
QA
מע

אלכס גורבצ'וב
АЛЕКСЕЙ ГОРБАЧЁВ



INTRODUCTION TO AUTOMATION

ВВЕДЕНИЕ В АВТОМАТИЗАЦИЮ



Введение в автоматизацию

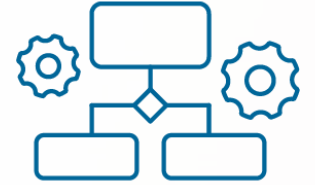


Автоматизация тестирования (test automation) – это процесс использования программных средств для выполнения тестовых сценариев, позволяющий исключить человека из выполнения **некоторых** задач в процессе тестирования.

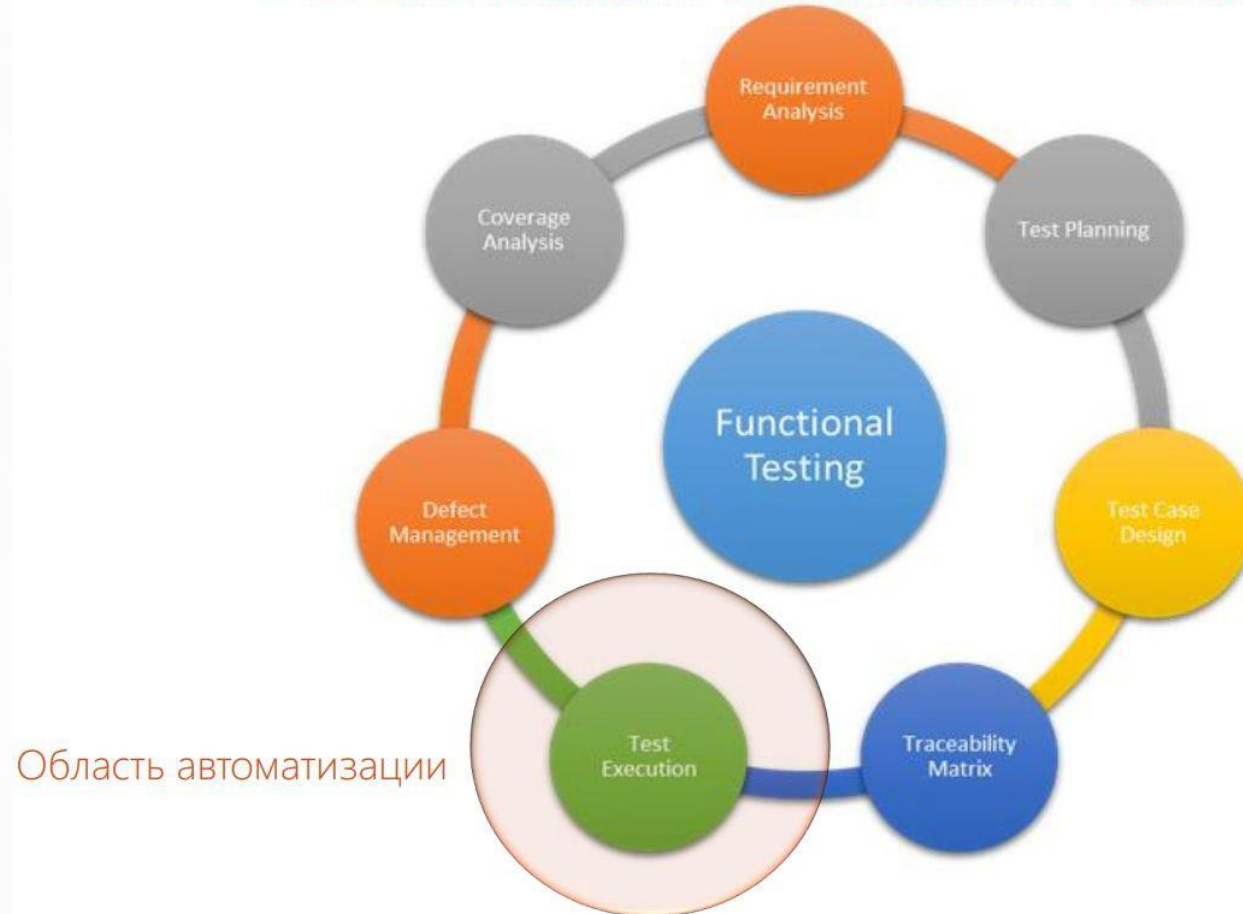
Проверить, качественный ли продукт и соответствует ли он ожиданиям, можно вручную или с помощью автоматизации тестирования. В первом случае QA-инженер воспроизводит действия пользователя и фиксируют ошибки, если таковые имеются.

Во втором случае запуск, анализ, выдача результата происходят автоматически, с использованием ПО. IT-специалист лишь обрабатывает собранную информацию. Поэтому автоматизация тестирования - это своего рода инструмент оптимизации процессов.

Введение в автоматизацию



Автоматизация в рамках ОА процессов



Введение в автоматизацию



Зачем нужна автоматизация тестирования?

Автоматизация тестирования позволяет улучшить качество и скорость тестирования, а также сократить затраты на тестирование.

Когда тестирование выполняется вручную, это может быть очень трудоемким процессом. Например, если тестирование включает множество шагов, которые необходимо повторить несколько раз, это может занять много времени и сил. Кроме того, при ручном тестировании возникает риск ошибок, так как человек может пропустить какую-то деталь или не заметить ошибку.

Автоматизация позволяет сократить время тестирования, увеличить точность тестирования и уменьшить риск ошибок.

Введение в автоматизацию



Введение в автоматизацию

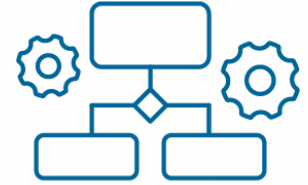


Каким тест-кейсам полезна автоматизация тестирования?

В целом, все тест-кейсы можно автоматизировать. Однако, не все тест-кейсы целесообразно автоматизировать. Следующие типы тестовых сценариев полезно автоматизировать:

- Тесты, которые необходимо повторять несколько раз
- Тесты, которые сложны для выполнения вручную
- Тесты, которые занимают много времени
- Тесты, которые требуют большого объема входных данных
- Тесты, которые требуют быстрого выявления ошибок

Введение в автоматизацию



Области высокой эффективности автоматизации тестирования

Длительные, рутинные, утомительные для человека и/или требующие повышенного внимания операции.

Конфигурационное тестирование и тестирование совместимости

Приложения (или их части) без графического интерфейса.

Использование комбинаторных техник тестирования.

Настройка тестового окружения.

Модульное тестирование.

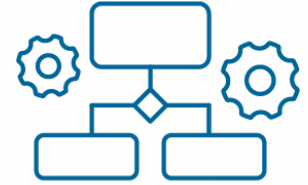
Тестирование безопасности.

Интеграционное тестирование.






Регрессионное тестирование.

Тестирование производительности.

Введение в автоматизацию



Инструменты автоматизации

Product	 Selenium	 Katalon Studio	 Unified Functional Testing	 TestComplete	 watir
Available since	2004	2015	1998	1999	2008
Application Under Test	Web apps	Web (UI & API), Mobile apps	Web (UI & API), Mobile, Desktop, Packaged apps	Web (UI & API), Mobile, Desktop apps	Web apps
Pricing	Free	Free	\$\$\$\$	\$\$	Free
Supported Platforms	Windows Linux OS X	Windows Linux OS X	Windows	Windows	Windows Linux OS X
Scripting languages	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++ and C#	Ruby
Programming skills	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Advanced skills needed to integrate various tools
Ease of Installation and Use	Require advanced skills to install and use	Easy to setup and use	Complex in installation. Need training to properly use the tool	Easy to setup. Need training to properly use the tool	Advanced skills needed to integrate various tools

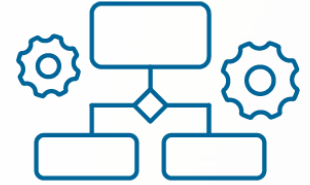
Введение в автоматизацию



Инструменты автоматизации

- Selenium — наиболее популярный инструмент для автоматизации веб-приложений. Selenium можно использовать для написания тестов на многих языках программирования, включая Java, Python, C# и Ruby.
- Appium — инструмент для автоматизации тестирования мобильных приложений на платформах Android и iOS. Appium использует WebDriver протокол для автоматизации действий на устройствах.
- JMeter — инструмент для тестирования производительности и нагрузочного тестирования веб-приложений. JMeter может использоваться для тестирования различных протоколов, включая HTTP, HTTPS, FTP, SMTP и JDBC.
- TestComplete — инструмент для автоматизации тестирования десктопных, мобильных и веб-приложений. TestComplete предоставляет множество готовых модулей для автоматизации тестирования, а также позволяет написать свои собственные скрипты на языках JavaScript, Python и VBScript.

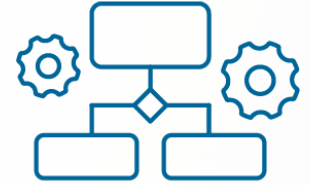
Введение в автоматизацию



Инструменты автоматизации

- Robot Framework — фреймворк для автоматизации тестирования, позволяющий писать тесты на многих языках программирования, включая Python, Java, .NET и Perl. Robot Framework поддерживает различные протоколы, такие как HTTP, FTP, SSH, JDBC и другие.
- Cucumber — инструмент для автоматизированного тестирования приложений на основе поведения. Cucumber позволяет писать тесты на естественном языке, который понятен как разработчикам, так и тестировщикам.
- Postman — инструмент для тестирования API и создания запросов. Postman позволяет тестировать API без написания кода, а также автоматизировать тесты с помощью JavaScript.

Введение в автоматизацию



Инструменты автоматизации

- SoapUI — инструмент для автоматизации тестирования веб-сервисов на основе протокола SOAP. SoapUI поддерживает различные протоколы, такие как HTTP, JMS, AMF и JDBC.
- LoadRunner — инструмент для тестирования производительности и нагрузочного тестирования веб-приложений. LoadRunner поддерживает множество протоколов, включая HTTP, FTP, SMTP и LDAP.
- Katalon Studio — инструмент для автоматизации тестирования веб-приложений, мобильных приложений и API. Katalon Studio предоставляет графический интерфейс

Введение в автоматизацию



Автоматизация тестирования алгоритм действий

Алгоритм действий при автоматизации тестирования может быть следующим:

1. Определение целей: определение задач, которые должны быть автоматизированы, и оценка возможных выгод от автоматизации.
2. Выбор инструментария: выбор инструментов, которые будут использоваться для автоматизации тестирования. Он зависит от особенностей тестируемого приложения и технических возможностей команды тестировщиков.
3. Создание тестовых сценариев: создание тестовых сценариев для автоматизации, учитывая особенности приложения и требования к тестированию.
4. Разработка тестовых скриптов: создание скриптов для автоматизации выполнения тестовых сценариев. В этом процессе тестировщик создает скрипт на выбранном инструменте, который будет эмулировать действия пользователя в приложении.

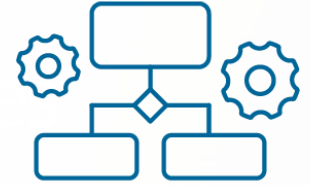
Введение в автоматизацию



Автоматизация тестирования алгоритм действий

5. Настройка окружения: настройка окружения тестирования, включая установку необходимых программных компонентов, баз данных, настройку серверов и т.д.
6. Запуск тестовых скриптов: запуск тестовых скриптов в автоматическом режиме и сбор результатов.
7. Анализ результатов: анализ результатов тестирования и сравнение их с ожидаемыми результатами. В случае обнаружения ошибок или несоответствий, тестировщик должен создать отчет о баге и отправить его разработчикам для исправления.
8. Поддержка и обновление: поддержка и обновление автоматизированных тестов при изменении приложения или его функциональности.

Введение в автоматизацию



Рекомендации для эффективной автоматизации

1. Определите, какие тест-кейсы будут автоматизированы. Не все тест-кейсы должны быть автоматизированы, поэтому необходимо выбрать те тесты, которые требуют больше времени и усилий при ручном тестировании, или тесты, которые необходимо повторять многократно.
2. Выберите подходящий инструмент для автоматизации. Существует множество инструментов для этого процесса, каждый из которых имеет свои преимущества и недостатки. Необходимо выбрать инструмент, который лучше всего соответствует требованиям проекта.
3. Создайте хорошо организованную структуру тестов. Эффективная структура тестов поможет избежать дублирования тестов и сократить время на разработку тест-кейсов. Тесты должны быть легко понятны и доступны для всех членов команды.

Введение в автоматизацию



Рекомендации для эффективной автоматизации

4. Используйте модульное тестирование. Модульное тестирование позволяет тестировать каждый модуль отдельно, что упрощает процесс отладки и позволяет избежать проблем, связанных с зависимостью модулей друг от друга.
5. Напишите надежный код тестов. Код тестов должен быть написан таким образом, чтобы он был легко понятен и изменяем. Код тестов должен быть также хорошо организован и поддерживаем.

Введение в автоматизацию



Особенности автоматизированного тестирования

1. Необходимость постоянного обновления тестовых скриптов: любое изменение в приложении может привести к тому, что ранее написанные тесты станут нерелевантными. Поэтому тестовые скрипты должны быть постоянно обновляемыми, чтобы отражать текущее состояние приложения.
2. Необходимость высокой степени автоматизации: для того чтобы автоматизация была эффективной, необходимо автоматизировать как можно большее количество тестов. Это позволяет существенно сократить время на тестирование и увеличить покрытие тестами.
3. Необходимость подготовки тестовых данных: для успешного автоматизированного тестирования необходимо создать тестовые данные, которые будут использоваться в тестах. Это может быть трудоемкой задачей, особенно если приложение имеет сложную структуру данных.

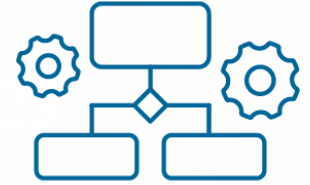
Введение в автоматизацию



Особенности автоматизированного тестирования

4. Необходимость правильного выбора инструментов: выбор инструментов для автоматизации тестирования является очень важным. Необходимо выбирать инструменты, которые подходят для конкретного проекта и которые имеют необходимую функциональность.
5. Необходимость обучения: автоматизация тестирования требует знаний и опыта в области программирования и тестирования. Поэтому необходимо обучить тестировщиков, которые будут работать с автоматизированными тестами.

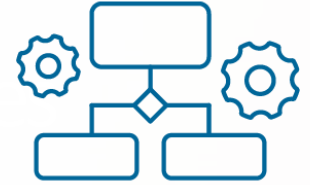
Введение в автоматизацию



Плюсы автоматизированного тестирования

1. Ускорение тестирования: автоматизация упрощает процесс тестирования и позволяет выполнить большое количество тестов за короткий промежуток времени.
2. Повышение качества: автоматическое выполнение тестов исключает возможность человеческой ошибки, что повышает точность и надежность тестирования.
3. Экономия времени и ресурсов: автоматические тесты выполняются быстрее, чем тесты, проводимые вручную, что позволяет экономить время и снижать затраты на тестирование.
4. Повторяемость: автоматические тесты могут быть запущены в любое время без необходимости повторного написания тест-кейсов, что обеспечивает повторяемость и консистентность тестирования.
5. Широкий охват тестирования: автоматизированные тесты могут проверить большой объем функциональности и данных, что невозможно при тестировании вручную.

Введение в автоматизацию



Минусы автоматизации тестирования

1. Необходимость технического знания: для создания и запуска автоматических тестов необходимы навыки программирования и технического тестирования.
2. Неэффективность в некоторых случаях: некоторые типы тестов, такие как тестирование интерфейса пользователя или тестирование взаимодействия с реальными устройствами, лучше проводить вручную.
3. Высокие затраты на начальную настройку: создание автоматических тестов требует времени и ресурсов, так как необходимо написать скрипты тестов и настроить инструменты автоматизации.
4. Требование постоянной поддержки: при изменении функциональности приложения или обновлении технологий необходимо обновлять и поддерживать автоматические тесты.
5. Ограниченность покрытия: автоматизированные тесты не могут проверить все аспекты приложения, такие как чувствительность к производительности или соответствие требованиям безопасности.

Введение в автоматизацию



Преимущества

Повторяемость

Скорость

Генерируемые отчеты

Автономность

Поддержка

Недостатки

Повторяемость

Разработка

Машинная точность

Анализ падений

Поддержка

INTRODUCTION TO JAVA

ЗНАКОМСТВО С ЯЗЫКОМ JAVA



Знакомство с языком Java



Какие языки применяются в автоматизации?

- JavaScript - это «безопасный» язык программирования, возможности которого ограничены ради защиты данных пользователя. В браузере для JavaScript доступны любые манипулирования веб-страницами и взаимодействия с пользователем, которые не предоставляют доступ к памяти или процессору.
- Python - приобретает огромную популярность среди дата-сайентистов. Его применяют в областях искусственного интеллекта и финансовых технологиях.
- Java - Широко применяется и изучается в университетах. Освоить Java однозначно стоит, так как популярность этот язык в ближайшие годы не потеряет.
- PHP - скриптовый язык общего назначения. Применяется для разработки веб-приложений и создания динамических веб-сайтов.

Знакомство с языком Java



Какие языки применяются в автоматизации?

- C# - объектно-ориентированный язык программирования. Предоставляет средства для разработки любого типа компонентов для платформы Windows и используется для написания сетевых и web-приложений.
- C++ - компилируемый язык программирования. Используется для написания различных редакторов и скоростных веб-серверов.
- Ruby - многоуровневый и постоянно развивающийся язык программирования. Применяется в веб-разработке. Отличается высоким уровнем защиты данных. Код, написанный на Ruby, может быть понятен новичку, который не разбирается в программировании.
- CSS - представляет собой язык описания и набора форматирования внешнего вида документа, написанного с использованием языка разметки. Используется как средство описания и оформления внешнего вида веб-страниц.

Знакомство с языком Java



Какие языки применяются в автоматизации?

- Go - язык разработан компанией Google. Отличается простым и понятным синтаксисом, что делает его привлекательным для новичков. Go имеет широкое сообщество и постоянно развивается.
- Shell - язык программирования очень высокого уровня. Способен хранить и использовать историю сеанса, редактировать командную строку.
- PowerShell - средство автоматизации от Microsoft. Применяется для ИТ-безопасности, защите доступа к данным.
- Perl - язык общего назначения, первоначально созданный для манипуляций с текстом. Сейчас используется в системном администрировании, веб-разработке, сетевом программировании, играх и др.
- Kotlin - язык будущего для Android-разработчиков. Его популярность начала стремительно расти после того, как Google признала Kotlin лучшим языком для своей платформы. А также запустила обучающие курсы.

Знакомство с языком Java



Какой язык выбрать для автоматизированного тестирования?

Самые популярные языки у автоматизаторов – это Java, Python и C#.

Язык программирования Java занимает большую часть рынка и предоставляет больше 30 тысяч рабочих мест для разработчиков. Поскольку Java является широко используемым языком в IT-индустрии, существует огромное сообщество, поддерживающее его. Крупные компании, такие как Amazon, Ebay, PayPal и другие высоко ценят Java. Еще немного фактов в пользу Java:

- Почти 77% тестировщиков Selenium используют Java.
- Java использует JVM, он независим от платформы. Другими словами, вы можете использовать его в любой операционной среде, где установлена JVM.
- Поскольку Java статически типизирован, Java IDE предоставляют много отзывов об ошибках, с которыми вы можете столкнуться.

Знакомство с языком Java



Какой язык выбрать для автоматизированного тестирования?

Идеальное сочетание Java Unit с Selenium WebDriver может расширить возможности автоматического тестирования. Существует множество важных для тестирования фреймворков, разработанных с использованием Java, поэтому это один из наиболее подходящих тестовых скриптовых языков.

Считается, что язык тестирования следует выбирать исходя из того языка, на котором функционирует проект, где вы хотите работать.

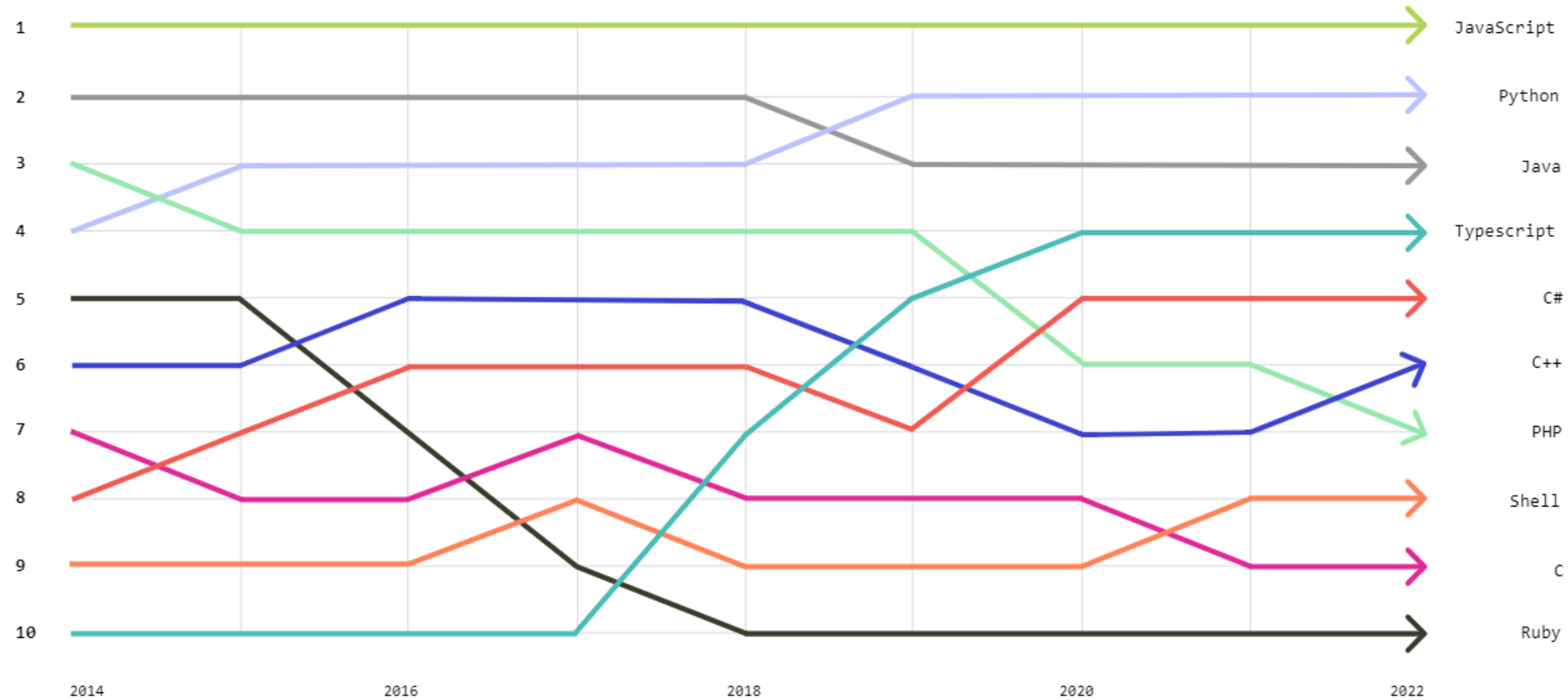
Наиболее популярный язык среди открытых вакансий тестировщиков – Java (64%), на втором месте Python (26%), затем JavaScript (7%), а после уже C# (3%).

В дальнейшем, с развитием в автоматизации, можно перейти с Java на другой язык программирования. Имея базу Java сделать это будет гораздо проще.

ava



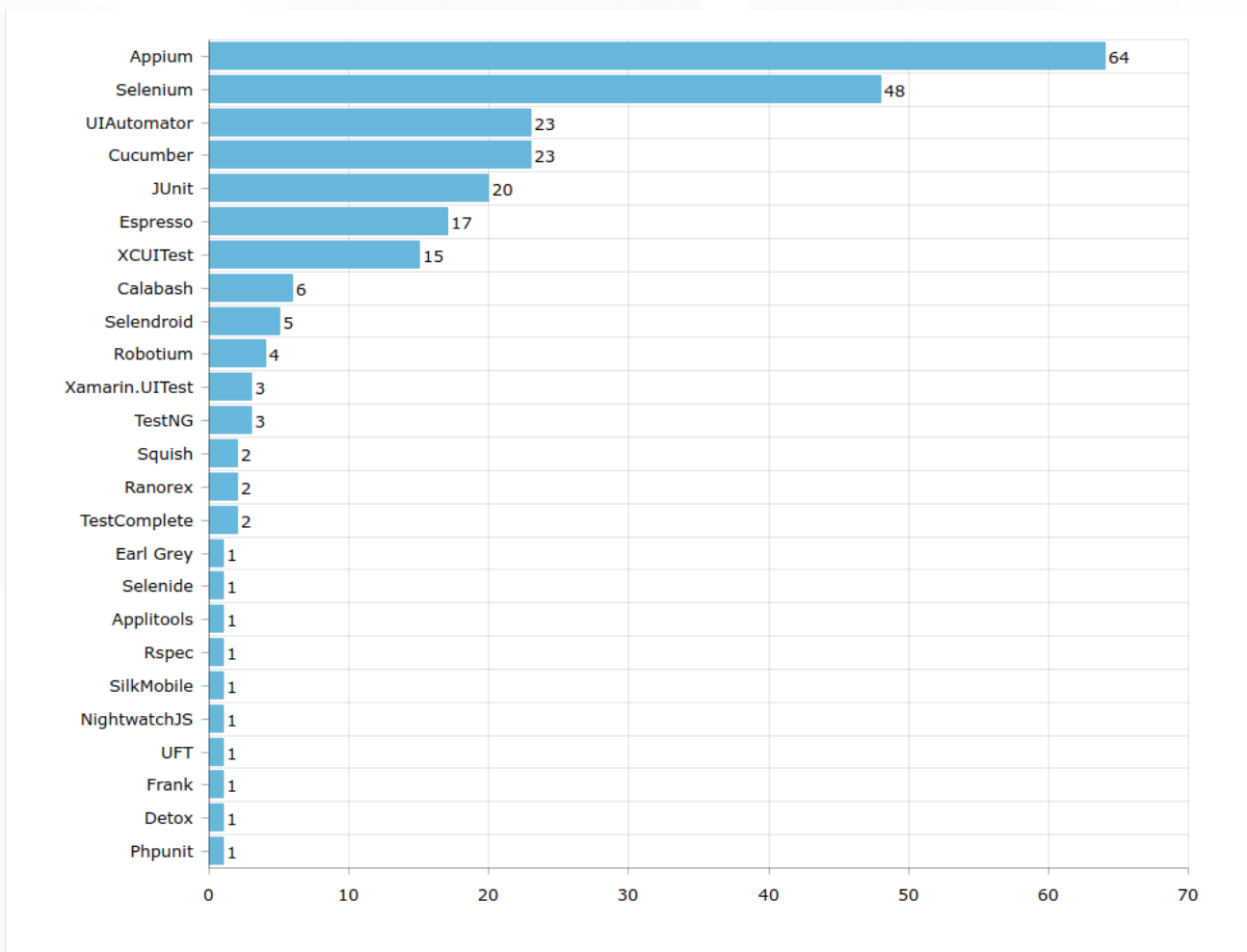
Top languages used in 2014-2022



Знакомство с языком Java



Самые популярные инструменты тестирования на 2022 год



Знакомство с языком Java



История

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретенной компанией Oracle). Дата официального выпуска - 23 мая 1995 года.

Изначально язык назывался Oak («Дуб») разрабатывался для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и серверного программного обеспечения.

Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с горячим кофе.

Знакомство с языком Java



Применения в отраслях программирования

Большинство программ на Android пишутся на Java. Поэтому, благодаря развитию мобильного сектора, Java занимает доминирующее положение в следующих отраслях программирования:

- Enterprise: тяжелые серверные приложения для банков, корпораций, инвестфондов и т.д.
- Mobile: мобильная разработка (телефоны, планшеты), благодаря Android.
- Web: лидирует PHP, но и Java держит солидный кусок рынка.
- Big Data: распределенные вычисления в кластерах из тысяч серверов.
- Smart Devices: программы для умного дома, электроники, холодильников с выходом в интернет.

Java – это не просто язык, это целая экосистема: миллионы готовых модулей, которые можно использовать в программе. Тысячи сообществ и форумов в интернете, где можно попросить помощи или совета.

GIT
GIT



Что такое Git и зачем он нужен?

Git - это консольная утилита и распределённая система контроля версий, для отслеживания и ведения истории изменения файлов, в вашем проекте и работать над одним проектом совместно с коллегами.

Чаще всего его используют для кода, но можно и для других файлов. Например, для картинок - полезно для дизайнеров. С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

Репозиторием называют хранилище вашего кода и историю его изменений. Git работает локально и все ваши репозитории хранятся в определенных папках на жестком диске.

Так же ваши репозитории можно хранить и в интернете. Обычно для этого используют три сервиса: **GitHub, BitBucket, GitLab.**

История

Была разработана в 2005 году Линусом Торвальдсом, создателем Linux, чтобы другие разработчики могли вносить свой вклад в ядро Linux.

Git известен своей скоростью, простым дизайном, поддержкой нелинейной разработки, полной децентрализацией и возможностью эффективно работать с большими проектами.

Подход **Git** к хранению данных похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Преимущества Git

- **Бесплатный и Open-source.** Можно бесплатно скачать и вносить любые изменения в исходный код
- **Небольшой и быстрый.** Выполняет все операции локально, что увеличивает его скорость. Кроме того, Git локально сохраняет весь репозиторий в небольшой файл без потери качества данных
- **Резервное копирование.** Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при использовании Git
- **Простое ветвление.** В других системах контроля версий (SVN) создание веток - утомительная и трудоёмкая задача, так как весь код копируется в новую ветку. В Git управление ветками реализовано гораздо проще и эффективнее.

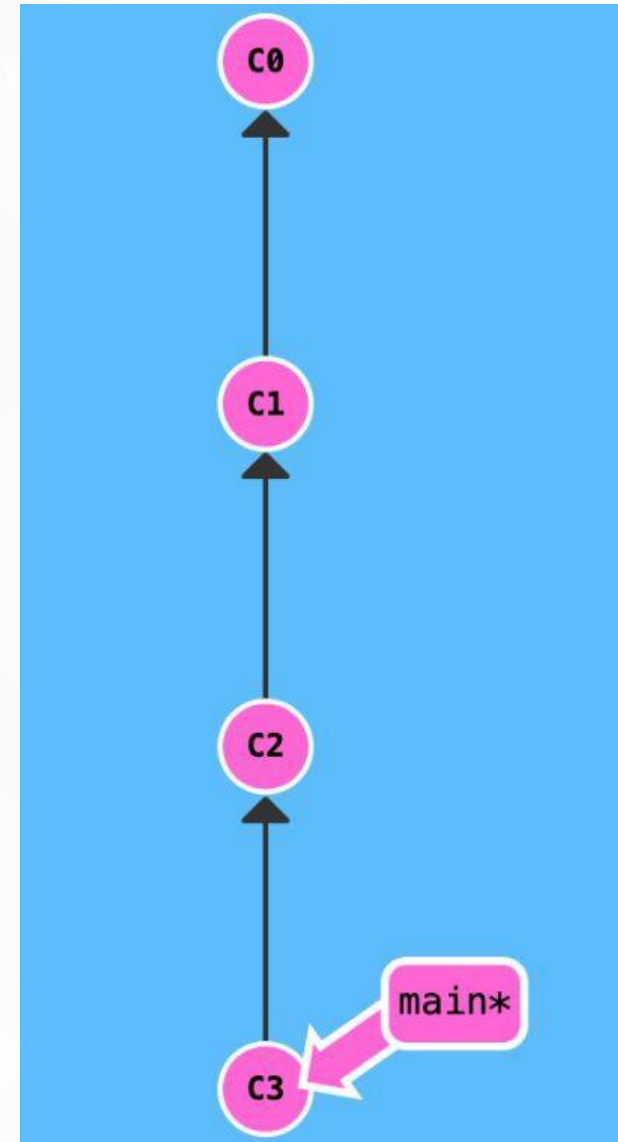
Git



Как работает Git?

Каждая точка сохранения вашего проекта носит название коммит (**commit**).

У каждого **commit** есть **hash** (уникальный id) и комментарий. Из таких **commit** собирается ветка.



Git

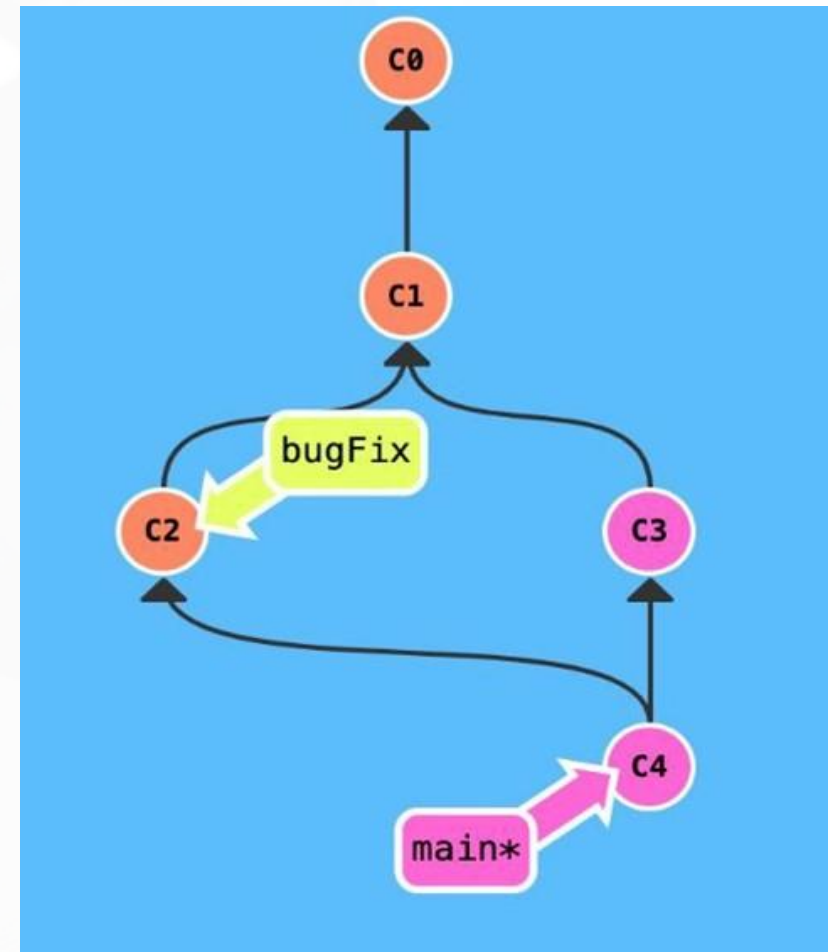


Как работает Git?

Ветка - это история изменений.

У каждой ветки есть свое название.

Репозиторий может содержать в себе несколько веток, которые создаются из других веток или вливаются в них.



Основные команды

Всего несколько команд нужно для базового варианта использования Git для ведения истории изменений.

- **git add:** Команда `git add` добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита.
- **git commit:** Команда `git commit` берёт все данные, добавленные в индекс с помощью `git add`, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.
- **git push:** Команда `git push` используется для выгрузки содержимого локального репозитория в удаленный репозиторий. Она позволяет передать коммиты из локального репозитория в удаленный.
- **git pull:** Команда `git pull` используется для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым.

Основные команды

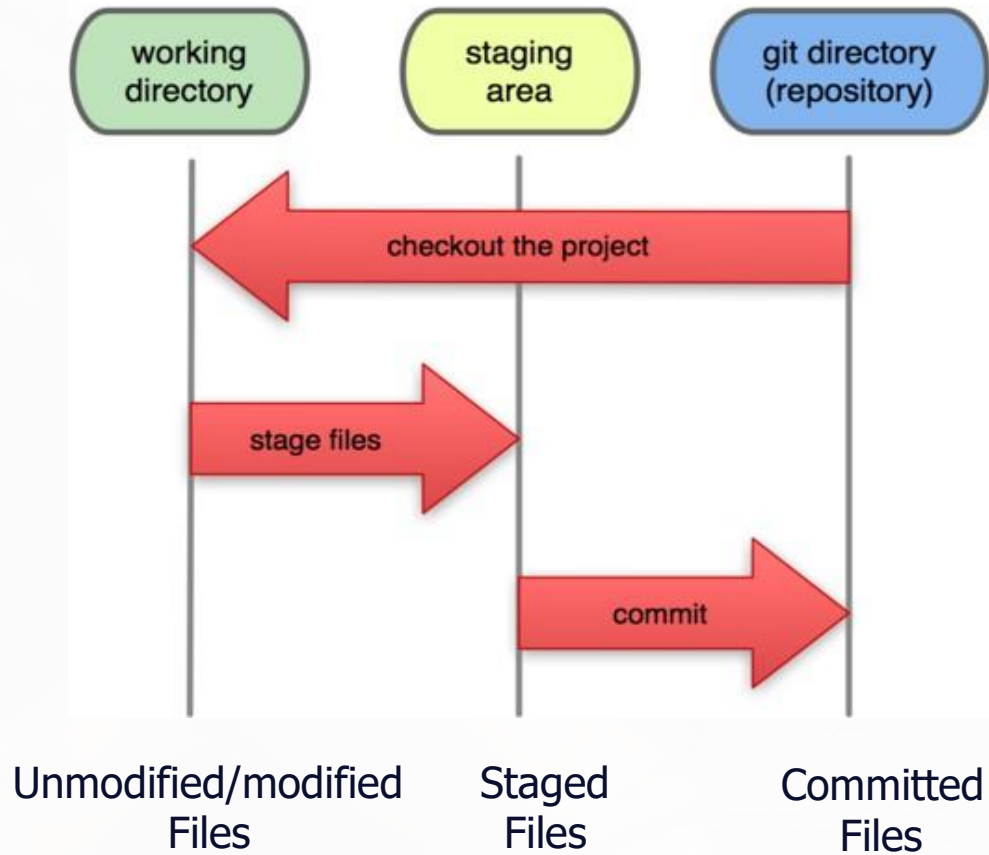
command	description
<code>git clone <i>url</i> [<i>dir</i>]</code>	copy a Git repository so you can add to it
<code>git add <i>file</i></code>	adds file contents to the staging area
<code>git commit</code>	records a snapshot of the staging area
<code>git status</code>	view the status of your files in the working directory and staging area
<code>git diff</code>	shows diff of what is staged and what is modified but unstaged
<code>git help [<i>command</i>]</code>	get help info about a particular command
<code>git pull</code>	fetch from a remote repo and try to merge into the current branch
<code>git push</code>	push your new branches and data to a remote repository
others: <code>init</code> , <code>reset</code> , <code>branch</code> , <code>checkout</code> , <code>merge</code> , <code>log</code> , <code>tag</code>	

<https://docs.github.com/en/get-started/quickstart/hello-world>

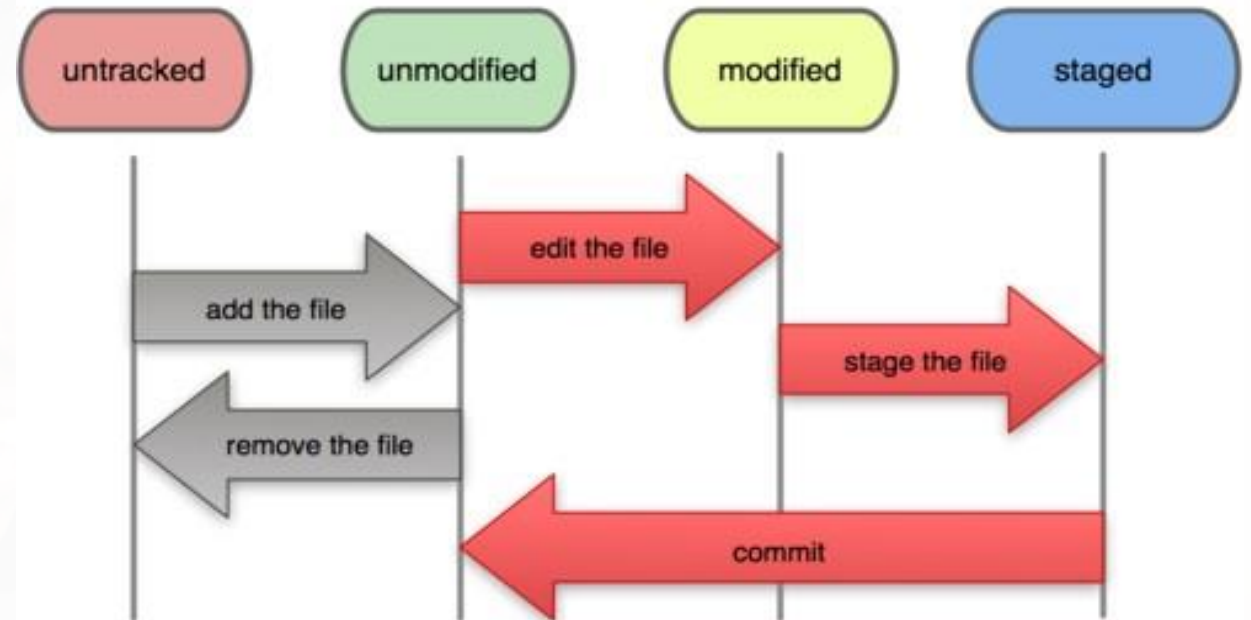
Git



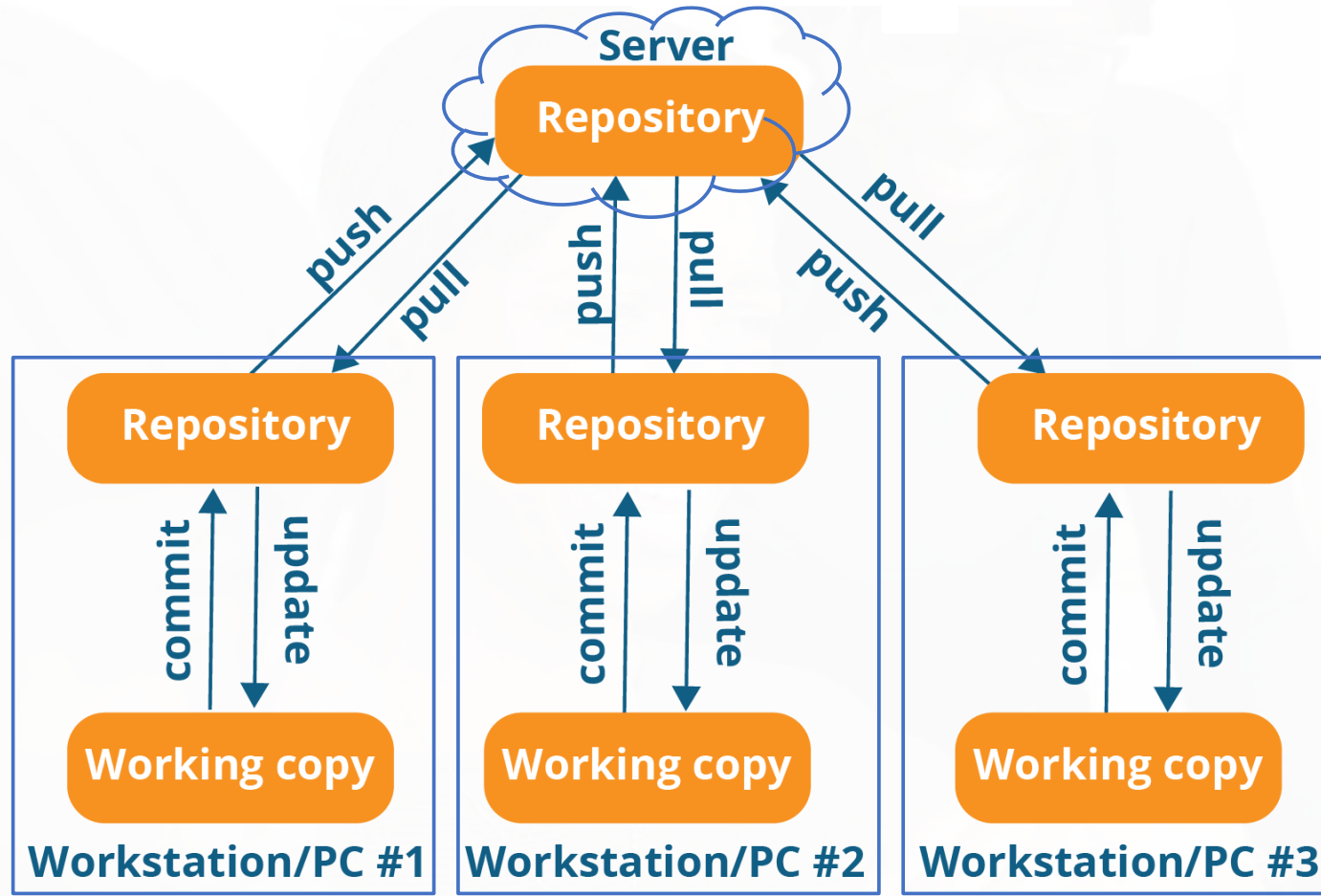
Local Operations



File Status Lifecycle



Git



Установка Git

1. Скачиваем Git для ОС на которой мы работаем: <https://git-scm.com/>
2. Если всё прошло успешно – команда `git --version` покажет актуальную версию: `git version 2.41.0.windows.1`
3. Теперь настроим, чтобы при создании commit, указывался автор, кто его создал:
 1. Открываем терминал (Linux и MacOS) или консоль (Windows) и вводим следующие команды:
 - `git config --global user.name "ваше_имя"`
 - `git config --global user.email "адрес_почты@email.com"`
 2. Проверим, если установка прошла удачно: `git config --list`



Thanks for your time 😊