KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект >6 урок > Gitlab CI/CD

> Оглавление

Файлы практики: ссылка

> Введение

Идея CI/CD — при совершении какого-либо события в репозитории запускается цепочка автоматизированных задач.

Для этого системы автоматизации CI/CD (Jenkins, Gitlab CI/CD) предоставляют вычислительные единицы, позволяющие эти задачи осуществить. Например, если необходимо собрать `.cpp` код, на вычислительные единицы должны быть установлены нужные компиляторы.

Автоматизированная CI/CD система состоит из:

- Кода, который пушится/мёржится в репозиторий;
- Вычислительной единицы;
- Задач, автоматизирующих CI/CD.

> Практика

Практика показана в репозитории на Gitlab.

В секции `CI/CD — Pipelines` пока ничего нет, предлагается создать пайплайны.

Также к CI/CD относится ещё одна секция: `Settings — CI/CD`.

Runners из `Settings — CI/CD` — та самая вычислительная единица, которая должна выполнять таски. Для каждой джобы поднимается контейнер, т.е. они изолированны. Передавать файлы и папки между джобами можно при помощи секции `cache`, но в данной лекции она не затрагивается.

Настройки для CI/CD хранятся в файле в корне репозитория. Дефолтное название можно сменить.

Gitlab предлагает возможность использовать готовые шаблоны для `.gitlab-ci.yml`.

Основное понятие CI/CD Gitlab — стейджи (stages). В основном это:

- build
- test
- deploy

После описания стейджей идёт описание тасок с указанием стейджа, где она должна выполняться. Секция script подразумевает описание основного функционала стейджа.

После сохранения файла с конфигурацией, можно увидеть, что появилась запись в Pipelines.

> Дополнительные полезные секции

- `before_script`
- `after_script`

Разницы между этими секциями и другими секциями со скриптами нет: они будут выполняться одним и тем же раннером, но будет соблюдён порядок. Использование секций помогает структурировать код.

+15

> Переменные

Секция `variables` позволяет объявить переменные окружения, которые будут видны глобально.

Если каждый раз менять `yml` файл, то будет создаваться новый пайплайн и в результате будет захламляться история. Чтобы этого избежать, можно, при наличии права maintainer, задать в `Settings — CI/CD — Variables` нужные переменные окружения. Есть выбор:

- Переменная;
- Файл — исходные данные можно закодировать в `base64`.

> Кавычки

Тип кавычек при подключении через `ssh` влияет на то, где будут подставляться переменные:

- Двойные кавычки (" ") — сначала подставляются значения на раннере, а далее команда идёт на удаленный сервер.
- Одинарные кавычки (' ') — сначала команда уходит на удалённый сервер, а потом подставляются значения.

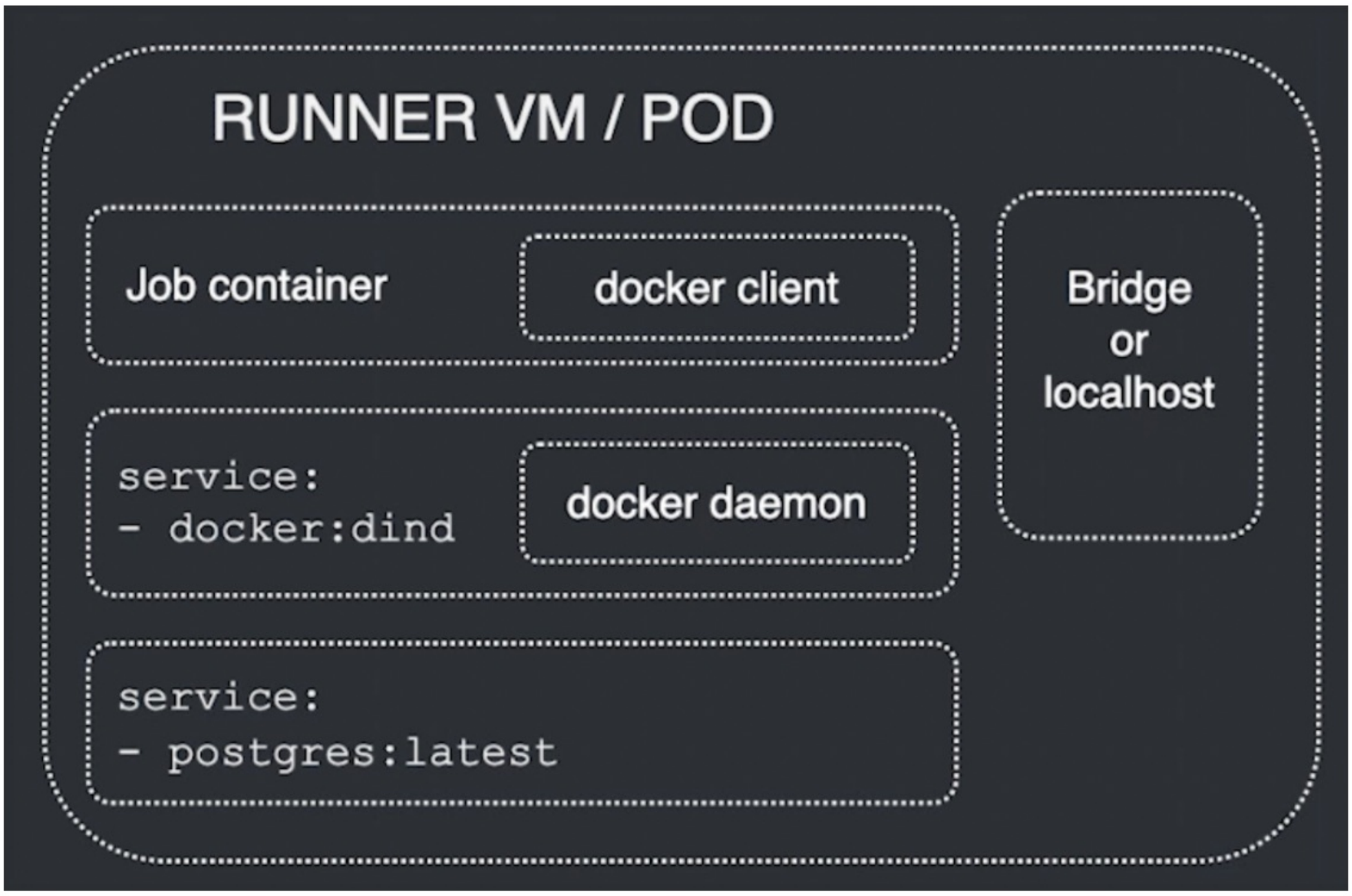
> Контуры

Контур — совокупность всех копий всех приложений, объединённых друг с другом. В Gitlab контур эквивалентен одной ветке.

При создании нескольких веток пайплайны начинают запускаться для каждой из них. Чтобы настроить поведение, можно использовать `rules` в нужной секции.

Для управления переменными разных контуров в Gitlab есть `Deployments — Environments`. Их можно создавать программно при помощи секции `environment`.

> Docker внутри раннера



- Необходимо найти образ для раннера внутри джобы, который содержит docker-клиент.
- Внутри раннера нужен ещё один контейнер с docker engine, к которому можно подключиться по tcp и выполнить нужные команды.
- После выполнения `build` на клиенте, контекст сборки будет отправлен в service dind, и там соберётся нужный образ.