

Состав платформы, инженерные практики и приложения для платформы

Не забудь включить запись!

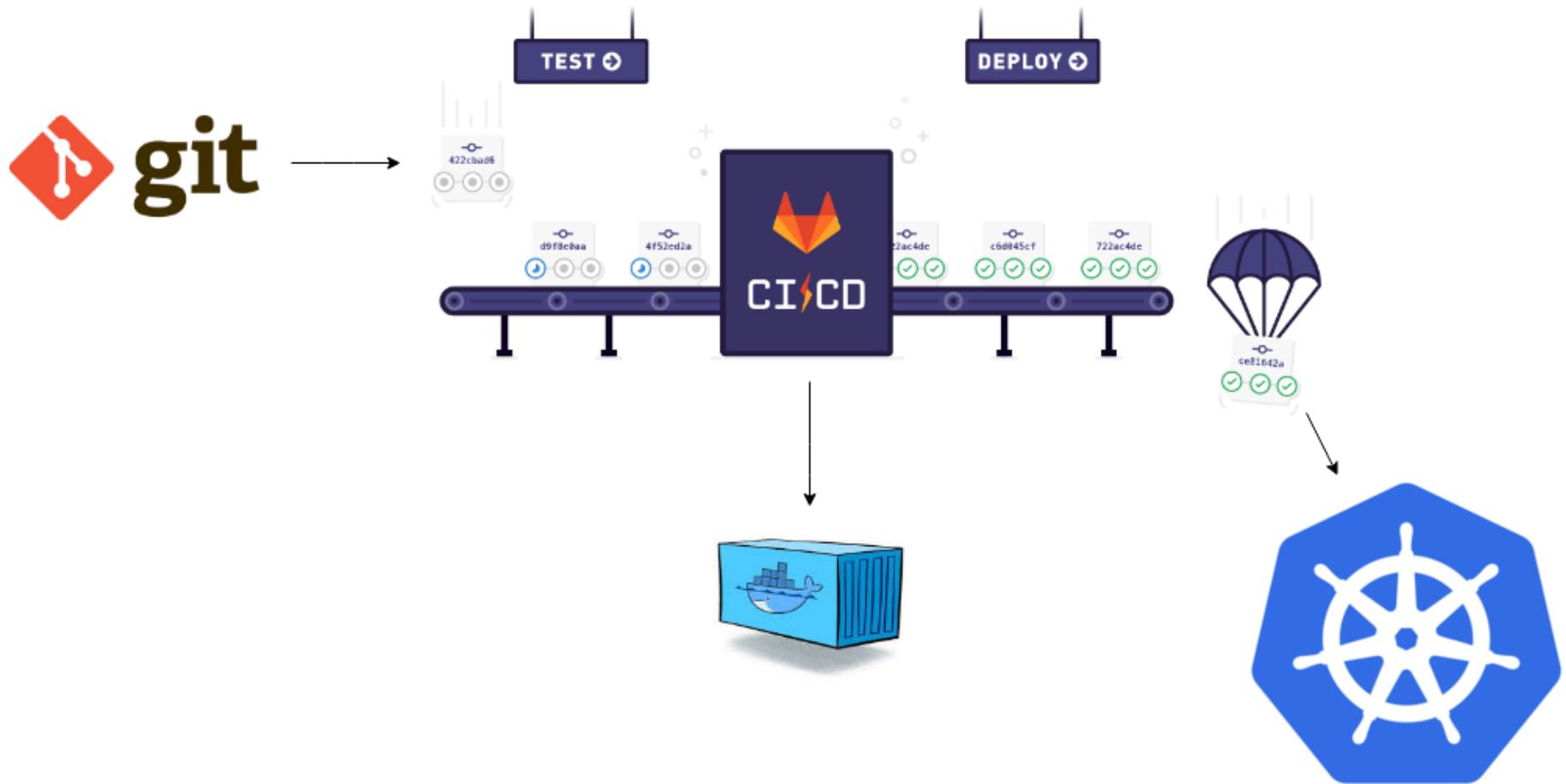


План

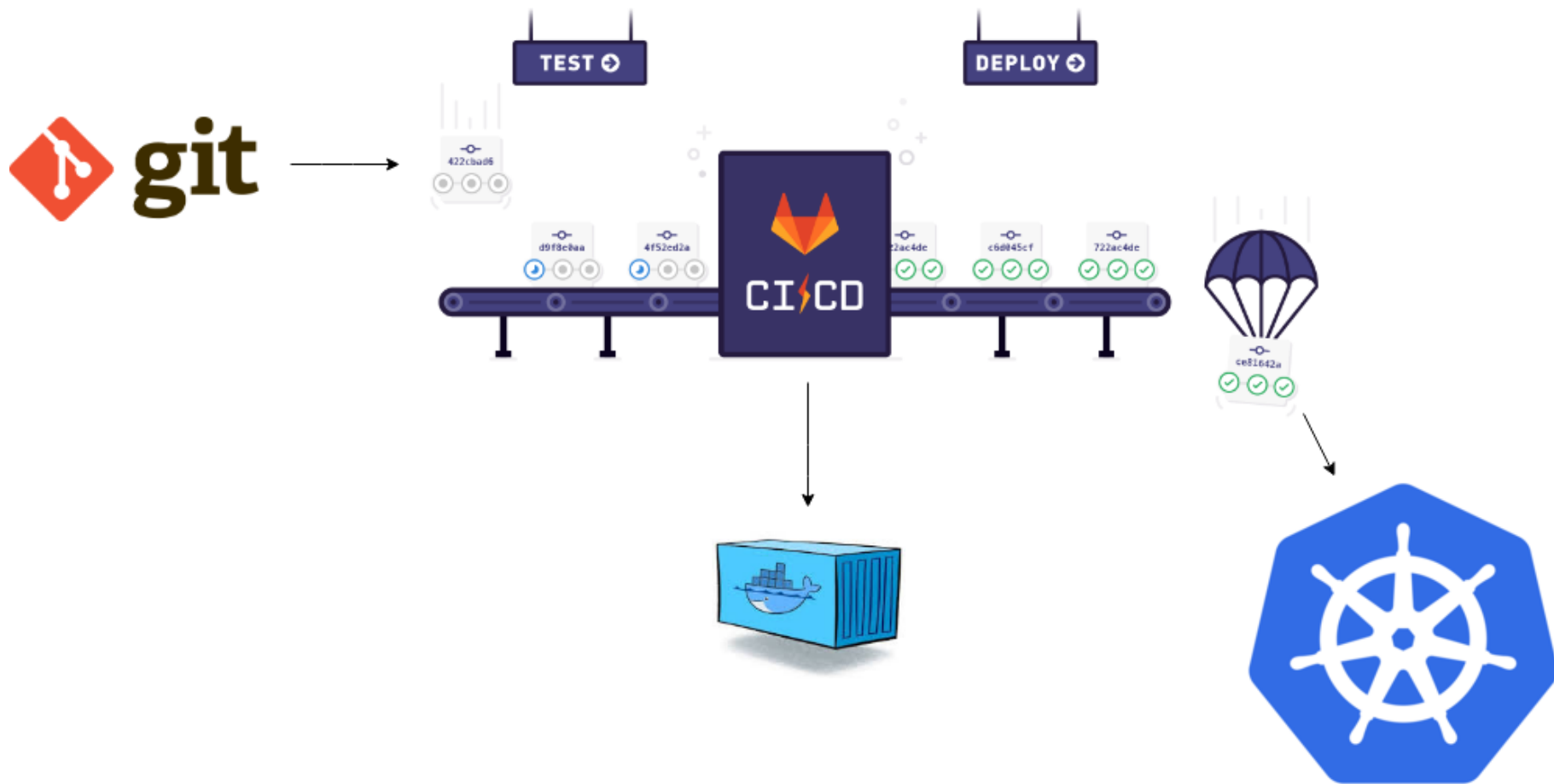
- Компоненты платформы и инженерные практики
- Platform-native applications
- Подготовка к домашней работе

Опрос: Вы и Kubernetes

Сервисы платформы для разработки



CI/CD-сервис



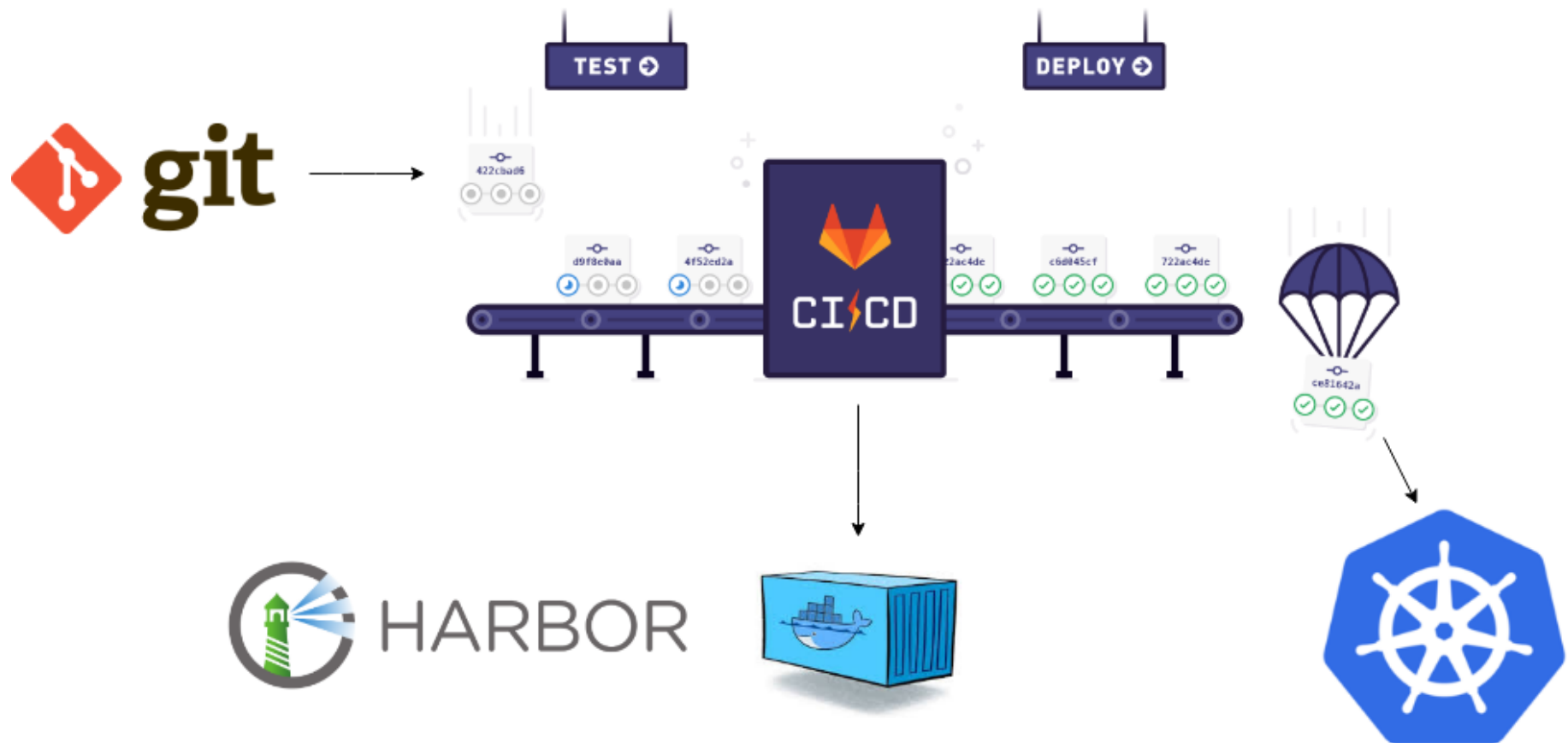
CI/CD-сервис

- Эталонное окружение для сборки и тестирования
- Запуск задач пайплайна поверх общего кластера k8s
- Автомасштабирование сборочных агентов

```
gitlab-runner register
  --non-interactive
  --url "$GITLAB_URL"
  --registration-token "$GITLAB_CI_TOKEN"
  --executor "kubernetes"
  --kubernetes-host "https://$KUBERNETES_URL"
  --kubernetes-ca-file "$KUBERNETES_CA_PATH"
  --kubernetes-bearer_token "$SERVICE_ACCOUNT_TOKEN"
```

[Gitlab Kubernetes executor](#), [Jenkins Kubernetes plugin](#)

Хранилище артефактов



Хранилище артефактов

- Основной формат артефакта – docker image
- Политики жизненного цикла образов
- За содержимое docker image теперь отвечают разработчики
- Проверки безопасности образов (и цифровая подпись)

Хранилище артефактов

Harbor

Search Harbor...

English

admin

Projects

Logs

Administration

- Users
- Registries
- Replications
- Configuration

< Projects < Repositories < library/mariadb

library/mariadb:10.3-signed

Authoranonymity

Architectureamd64

OSlinux

Docker Version17.06.2-ce

Scan CompletedAug 20, 2018

0 High Level Vulnerabilities

4 Medium Level Vulnerabilities

7 Low Level Vulnerabilities

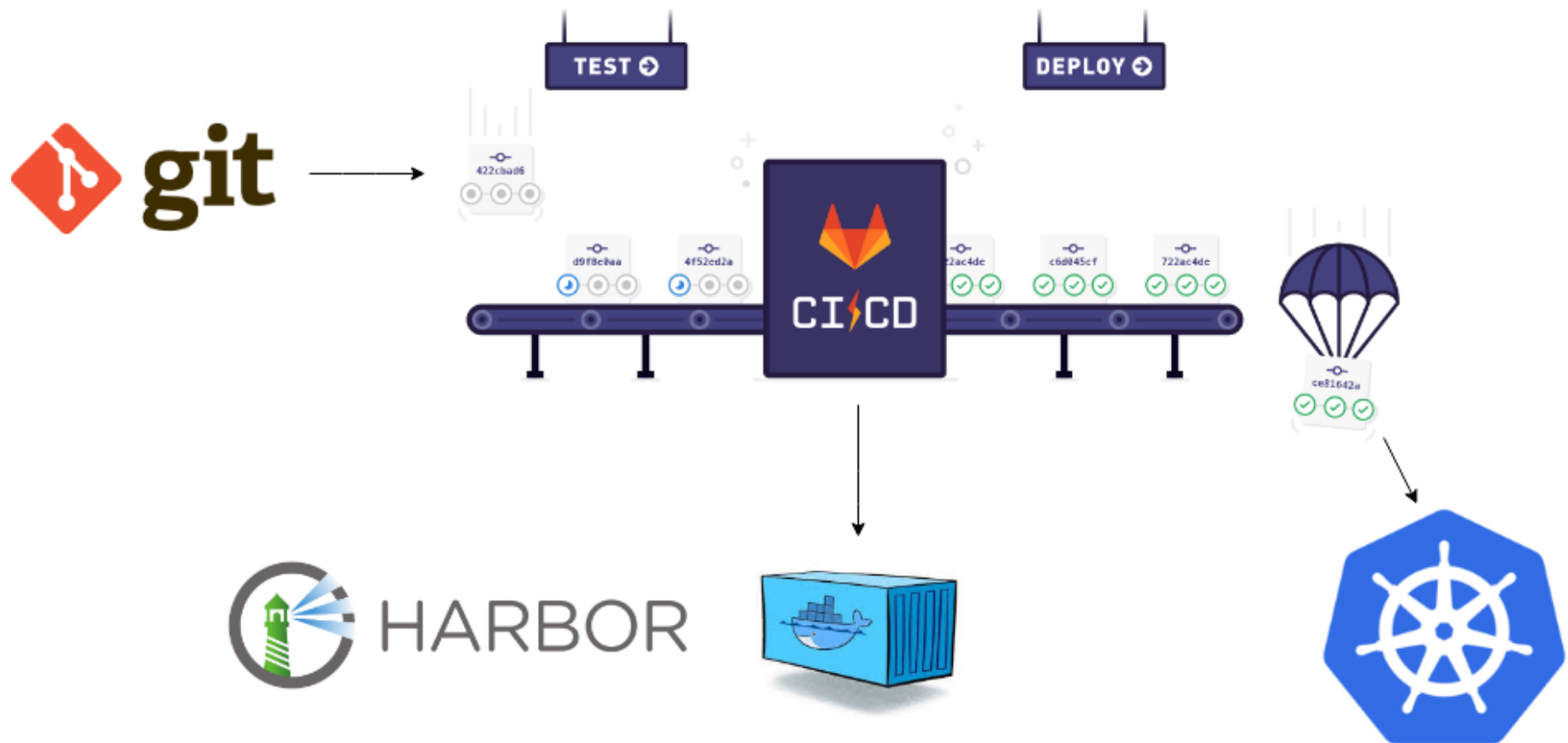
0 Unknown Level Vulnerabilities

SCAN

	Vulnerability	Severity	Package	Current version	Fixed in version
>	CVE-2018-6485	Medium	glibc	2.27-3ubuntu1	
>	CVE-2015-8985	Low	glibc	2.27-3ubuntu1	
>	CVE-2017-1000409	Low	glibc	2.27-3ubuntu1	
>	CVE-2017-1000408	Low	glibc	2.27-3ubuntu1	
>	CVE-2017-16997	Low	glibc	2.27-3ubuntu1	
>	CVE-2017-17426	Medium	glibc	2.27-3ubuntu1	

Harbor container registry

Тестирование



Тестирование

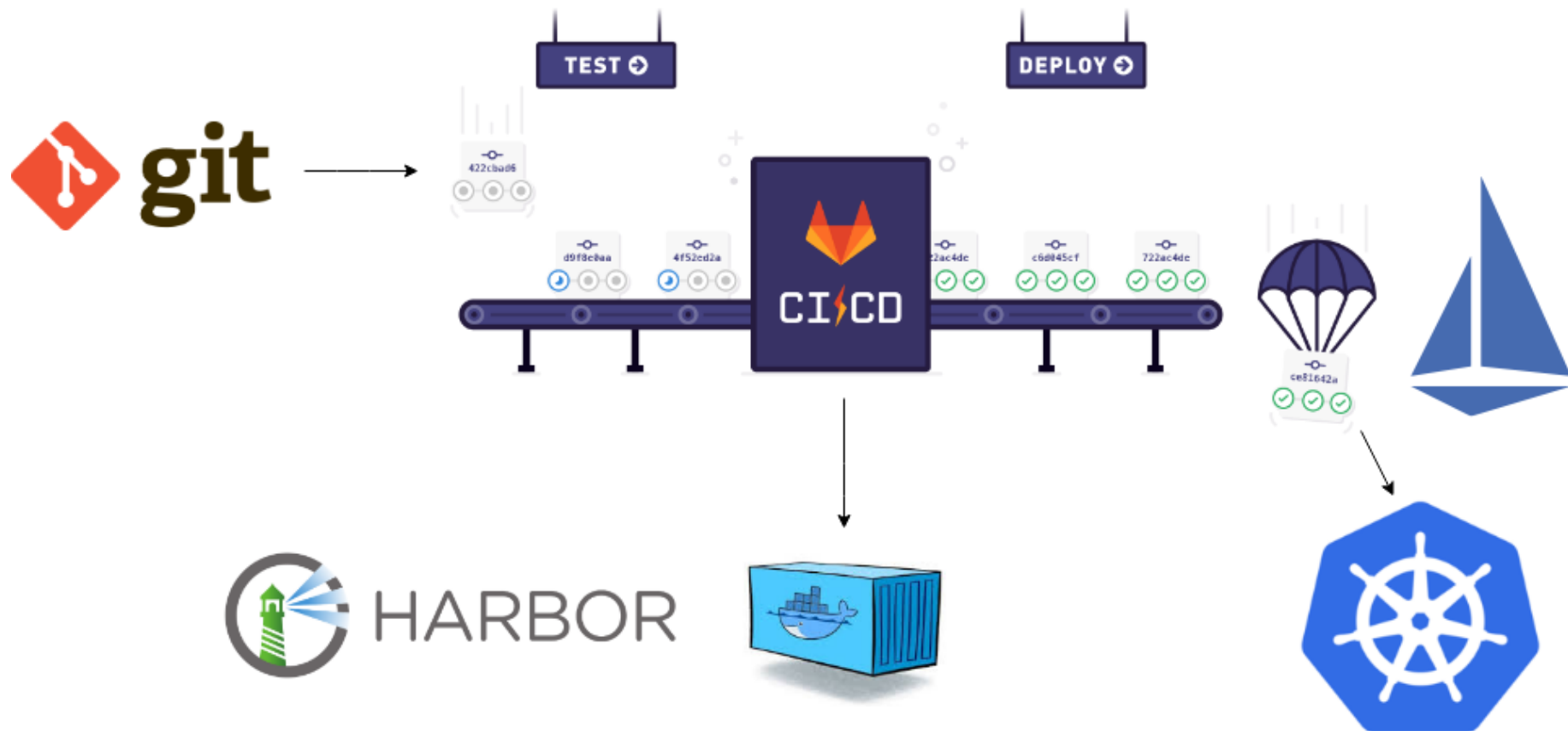
- Динамические окружения для тестов
- Нагрузочное тестирование
- UI-тестирование

selenium-hub-178e1	1/1	Running	0	4h
selenium-node-chrome-s116a	1/1	Running	0	4h
selenium-node-chrome-qx8wm	1/1	Running	0	32s
selenium-node-firefox-715t8	1/1	Running	0	4h

[Running Tsung in Kubernetes](#)

[Selenium Grid on Kubernetes](#)

Обновление без простоев



Обновление без простоев

- Встроенные в k8s стратегии развертывания
- Canary, Focused testing in production

```
kind: VirtualService
metadata:
  name: mobile-api
spec:
  http:
  - route:
    - destination:
        host: mobile-api
        subset: v1
      weight: 95
    - destination:
        host: mobile-api
        subset: v2
      weight: 5
```

```
kind: VirtualService
metadata:
  name: mobile-api
spec:
  http:
  - match:
    - headers:
        cookie:
          regex: "^(*?;)?(email=[^;]*@example.com)(;.*)?$"
      route:
    - destination:
        host: mobile-api
        subset: v1
      weight: 100
```

Monitoring



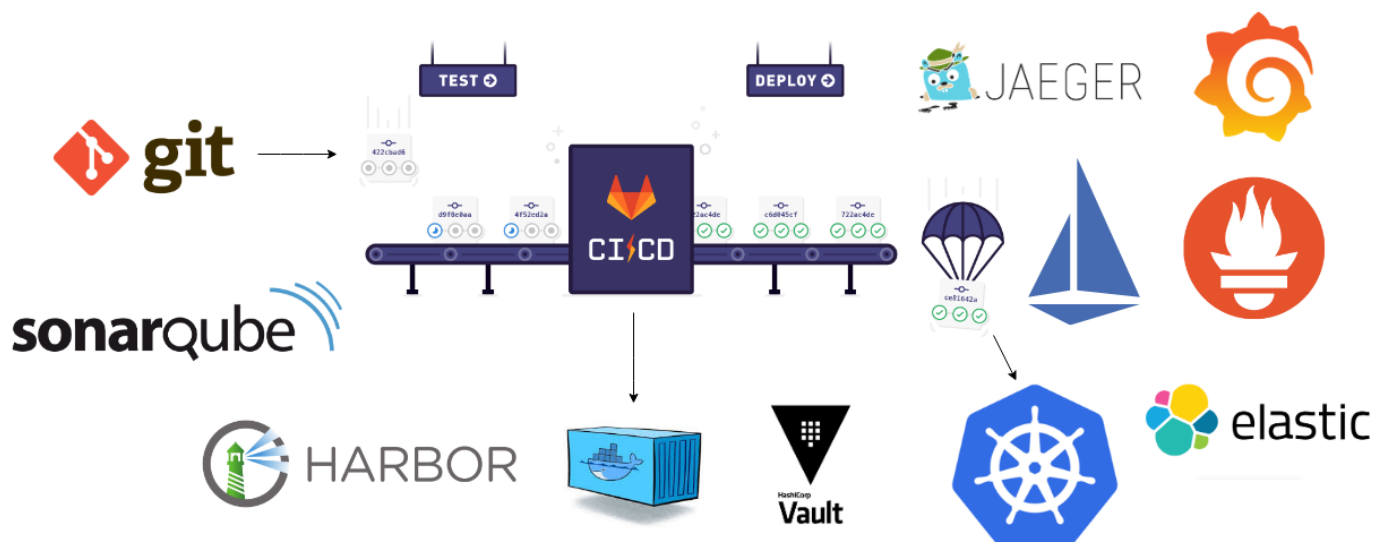
Monitoring

- Традиционный мониторинг в современном мире не работает
- Платформенная команда предоставляет сервис мониторинга
- Команда продукта распоряжается что и как мониторить

```
kind: ServiceMonitor
metadata:
  ...
spec:
  jobLabel: shop-landing
  selector:
    matchLabels:
      app: shop-landing
  targetLabels:
    - shop-landing
  endpoints:
    - port: shop-landing-http
      interval: 10s
```


В составе платформ также бывают

- Tracing
- Feature flags
- Service discovery
- Chaos engineering
- Secrets management
- Code quality and Quality gates



Platform-native applications



THE TWELVE-FACTOR APP

- Рекомендации из 12 factor by Heroku
- Специфика вашей платформы

<https://12factor.net>

I. Кодовая база

- Одна кодовая база и несколько развертываний, не наоборот
- Инфраструктура и пайплайн также описаны кодом (IaC)

```
kind: Deployment
metadata:
  ...
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: example-webapp
        image: registry/example/webapp:v1
        resources:
          requests:
            memory: "128Mi"
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "200m"
```

```
build-image:
  stage: build
  script:
    - docker build
    - docker push

run-tests:
  stage: test
  script:
    - docker pull
    - docker run

deploy-test-env:
  stage: review
  script:
    ...
```

II. Зависимости

- Зависимости явно определены и изолированы
- Docker отлично с этим справляется

```
FROM python:3.7
```

```
RUN apt install imagemagick=8:6.9.10.14
```

```
COPY requirements.txt /app/requirements.txt
```

```
RUN pipenv install -r /app/requirements.txt
```

III. Конфигурация

- Конфигурация отделена от кода и артефакта
- Среда выполнения хранит конфигурацию
- Environment variables / ConfigMaps / Secrets

```
kind: ConfigMap
metadata:
  name: shop-api-configmap
data:
  database: 'postgresl'
  database_uri: 'postgresql://postgres-master:5432'
```

VI. Приложение – процесс и только

- Приложения не хранят состояние (stateless)
- Все обращения к данным происходят по сети (share nothing)

VII. Привязка портов

- Каждое приложение экспортирует сетевой порт
- Взаимодействие приложений только по сети

```
kind: Service
metadata:
  ...
spec:
  type: ClusterIP
  ports:
    - name: shop-landing-http
      port: 8000
      protocol: TCP
  selector:
    app: landing
```

VIII. Параллелизм

- Приложение декомпозировано на процессы
- Каждый процесс масштабируется независимо
- Предпочтительнее горизонтальное масштабирование

```
kind: Deployment
metadata:
  ...
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: avatar-resizer
```

```
kind: Deployment
metadata:
  ...
spec:
  replicas: 10
  template:
    spec:
      containers:
      - name: avatar-resizer
```


IX. Утилизируемость

- Запуск приложения не занимает длительного времени
- Приложение корректно обрабатывает сигналы ОС
- Приложение можно остановить в любой момент без последствий

XI. Журналирование

- Логи всех приложений – единый поток событий
- Приложение не занимается хранением и ротацией логов
- Все логи пишутся только на stdout / stderr
- В логах одновременно могут быть сообщения от разных версий
- Используйте структурированные логи по-возможности

XII. Задачи администрирования

- Административные процессы запускаются на том же кластере
- Но с другим подходом: с помощью разовых процессов

```
kind: Job
...
containers:
- name: landing
  image: registry/example/webapp:v1
  command: ['python', 'manage.py', 'migrate']
```

```
kind: CronJob
...
schedule: "0 0 1 * *"
...
containers:
- name: landing
  image: registry/example/webapp:v1
  command: ['python', 'manage.py', 'generate-report']
```

XXX: Интеграция с платформой

- Liveness and Readiness checks

```
kind: Deployment
metadata:
  ...
  containers:
  - name: example-webapp
    image: registry/example/webapp:v1
    livenessProbe:
      httpGet:
        path: /internal/platform/liveness
        port: 8000
    readinessProbe:
      httpGet:
        path: /internal/platform/readiness
        port: 8000
```

XXX+1: Интеграция с платформой

```
import os
import hvac

client = hvac.Client(
    url=os.environ['VAULT_URL'],
    token=os.environ['VAULT_TOKEN']
)

postgresql_password = client.read('secret/postgresql/password')
```

Подготовка к выполнению домашних заданий

Подключение GitHub аккаунта к платформе Отус

- Зайдите в [личный кабинет Отус](#)
- Нажмите на иконку **Octocat** и авторизуйте приложение ОТУС на GitHub.

Аккаунты в соцсетях



Facebook
Не привязан



ВКонтакте
Не привязан



Github
Не привязан



Mail.Ru
Не привязан

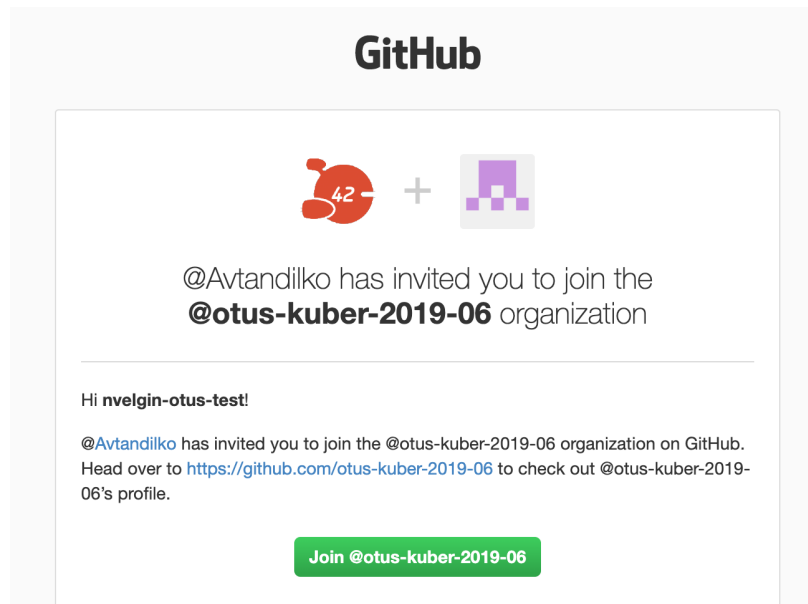


Google
Не привязан

Работа с инфраструктурой

Через некоторое время (перед первой домашней работой) вам на email **придут приглашения**:

1. Вступить в организацию [otus-kuber-2019-06](#)
2. В этой организации поработать с репозиторием `<YourGithubLogin>_platform`



Работа с инфраструктурой

Примите приглашения. После этого у вас должен появиться write доступ к репозиторию `<YourGithubLogin>_platform`, то есть возможность делать commit в ветки, отличные от master, и создавать PR.

The screenshot shows a GitHub repository page for 'otus-kuber-2019-06 / express42-test_platform'. The repository has 1 commit, 1 branch, 0 releases, 1 contributor, and 0 stars. The 'Code' tab is selected, showing the repository's structure. The 'LICENSE' and 'README.md' files are listed as initial commits from 'Avtandilko' 5 days ago. The 'README.md' file content is displayed below, showing the repository name 'express42-test_platform' and its description 'express42-test Platform repository'.

otus-kuber-2019-06 / [express42-test_platform](#) Watch 1 Star 0 Fork 0

Code Pull requests 0 Security Insights

express42-test Platform repository

1 commit 1 branch 0 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find File Clone or download

Avtandilko Initial commit Latest commit 81d6c14 5 days ago

File	Commit	Time
LICENSE	Initial commit	5 days ago
README.md	Initial commit	5 days ago

README.md

express42-test_platform

express42-test Platform repository

Общий flow работы

git clone

Для выполнения ДЗ необходимо клонировать репозиторий `<YourGithubLogin>_platform` на локальную машину

The screenshot shows the GitHub interface for a repository named 'express42-test Platform repository'. At the top, it indicates '1 commit', '1 branch', '0 releases', '1 contributor', and the 'MIT' license. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and a prominent green 'Clone or download' button. A dropdown menu is open from the 'Clone or download' button, showing options to 'Clone with SSH' (with a 'Use HTTPS' link) and 'Open in Desktop' or 'Download ZIP'. The SSH command 'git@github.com:otus-kuber-2019-06/exp' is visible in the dropdown. The repository content shows an 'Initial commit' by 'Avtandilko' with files 'LICENSE' and 'README.md'. The repository name 'express42-test_platform' is displayed in a large font, followed by the description 'express42-test Platform repository'.

Общий flow работы

git checkout

Каждое ДЗ выполняется в отдельной ветке, соответствующей названию данного ДЗ (будет указано в материалах).

Например, для выполнения задания по лекции **kubernetes-intro** потребуется создать ветку **kubernetes-intro**:

```
git checkout -b kubernetes-intro
```

Общий flow работы


Pull Request


Когда вы чувствуете, что задание выполнено, необходимо сдать его на проверку. На нашем курсе мы проводим приемку ДЗ через Pull Request.

Создание Pull Request из ветки, в которой вы делали работу в ветку **master**:

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).


 base: master


 compare: kubernetes-intro


✓ Able to merge. These branches can be automatically merged.


Create pull request


Discuss and review the changes in this comparison with others.



 1 commit

 2 files changed


 0 commit comments

 1 contributor

 Commits on Jul 01, 2019

  nvelgin-otus-test

Demo

 37d2629

Общий flow работы

Travis CI

После создания PR запустится проверка (Travis CI):

Add more commits by pushing to the **kubernetes-intro** branch on **otus-kuber-2019-06/nvelgin-otus-test_platform**.



Some checks haven't completed yet

[Hide all checks](#)

1 in progress and 1 successful checks



Travis CI - Pull Request *In progress — Build Started*

Required

[Details](#)



Travis CI - Branch *Successful in 3m — Build Passed*

[Details](#)



Required statuses must pass before merging

All required [statuses](#) and check runs on this pull request must run successfully to enable automatic merging.

Merge pull request




You can also [open this in GitHub Desktop](#) or view [command line instructions](#).


Общий flow работы

Приемка ДЗ



PR прошел автоматизированные тесты и merge в master может быть сделан без вмешательства проверяющих:

Add more commits by pushing to the **kubernetes-intro** branch on **otus-kuber-2019-06/nvelgin-otus-test_platform**.





**All checks have passed**[Hide all checks](#)


2 successful checks




Travis CI - Branch Successful in 3m — Build Passed[Details](#)



Travis CI - Pull Request Successful in 3m — Build Passed **Required** [Details](#)



This branch has no conflicts with the base branch
Merging can be performed automatically.


Merge pull request 


You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Общий flow работы



Приемка ДЗ



PR не прошел автоматизированные тесты и merge в master запрещен, необходимо доработка:




**All checks have failed**[Hide all checks](#)

2 action required checks

**Travis CI - Branch** Action required after 8m — Build Errored[Details](#)

**Travis CI - Pull Request** Action required after 8m — Build Errored

Required [Details](#)

**Required statuses must pass before merging**

All required [statuses](#) and check runs on this pull request must run successfully to enable automatic merging.

Merge pull request ▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Избавляем бизнес от ИТ-зависимости

42

23.10

Общий flow работы

Приемка ДЗ


Некоторые задания после проверки автоматизированными тестами потребуют ручной проверки преподавателем. Для таких заданий мы:

1. Укажем этот факт в руководстве по ДЗ
2. В build log Travis CI уточним, что тесты выполнены успешно и необходимо ожидать ручной проверки:


```
+echo 'All tests passed. Proceed with manual approve'
All tests passed. Proceed with manual approve
+exit 1
The command "curl https://raw.githubusercontent.com/express42/otus-platform-tests/2019-06/run.sh | bash" failed and exited with 1 during .
```



Общий flow работы


В таком случае установите на PR метку **review required**.
Данный PR будет проверен преподавателем, merge также сделает преподаватель.

Reviewers 


No reviews

Assignees 

 **Avtandilko**

Labels 

review required

Projects 

Очередь проверки in Homeworks

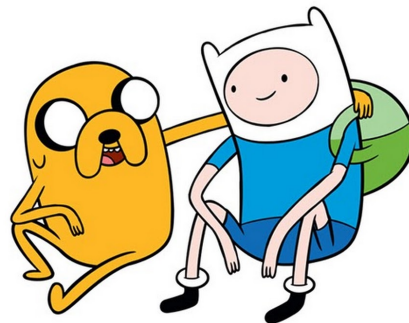
Общий flow работы

В организации есть доска, на которой удобно отслеживать процесс проверки PR

The screenshot displays the GitHub interface for the repository **otus-kuber-2019-06**. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. Below the repository name, there are tabs for Repositories (3), Packages, People (8), Teams (2), Projects (1), and Settings. The main section is titled **Homeworks** and shows a Kanban board with four columns:

- Возвращенные на доработку** (0 items)
- Очередь проверки** (1 item): A pull request titled **Kubernetes intro** is shown. It has a status of **0 of 4** and a label **review required**. The pull request was opened by **nvelgin-otus-test_platform#12** and **nvelgin-otus-test**.
- Проверено** (0 items)
- Готово** (0 items)

At the top of the board, there is a search bar labeled **Filter cards** and buttons for **Add cards**, **Fullscreen**, and **Menu**.



Спасибо за внимание!

Время для ваших вопросов!