??????????

Лучше тестовая среда - предпрод среда - прод среда

Виды тестирования

E2E	Тестирование основного бизнес функционала. Объединенная работа нескольких сервисов
Системное тестирование	Тестирование одного сервиса. Обычно ручное. Варианты: • На базе требований: тесты на основании требований к ПО • На базе действий: тесты на основании использовании ПО
Интеграционное тестирование	Тестирование взаимодействий сервисов
Unit тестирование	Проверка корректности отдельных модулей
Mock тестирование	Фиктивная реализация интерфейса для тестирования

Quality gate - модуль тестирования

Выполнение скрипта с кодом возврата:

```
tasks:
  - script: test_script_1
  - script: test_script_2 --parameter1 value1 --parameter2 value2
```

Проверка наличия файлов модулем stat

```
tasks:
  - stat:
      path: /path/to/something
      register: p
  - assert:
      that:
      - p.stat.exist and p.stat.isdir
```

Использование модуля assert

```
tasks:
  - shell: /usr/bin/some_command --some_parameter value
    register: cmd_result
  - assert:
    that:
    - "'not ready' not in cmd_result.stderr"
    - "'gizmo enabled' in cmd_result.stdout"
```

Жизненный цикл тестирования:

- Использовать один и тот же playbook при тестировании всех сред, включая production
- Запускайте тестирование каждый раз
- Использовайте тесты, написанные командой обеспечения качества
- Используйте те же самые тесты при деплое в production

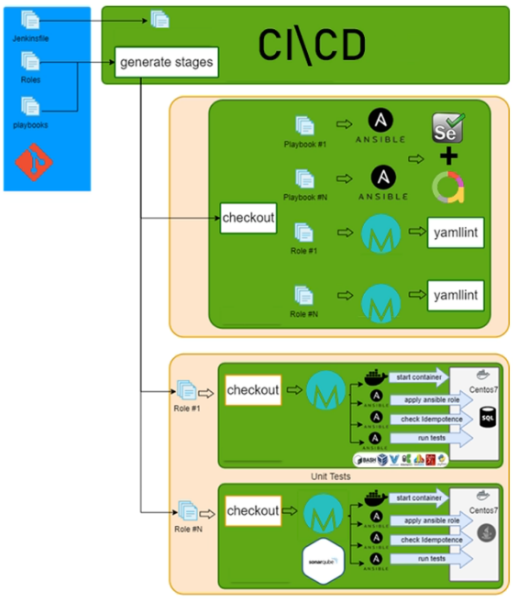
Достижение непрерывного развертывания

- Использовать автоматизацию при развертывании виртуальных машин (развертывание по кнопке)
- При помощи CI системы развертывайте на тестовой среде
- Одна из задач при деплое вызывает тестирующий скрипт со статусом Прошел/ не прошел перед каждым деплоем
- Если деплой прошел успешно, то сборка повторяется на production контуре

Пример схемы CI/CD

Пример схемы CI\CD

- Repo Git
- Build CI + Unit test + QG + Sonar report + docker
- Deploy CD + Ansible + QG + Mock\integration test (selenium+molecula) + Allure report



Molecule

Поддерживает текущую версию и -1 промежуточной (второй цифры). Установка:

```
pipx install molecule
```

Инициализация molecule

В директории с папкой roles выполнить

```
molecule init scenario name_of_role
```

При инициализации создается несколько файлов в molecule/name_of_role/

Назначения файлов

molecule.yml	Конфигурация запуска фреймворка. Основной файл. Блоки настройки: dependency: управление зависимостями. driver: управление сервером тестирования. По умолчанию docker. Устанавливаются отдельно. platforms: конкретный элемент для запуска provisioner: инструмент запуска converge.yml verifier: инструмент запуска verify.yml для проверки перехода в нужное состояние. lint: инструмент поиска синтаксических ошибок. scenario: жизненный цикл тестов. Может быть несколько.
converge.yml	Playbook для накатывания роли
verify.yml	Тестирование того, что роль успешно применилась

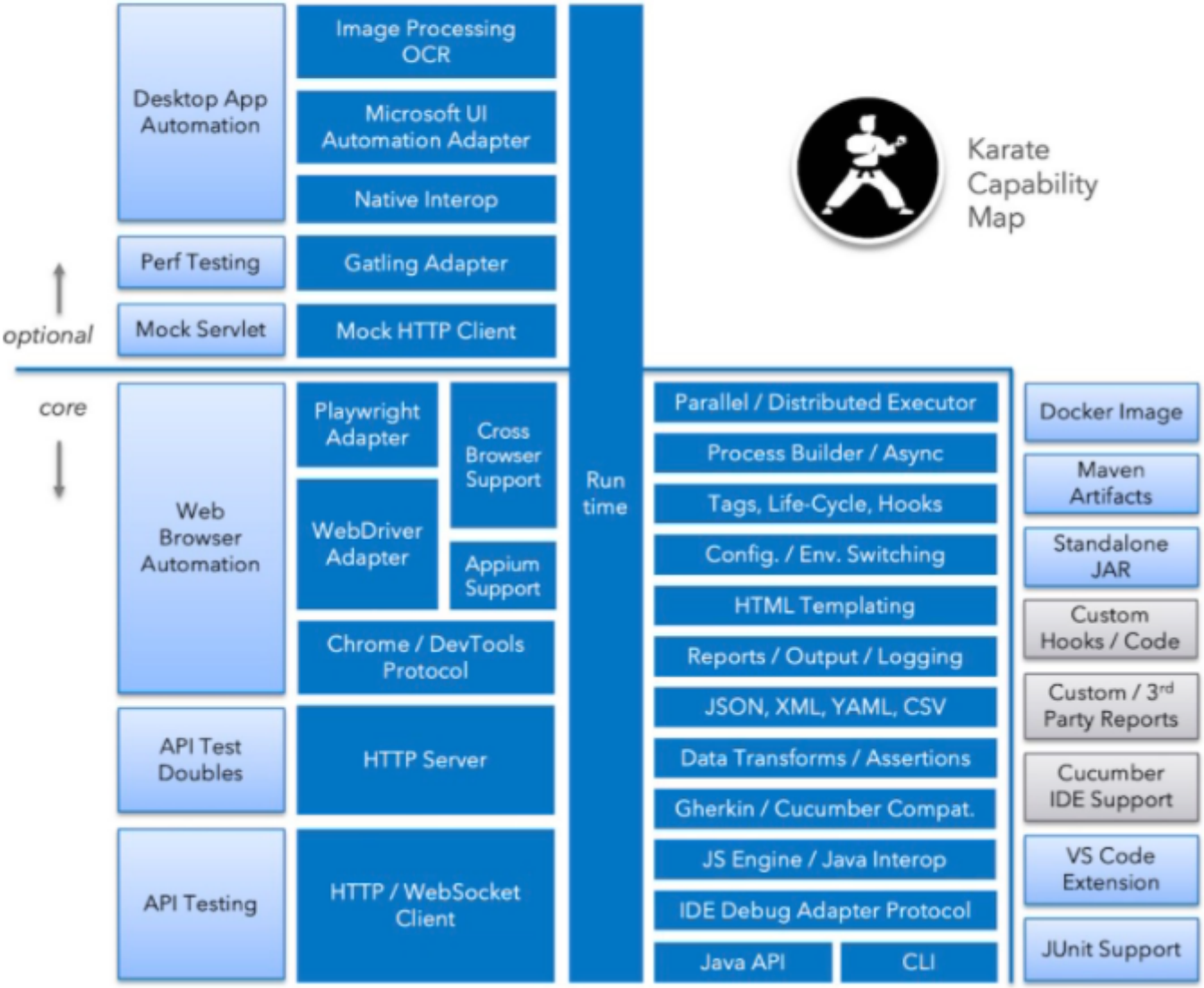
В сценариях используются еще блоки, их называют программами.

molecule test запускает test_sequence

Karate

OpenSource инструмент для тестирования, удобно тестировать API

Структура Karate



Формирует отчеты.

Revision #4
Created 4 June 2026 13:25:53 by Admin
Updated 8 June 2026 15:33:58 by Admin