяющее использовать информацию о пользовательских учетных записях, хранящуюся на серверах каталогов (LDAP), для аутентификации пользователей в Kubernetes; • FreeIPA. Сервер каталогов (LDAP), а также

Модуль	ПО
	программный и визуальный интерфейс для централизованного управления пользователями платформенных сервисов; • HashiCorp Vault. Распределенное хранилище данных, чувствительных к компрометации (пароли, ключи, токены и т.д.); • Kyverno. ПО, реализующее декларативный программный интерфейс для управления политиками безопасности, применяющимися к СПО и ППО, запущенному в кластерах Kubernetes; • SELinux. Встроенный механизм контроля доступа, реализованный на уровне ядра. Определяет политики доступа к приложениям, процессам и файлам; • Sysdig Falco. ПО для аудита событий на уровне ОС, СПО и ППО; • CFSSL. CloudFlare SSL, удостоверяющий центр в составе Платформы.
Мониторинг	• Alertmanager. ПО, предоставляющее декларативный программный интерфейс для управления правилами генерации и отправки оповещений по данным из централизованного хранилища метрик; • Grafana. ПО для для визуализации и анализа данных, находящихся в централизованном хранилище метрик; • Prometheus. Агрегатор и централизованное хранилище метрик, агрегируемых с уровней ОС, СПО и ППО; • Prometheus Operator. ПО, реализующее декларативный программный интерфейс для управления экземплярами Prometheus, запускаемыми внутри кластеров Kubernetes; • VictoriaMetrics. Централизованное долгосрочное хранилище метрик; • Karma. ПО, предоставляющее веб-интерфейс для просмотра активных событий; • kube-state-metrics. ПО, реализующее функционал сбора метрик мониторинга состояния кластеров Kubernetes; • metrics-server. ПО, реализующее сбор информации о потреблении ресурсов CPU/RAM приложениями в кластере Kubernetes. • CPU-hiccup. Расширенный по функциональности экспортер метрик мониторинга. • Jaeger. ПО для агрегации данных распределенной трассировки запросов, генерируемых СПО и ППО, запущенным в кластерах Kubernetes.

Модуль	ПО
Мониторинг *(логирование)*	• Amazon OpenDistro/OpenSearch (Logstash, OpenSearch, Kibana). Централизованное хранилище диагностических данных из разных источников (например, из журналов событий на уровне ОС, СПО и ППО), а также набор инструментов для визуализации и анализа накопленных данных; • DataDog Vector. ПО для организации сбора событий из различных журналов на уровне ОС, СПО и ППО и отправки этих данных в централизованное хранилище.
Деплой *(конвейер CI/CD)*	• Flagger. ПО, реализующее поддержку продвинутых сценариев деплоя (blue/green, canary) в Kubernetes; • Flux. ПО, реализующее декларативный программный интерфейс для автоматизации установки СПО и ППО в кластерах Kubernetes; • GitLab. Менеджер Git-репозиториев и CI/CD-сервер; • Helm. Пакетный менеджер, реализующий метод упаковки СПО и ППО для развертывания в кластерах Kubernetes; • Nexus OSS. Менеджер репозиториев.
Администрирование *(управление инфраструктурой и Платформой)*	• AlmaLinux. Свободно распространяемый дистрибутив Linux; • Ansible. ПО для автоматизации конфигурирования ОС и СПО; • HashiCorp Terraform. ПО для автоматизации управления виртуальной и сетевой инфраструктурой; • PowerDNS. DNS-сервер для Unix-подобных систем. Может получать DNS-информацию из различных источников данных. Используется для организации балансировки DNS-трафика. • DNS-inventory. Инструмент, обрабатывающий наборы атрибутов хоста для создания инвентаризационного файла Ansible. • Onadm. Утилита, которая на текущий момент скачивает обновленные версии свободного ПО из Интернета и после сборки размещает их во внутреннем хранилище пакетов Платформы.
Kubernetes data plane *(резервное копирование)*	• MinIO. Распределенное объектное хранилище данных с программным интерфейсом, совместимым с Amazon S3; • Velero. ПО, реализующее декларативный программный интерфейс для управления резервным копированием данных, генерируемых СПО и ППО, запущенным в кластерах Kubernetes.

2.2 Уровень подготовки пользователя

Пользователи Платформы должны обладать навыками использования персональных компьютеров под управлением операционных систем семейства Windows и работы с пакетом программ MSOffice, а также знания Git, CI/CD конвейеров, контейнеризации приложений, написания Helm-чартов и развёртывания приложений в Kubernetes.

Перед началом работы с Платформой пользователь должен ознакомиться с настоящим руководством.

2.3 Техническое обеспечение рабочего места пользователя

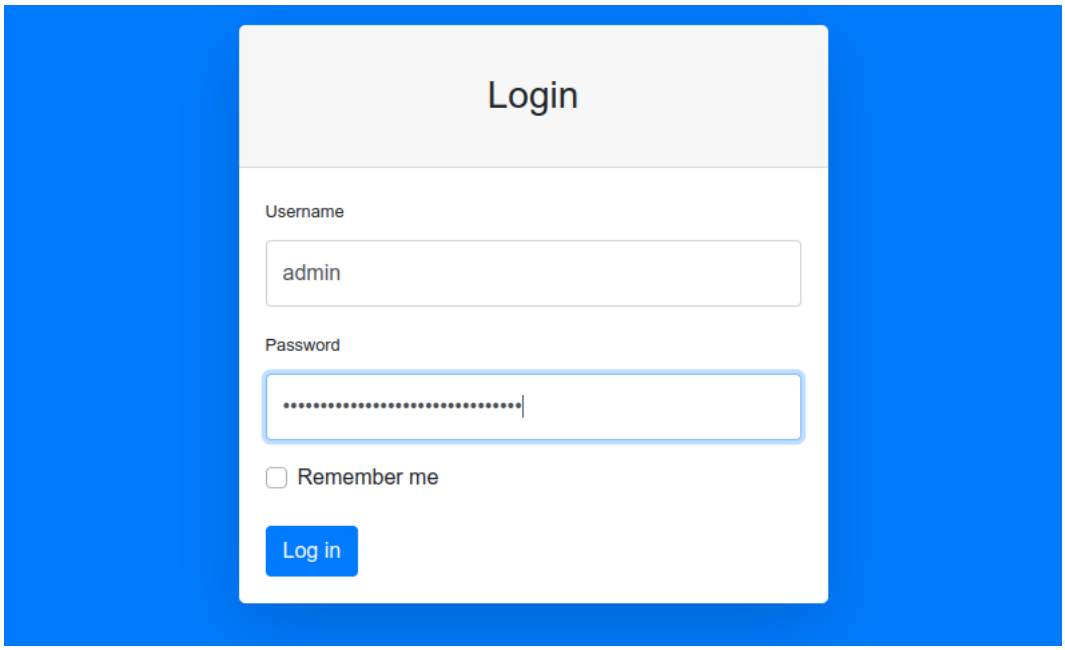
Рабочее место пользователя Платформы должно отвечать следующим требованиям:

- наличие возможности удаленного доступа к инфраструктуре, на которой развернута Платформа;
- наличие доступа в сеть Интернет – скорость не менее 10 Мбит;
- наличие SSH-клиента – SSH-доступ по ключу до узла, являющегося Master-узлом кластера;
- наличие VPN-клиента WireGuard для подключения к внутренней платформенной сети передачи данных.

2.4 Создание учётных записей пользователя

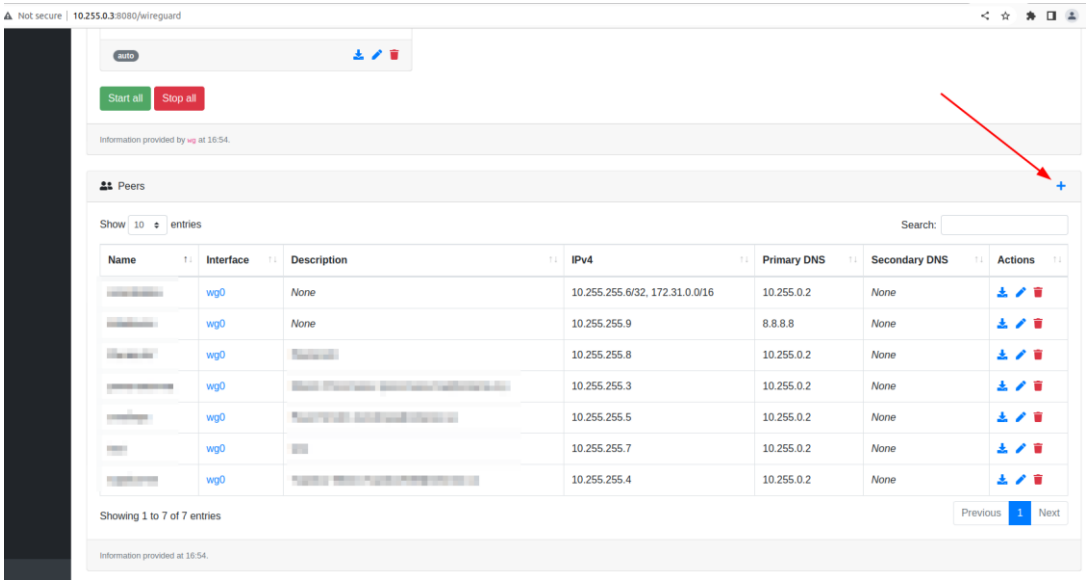
Создавать учетные записи пользователей Платформы можно выполнив следующие действия:

1) Запросить у пользователя публичный ssh-ключ для обеспечения доступа на сервер администрирования;
2) Создать учетную запись пользователя с использованием linguard:
   - открыть веб-интерфейс по адресу http://10.255.0.3:8080/wireguard и авторизоваться, используя учётную запись администратора:

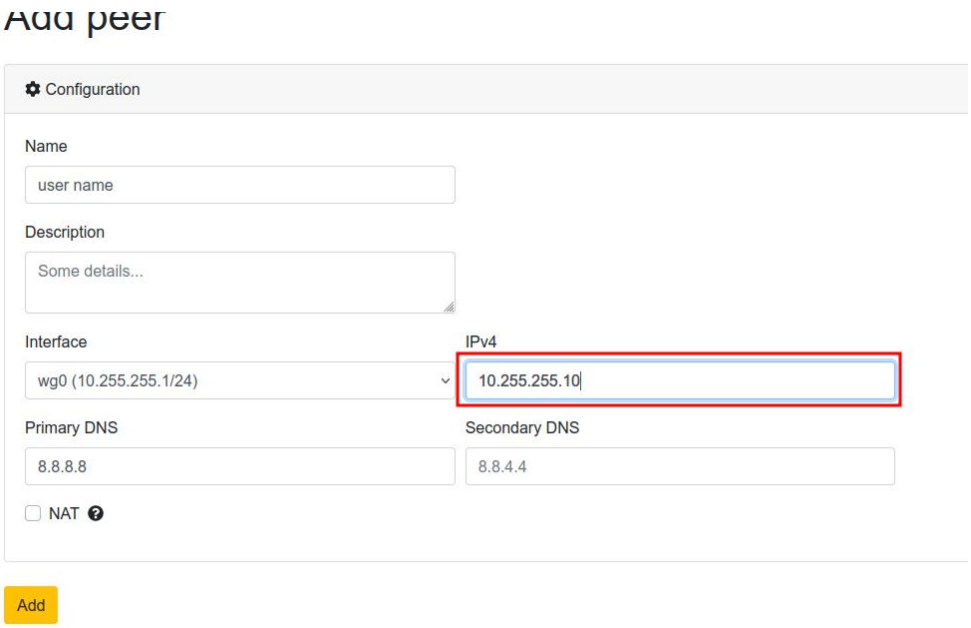


– определить по списку пользователей первый незанятый IP-адрес, чтобы использовать его для учетной записи нового пользователя;

– нажать на кнопку, показанную на рисунке ниже, для добавления учетной записи пользователя:



– ввести имя пользователя, заменить Primary DNS на адрес DNS Платформы (чаще всего 10.255.0.2) и убедиться, что предлагаемый IP свободен, либо ввести своё значение, определённое ранее:



– нажатием кнопки «Download» в строке с учетной записью нового пользователя скачать конфигурацию VPN для передачи пользователю;

3) Создать учётную запись пользователя в IDM:
   – перейти по адресу https://idm.platformdomain.local/ipa/ui/ (заменить platformdomain на название актуального домена Платформы) и авторизоваться, используя учётную запись администратора;
   – нажать кнопку «+ Добавить», заполнить анкету, нажать кнопку «Добавить и изменить»;
   – передать пользователю логин и пароль, которые пользователю необходимо сменить при первом использовании Платформы;
   – перейти на вкладку «Группы пользователей» и добавить пользователя в необходимые группы;
4) Создать учётную запись пользователя на сервере администрирования:
   – зайти на BM master с помощью ssh клиента, используя учетную запись имеющую права администратора;
   – создать учётную запись пользователя и добавить её в группы wheel, ansible, docker;
   – внести пользовательский ключ, полученный на шаге 1, в .ssh/Authorized_keys в каталоге пользователя.

3 Описание операций

3.1 Начало использования Платформы и настройка рабочего места пользователя

При первом использовании Платформы нужно выполнить следующие действия:

1) Убедиться в наличии переданных администратором Платформы данных: доменной учётной записи Платформы и конфигурационного файла для Wireguard VPN. При их отсутствии, обратиться за ними к администратору Платформы.

   Примечание – если используется resolved в среде Linux, то нужно внести изменения в конфигурационный файл, заменив строку

   `DNS = 10.255.0.2, platformdomain.local`

   на

   `PostUp = resolvectl dns %i 10.255.0.2; resolvectl domain %i platformdomain.local`

   изменив при этом адрес DNS-сервера и домен Платформы на актуальные для текущей инсталляции. Справку по работе с приложением в среде Linux можно получить с помощью команды wg-quick без указания параметров;

2) Установить переданную конфигурацию в Wireguard VPN клиент, для чего в окне клиента нажать «Add tunnel — Import Tunnel(s) from File…» и выбрать файл, переданный администратором Платформы;

3) Подключиться к серверу Wireguard, нажав кнопку «Activate» в окне «Wireguard»;

4) Скачать сертификат, перейдя по адресу http://pki.platformdomain.local/ca (заменить platformdomain на название актуального домена Платформы) и импортировать его в хранилище корневых сертификатов используемого браузера (если поддерживается) или операционной системы;

5) Перейти на https://idm.platformdomain.local/ipa/ui/ (заменить platformdomain на название актуального домена Платформы) и изменить переданный временный пароль на постоянный;

6) Авторизоваться в gitlab.platformdomain.local и проверить доступность необходимых проектов (при наличии). Обратиться к администратору для предоставления прав к необходимым репозиториям, в частности, gitops.

3.2 Публикация приложения

Для публикации приложения нужно выполнить следующие действия:

1) Авторизоваться на https://gitlab.platformdomain.local используя свою учётную запись;

2) Создать проект. Для чего в разделе с настройками проекта («Settings» — «Repository») в разделе «Protected tags», в поле «Protect a tag — Tag» ввести символ «*» (Create wildcard *) и нажать кнопку «Protect»;

3) Клонировать созданный проект на рабочую станцию, используя протокол ssh;

4) Создать структуру каталогов:

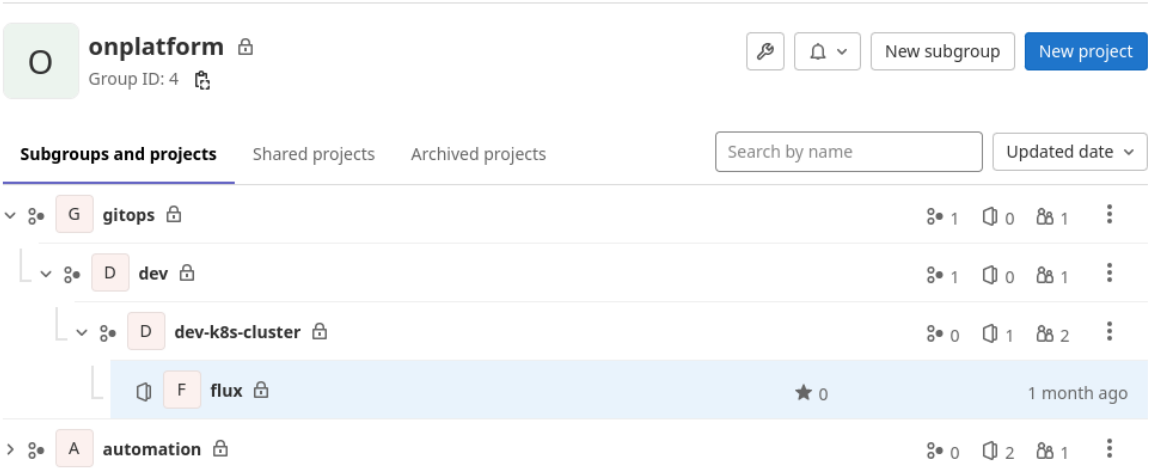
– /src – каталог для исходного кода приложения;
– /helm – каталог для helm чарта;
– /Dockerfile – описание процесса сборки контейнера;
– /.gitlab-ci.yml – пайплайн, составленный соответствии с примером, приведенном в приложении 1.

Рядом с созданными каталогами разместить файлы, которые могут потребоваться для корректной записи приложений, например, скрипты entrypoint, которые будут подключаться в Dockerfile;

5) Создать приложение и helm chart в созданной структуре с учётом требований, предъявляемых Платформой к приложению, при этом:

– запрещено использовать root внутри контейнеров;
– запрещена запись в корневую файловую систему контейнера (необходимо подключить emptyDir разделы в точки файловой системы контейнера, в которые необходимо выполнять запись).

6) Разместить helm release в репозитории нужого кластера «gitops/<cluster>/<cluster>-k8s-cluster/flux»:



4 Приложения

4.1 Приложение 1

Типовая структура файла .gitlab-ci.yml на примере приложения php + nodejs

```
stages:
  - build
  - publish


build app (JS):
  stage: build
  image:
    name: ${ONPLATFORM_CI_REGISTRY}/trkmir/node11-builder:11.15.0-
builder-r0
  variables:
    CI_BUILDER_SRC: ./src
  cache:
    - key:
        files:
          - src/package-lock.json
      paths:
        - src/node_modules/
        - src/.npm/
      policy: pull-push
  script:
    - cd "${CI_BUILDER_SRC}"
    - npm --cache .npm install
    - npm --cache .npm run build
  artifacts:
    paths:
      - src/public/build/
  rules:
    - if: $CI_COMMIT_TAG


build app (PHP):
  stage: build
  image:
    name: ${ONPLATFORM_CI_REGISTRY}/project/imagename-imagetag
```

```
      before_script:
        - echo "${ONPLATFORM_ROOT_CA}" > /usr/local/share/ca-
certificates/onplatform-root-ca.crt
        - update-ca-certificates
        - git config --global url."https://github.com/".insteadOf
"git://github.com/"
        - composer config --global github-protocols https
        - composer config --global "http-basic.${CI_SERVER_HOST}"
"${ONPLATFORM_CI_GIT_USERNAME}" "${ONPLATFORM_CI_GIT_PASSWORD}"
      script:
        - cd ./src
        - composer install -vvv --prefer-dist --no-progress --no-interaction
--no-dev --optimize-autoloader
      artifacts:
        paths:
          - src/vendor/
      rules:
        - if: $CI_COMMIT_TAG


    publish chart:
      stage: publish
      image: ${ONPLATFORM_CI_REGISTRY}/docker.io/instrumentisto/gitlab-
builder:0.9.0
      before_script:
        - echo "${ONPLATFORM_ROOT_CA}" >> /etc/ssl/certs/ca-certificates.crt
      script:
        - PACKAGE="$(helm package helm | cut -d ':' -f2 | tr -d '[:space:]')"
        - curl -u "${ONPLATFORM_CI_USERNAME}:${ONPLATFORM_CI_PASSWORD}" --
upload-file "${PACKAGE}" "${ONPLATFORM_CI_HELM_REPOSITORY}"
      rules:
        - changes:
            - helm/**/*


    publish image:
      stage: publish
      image:
        name: ${ONPLATFORM_CI_REGISTRY}/gcr.io/kaniko-
project/executor:v1.8.1-debug
        entrypoint: [""]
```

```
      before_script:
        - echo "${ONPLATFORM_ROOT_CA}" >> /kaniko/ssl/certs/ca-
certificates.crt
        - mkdir -p /kaniko/.docker
        - echo
"{\"auths\":{\"${ONPLATFORM_CI_REGISTRY}\":{\"auth\":\"$(printf "%s:%s"
"${ONPLATFORM_CI_USERNAME}" "${ONPLATFORM_CI_PASSWORD}" | base64 | tr -d
'\n')\"}}}" > /kaniko/.docker/config.json
      script:
        - >-
          /kaniko/executor
          --context "${CI_PROJECT_DIR}"
          --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
          --destination
"${ONPLATFORM_CI_REGISTRY}/project/${CI_PROJECT_NAME}:${CI_COMMIT_TAG}"
      dependencies:
        - build app (JS)
        - build app (PHP)
      rules:
        - if: $CI_COMMIT_TAG
```