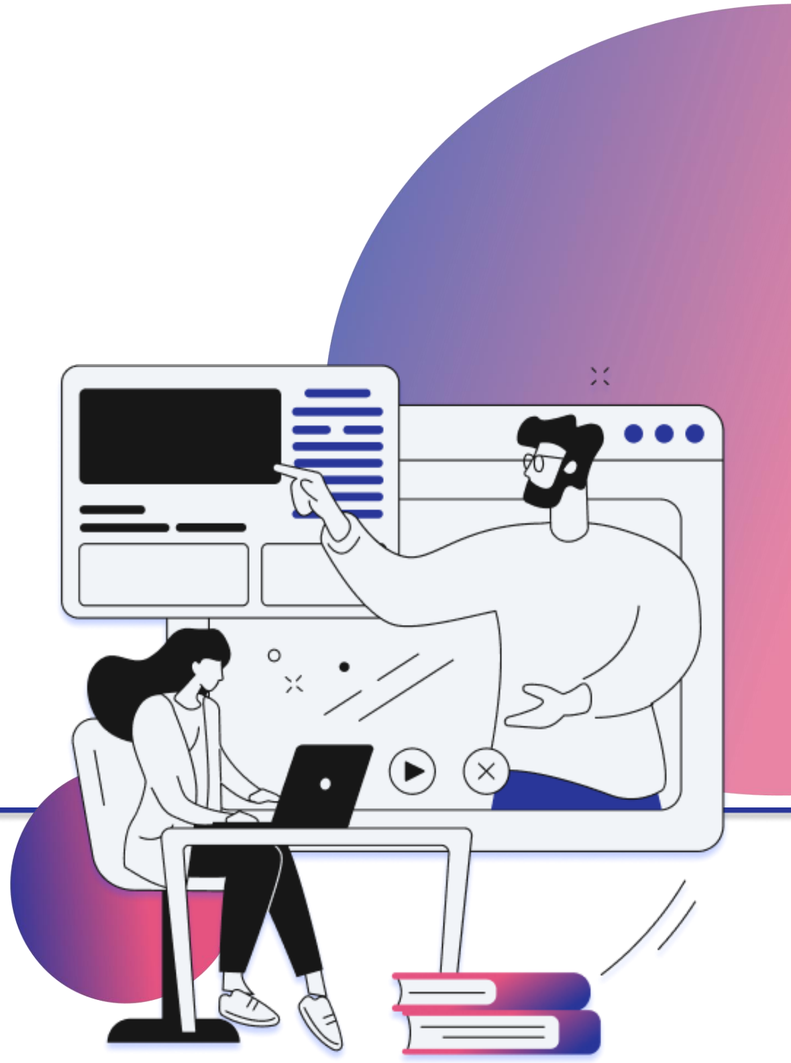
CLOUD
УЧЕБНЫЙ ЦЕНТР

Учебный центр АйТи Клауд

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОСТЫМ ЯЗЫКОМ

Преподаватель _____

Учебный центр АйТи Клауд



Реализация CI\CD на базе Jenkins

современные технологии простым языком

Continuous Integration & Continuous Delivery

Я только запушил, а у клиента уже всё упало

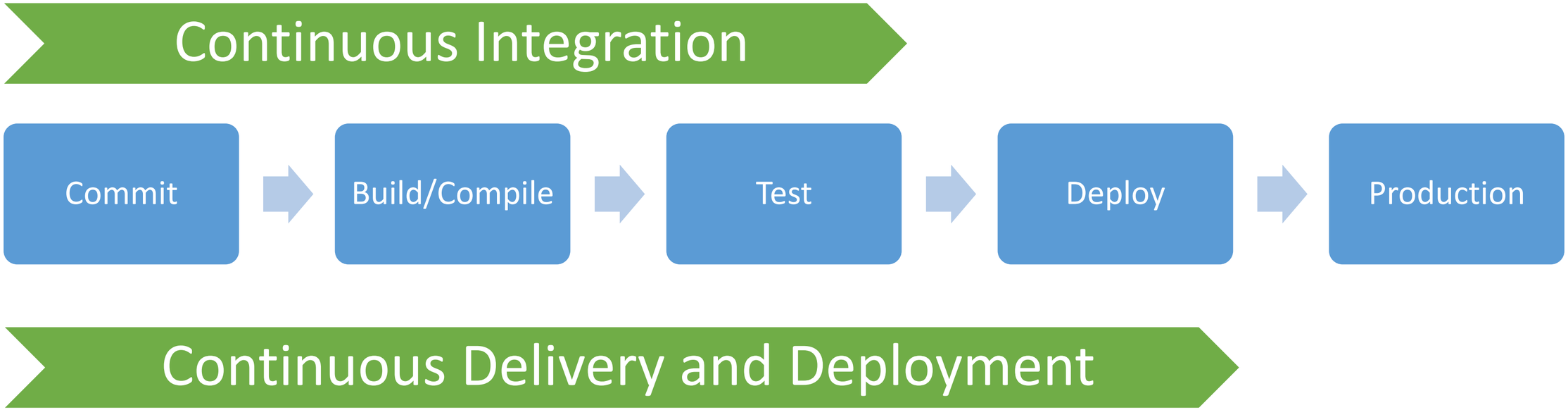
CI – Continuous Integration

Это DevOps-модель, в которой разработчики делают Commit кода в репозиторий и автоматически запускается Build или компиляция это кода, после этого запускаются автоматические тесты: Smoke Test, Unit test, integration test, functionality test

CD – Continuous Delivery|Deployment

Это DevOps-модель, в которой разработчики делают Commit кода в репозиторий и автоматически запускается Build или компиляция это кода, после этого запускаются автоматические тесты и готовый Artifact (скомпилированный код) развёртывается в staging, production

CI/CD


Continuous Integration
Commit
Build/Compile
Test
Deploy
Production
Continuous Delivery and Deployment

Знакомство с Jenkins

Jenkins — это система с открытым исходным кодом для непрерывной интеграции и непрерывной доставки (CI/CD). Она автоматизирует различные этапы процесса разработки программного обеспечения: сборку, тестирование, развёртывание и мониторинг приложений. Jenkins позволяет разработчикам интегрировать код в общий проект автоматически, обеспечивая непрерывную проверку и интеграцию изменений.

Знакомство с Jenkins

Основные особенности Jenkins:

1. Автоматизация сборок: Jenkins может запускать сборки кода автоматически при каждом изменении в репозитории, таких как Git, Mercurial и других.
2. Расширяемость: Jenkins поддерживает большое количество плагинов, которые позволяют интегрироваться с различными системами и инструментами, такими как Docker, Kubernetes, Maven, Gradle и другие.
3. Гибкость: Jenkins может быть настроен под различные процессы разработки, как с использованием простых скриптов, так и сложных пайплайнов на основе Jenkins Pipeline (скрипты на языке Groovy).
4. Пайплайны CI/CD: Jenkins позволяет создавать конвейеры для автоматизации процессов от написания кода до его развёртывания на продакшн.
5. Интерфейс: Имеет веб-интерфейс, через который можно отслеживать и управлять процессами.

Build

В Jenkins термин build (сборка) обозначает процесс компиляции, тестирования и подготовки проекта к выпуску, выполненный автоматически в соответствии с заданной конфигурацией. Этот процесс включает последовательность шагов, направленных на сборку (build) и тестирование (test) программного обеспечения, и может содержать разное количество этапов в зависимости от сложности проекта и его требований.

Build

Когда в Jenkins запускается сборка, происходят следующие действия:

1. Загрузка кода: Jenkins сначала получает последнюю версию кода из системы контроля версий (например, Git), если проект настроен для работы с репозиторием.
2. Выполнение этапов сборки: Эти этапы обычно описаны в конфигурации job или в Jenkinsfile, где можно определить команды для компиляции, тестирования, сборки артефактов, проверки кода на стиль и выполнение других задач, необходимых для подготовки проекта.
3. Тестирование: Запускаются юнит-тесты, интеграционные тесты или любые другие проверки, которые помогают убедиться, что сборка проходит успешно и что код работает, как ожидается.

Build

4. Создание артефактов: Если сборка успешна, Jenkins может собрать артефакты — готовые файлы, такие как .jar или .war для Java, контейнеры Docker, бинарные файлы, архивы и т.п.
5. Уведомление и публикация: После выполнения сборки Jenkins может отправить уведомления об успешной или неудачной сборке, сохранить артефакты и результаты тестов, а также запустить последующие процессы, например, развертывание на сервере.

Jenkins автоматически выполняет эти этапы по заданным правилам, проверяя, что каждый этап завершается успешно, прежде чем перейти к следующему.

Jenkins Nodes\Slaves

Использование дополнительных нод Jenkins позволит вам:

- Снизить нагрузку на основном сервере Jenkins
- Выполнять больше builds одновременно (повысить количество экзекуторов)
- Совершать build для разных платформ на разных серверах, например один сервер для .NET, а другой для Java или Ansible

Node может быть и на Linux и на Windows

Jenkins Nodes\Slaves

Master

Node1 .Net

Node2 JAVA

Node3 Ansible

Jenkins cli

Jenkins CLI (Command Line Interface) — это мощный инструмент, который предоставляет возможность управлять Jenkins сервером и его заданиями (jobs) через командную строку, без использования веб-интерфейса. Это полезно для автоматизации задач администрирования Jenkins, создания, запуска и управления заданиями и многого другого.

Возможности Jenkins CLI

Создание и управление заданиями:

- Создание нового задания из конфигурационного XML-файла.
- Экспорт существующего задания в XML.
- Удаление задания.

Запуск и контроль выполнения задач:

- Запуск заданий с возможностью передавать параметры.
- Просмотр статуса выполнения заданий.
- Просмотр логов выполнения задания.

Возможности Jenkins CLI

Управление Jenkins:

- Остановка или перезапуск Jenkins.
- Просмотр информации о плагинах и их управление.
- Управление пользователями и авторизацией.

Управление конфигурацией:

- Экспорт и импорт конфигурации Jenkins.
- Управление очередью задач и блокировкой выполнения.

Установка и подключение Jenkins CLI

Получение CLI клиента:

CLI клиент Jenkins доступен через веб-интерфейс. Его можно скачать, добавив к URL Jenkins серверу /jnlpJars/jenkins-cli.jar. Например:

```
wget http://your-jenkins-url/jnlpJars/jenkins-cli.jar
```

Установка и подключение Jenkins CLI

Подключение к Jenkins:

CLI работает по протоколу HTTP(S), и для подключения необходимо указать URL Jenkins сервера, а также может потребоваться аутентификация.

Пример команды для подключения и выполнения команды help:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ help

Установка и подключение Jenkins CLI

Аутентификация: В зависимости от настроек безопасности Jenkins, для выполнения некоторых команд может потребоваться указать токен доступа или логин с паролем. Аутентификация может быть выполнена с помощью:

- API Token:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ -auth user:apitoken <command>

- SSH-ключ:

Jenkins также поддерживает аутентификацию через SSH. Для этого на Jenkins сервере должен быть настроен SSH доступ.

Основные команды Jenkins CLI

Запуск задания:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ build my-job

Для запуска задания с параметрами:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ build my-job -p param1=value1 -p param2=value2

Получение списка заданий:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ list-jobs

Экспорт конфигурации задания:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ get-job my-job > my-job-config.xml

Основные команды Jenkins CLI

Создание нового задания:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ create-job new-job < my-job-config.xml

Получение логов задания:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ console my-job

Перезапуск Jenkins:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ restart

Остановка Jenkins:

java -jar jenkins-cli.jar -s http://your-jenkins-url/ safe-shutdown

Основные команды Jenkins CLI

Запускаем задание:

java -jar jenkins-cli.jar -s http://jenkins-url/ build my-job

Следим за статусом:

java -jar jenkins-cli.jar -s http://jenkins-url/ job-status my-job

Jenkins Build Trigger

В Jenkins триггеры сборок (build triggers) управляют автоматическим запуском джоб. Они позволяют настраивать события, которые инициируют запуск сборки. Jenkins поддерживает несколько способов инициировать сборки, включая:

- Полинг SCM (Poll SCM): Jenkins может проверять репозиторий на наличие изменений через указанные интервалы времени. Если обнаруживаются изменения в коде, Jenkins запускает сборку. В настройках джоба можно задать cron-подобное расписание.
- Webhook триггеры (GitHub/GitLab/Webhooks): Используется для немедленного запуска сборки при изменениях в коде, отправленных в репозиторий. GitHub и GitLab поддерживают отправку вебхуков, которые уведомляют Jenkins, когда происходят изменения, чтобы избежать постоянного полинга и уменьшить нагрузку.

Jenkins Build Trigger

- Запуск по расписанию (Build periodically): Этот триггер позволяет запускать сборку по заданному расписанию с использованием cron-подобных форматов. Подходит для задач, которые должны выполняться регулярно, например, ежедневные сборки.
- Запуск после завершения других сборок (Build after other projects are built): Позволяет запускать сборку только после завершения других, указанных задач. Это полезно для выстраивания цепочек задач и создания конвейеров.

Jenkins Build Trigger

- Запуск по событию (Triggered remotely): Можно настроить удалённый запуск сборки через HTTP-запросы, используя API Jenkins. Для этого задаётся URL с токеном, который можно вызвать для запуска задачи извне.
- Плагины для триггеров: Jenkins поддерживает множество плагинов для интеграции с другими инструментами и триггерами. Например, плагин для Bitbucket может запускать сборку при push'e в репозиторий Bitbucket, а плагин для RabbitMQ позволяет запускать сборки при получении определённых сообщений из брокера сообщений.

Выбор подходящего триггера зависит от требований и особенностей сборки.

Jenkins Groovy

В Jenkins Groovy script – это сценарий на языке программирования Groovy, используемый для автоматизации и настройки сборок, а также для управления Jenkins и его ресурсами. Groovy является встроенным языком в Jenkins, что позволяет выполнять сложные задачи конфигурации и управления, которые сложно реализовать стандартными средствами. Сценарии Groovy можно использовать как в Pipeline, так и в консоли управления Jenkins (Script Console).

Script Console

В Jenkins также есть Script Console, доступная из панели администратора, которая позволяет выполнять Groovy-код для управления сервером Jenkins и его конфигурацией. Это полезно для массового обновления конфигураций, управления пользователями, задачами, настройками безопасности, плагинами и других административных операций.

Пример: получение списка всех job на сервере:

```
Jenkins.instance.items.each { job ->    println job.name}
```

Jenkins API на Groovy

Groovy предоставляет доступ к внутреннему API Jenkins, что позволяет взаимодействовать с системными объектами. Например, можно управлять заданиями, устанавливать настройки для агентов и узлов, получать статистику сборок, управлять зависимостями и многое другое.

Пример: перезапуск определённого задания:

```
def job = Jenkins.instance.getItemByFullName("job-name")job.scheduleBuild2(0)
```

Jenkins Pipeline

Jenkins Pipeline — это мощная система автоматизации сборок, тестирования и деплоя кода в Jenkins, которая описывает весь процесс CI/CD в виде кодовых скриптов. Pipelines (конвейеры) делают процессы в Jenkins более гибкими и настраиваемыми, позволяя легко организовать сложные, многоэтапные рабочие процессы. Они записываются в виде скриптов на Groovy, которые можно сохранить в репозитории как файл Jenkinsfile.

Jenkins Pipeline

В Jenkins есть два основных типа Pipeline:

1. Declarative Pipeline (Декларативный): более упрощённый, использующий фиксированную структуру, что делает его удобным для большинства пользователей.
2. Scripted Pipeline (Скриптовый): более гибкий, предоставляет полный доступ к API Jenkins. Чаще всего используется для более сложных сценариев, где требуется кастомная логика.

Jenkins Pipeline

Основные компоненты Jenkins Pipeline

- Agent: Определяет, на каком узле или агенте будет выполняться pipeline. Например, agent any указывает на выполнение на любом доступном агенте, а agent { label 'linux' } запустит конвейер на агенте с меткой linux.
- Stages: Pipeline состоит из стадий (stage). Каждая стадия обозначает логическую часть конвейера, например, стадии Build, Test, Deploy.
- Steps: Внутри каждой стадии определяются шаги (steps), которые выполняют конкретные действия. Это могут быть команды сборки, тестирования или любые другие задачи.
- Post Actions: Этот раздел описывает действия, которые нужно выполнить после завершения pipeline. Например, отправка уведомлений при успешной или неудачной сборке, очистка ресурсов и др.

Функции и возможности Jenkins Pipeline

Параллельные этапы: Можно выполнять этапы параллельно, что сокращает общее время выполнения сборки.

```
stage('Test') {
    parallel {
        stage('Unit Tests') {
            steps {
                echo 'Running unit tests...'            }        }
        stage('Integration Tests') {
            steps {                echo 'Running integration tests...'      }        }    }}
```

Функции и возможности Jenkins Pipeline

Обработка ошибок: Pipeline позволяет ловить и обрабатывать ошибки, используя блоки try-catch, а также выполнять очистку через блок post.

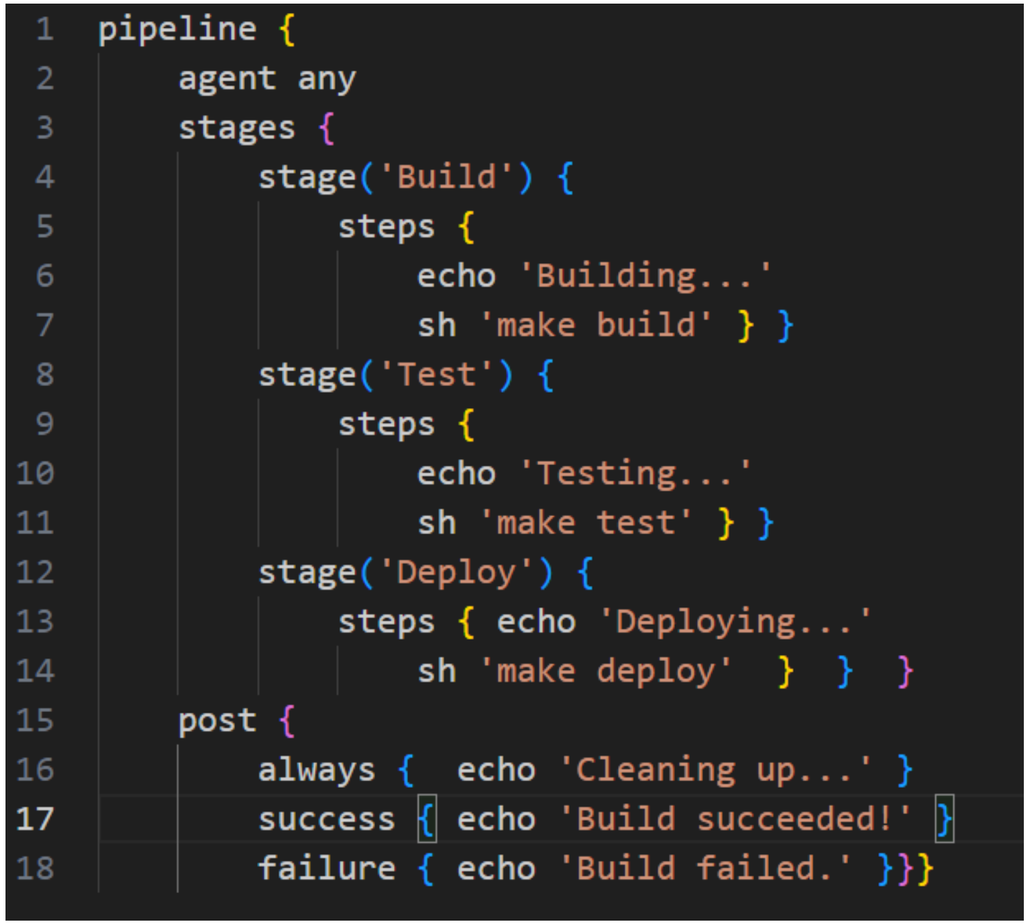
```
post {
    always {
        echo 'Cleaning up...'    }
    success {
        echo 'Build succeeded!'    }
    failure {        echo 'Build failed.'    }}
```

Jenkinsfile

Jenkinsfile — это текстовый файл с описанием процесса CI/CD в Jenkins, написанный на Groovy. Jenkinsfile используется для создания Pipeline, который автоматизирует сборку, тестирование и деплой кода. Хранясь вместе с исходным кодом, Jenkinsfile позволяет управлять конфигурацией Pipeline, версионировать её и легко изменять.

Jenkinsfile

- pipeline — основной блок, содержащий всё описание Pipeline.
- agent — указывает, на каком агенте Jenkins будет выполняться Pipeline. agent any означает, что Pipeline может выполняться на любом доступном агенте.
- stages — блок, содержащий последовательные этапы (например, Build, Test, Deploy).
- steps — конкретные шаги, выполняемые на каждом этапе, такие как команды сборки и тестирования.
- post — блок, содержащий инструкции, которые выполняются после завершения Pipeline. Например, always, success и failure позволяют выполнять действия в зависимости от результата Pipeline.



```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building...'
                sh 'make build' } }
        stage('Test') {
            steps {
                echo 'Testing...'
                sh 'make test' } }
        stage('Deploy') {
            steps { echo 'Deploying...'
                sh 'make deploy'  }  }  }
    post {
        always {  echo 'Cleaning up...' }
        success { echo 'Build succeeded!' }
        failure { echo 'Build failed.' }}}
```

Пример Scripted Pipeline

- node — блок, обозначающий узел Jenkins, где будут выполняться команды.
- stage — этап выполнения, содержащий конкретные команды для определённой стадии (Build, Test, Deploy).
- echo и sh — шаги, выполняющие команды, где echo выводит сообщение, а sh запускает команду в shell.

```
node {
    stage('Build') {
        echo 'Building...'
        sh 'make build'
    }
    stage('Test') {
        echo 'Testing...'
        sh 'make test'
    }
    stage('Deploy') {
        echo 'Deploying...'
        sh 'make deploy'
    }}
```

Ключевые компоненты Jenkinsfile

- Agent — указывает, на каком агенте или узле будет выполняться Pipeline.
- Environment — задаёт переменные окружения, доступные на всех этапах.
- Parameters — определяет параметры, которые можно задать перед запуском Pipeline, например, ветку или тип сборки.
- Stages — основная структура Pipeline, где каждый stage представляет этап работы.
- Steps — шаги, выполняемые на каждом этапе, такие как команды или вызов скриптов.
- Post — позволяет определить действия, которые выполняются после завершения Pipeline, в зависимости от результата.

```
environment {
    JAVA_HOME = '/usr/lib/jvm/java-11-openjdk'
    PATH = "${JAVA_HOME}/bin:${env.PATH}"}
```

```
parameters {
    string(name: 'BRANCH', defaultValue: 'main', description: 'Git branch to build')
}
```

Спасибо, что выбрали обучение в УЦ АйТи Клауд!

Желаем успешно применить полученные знания на практике!

И будем ждать новых встреч!

Связаться с преподавателем _____

Наш сайт