нетология

Практическое знакомство с Jenkins

Алексей
Метляков

Алексей Метляков

DevOps Engineer

OpenWay

Алексей Метляков

План занятия

1. Jenkins
2. Сущности Jenkins
3. Дополнения
4. Итоги
5. Домашнее задание

Jenkins

Что такое Jenkins?

Jenkins - ППО для управления **CI\CD** процессами

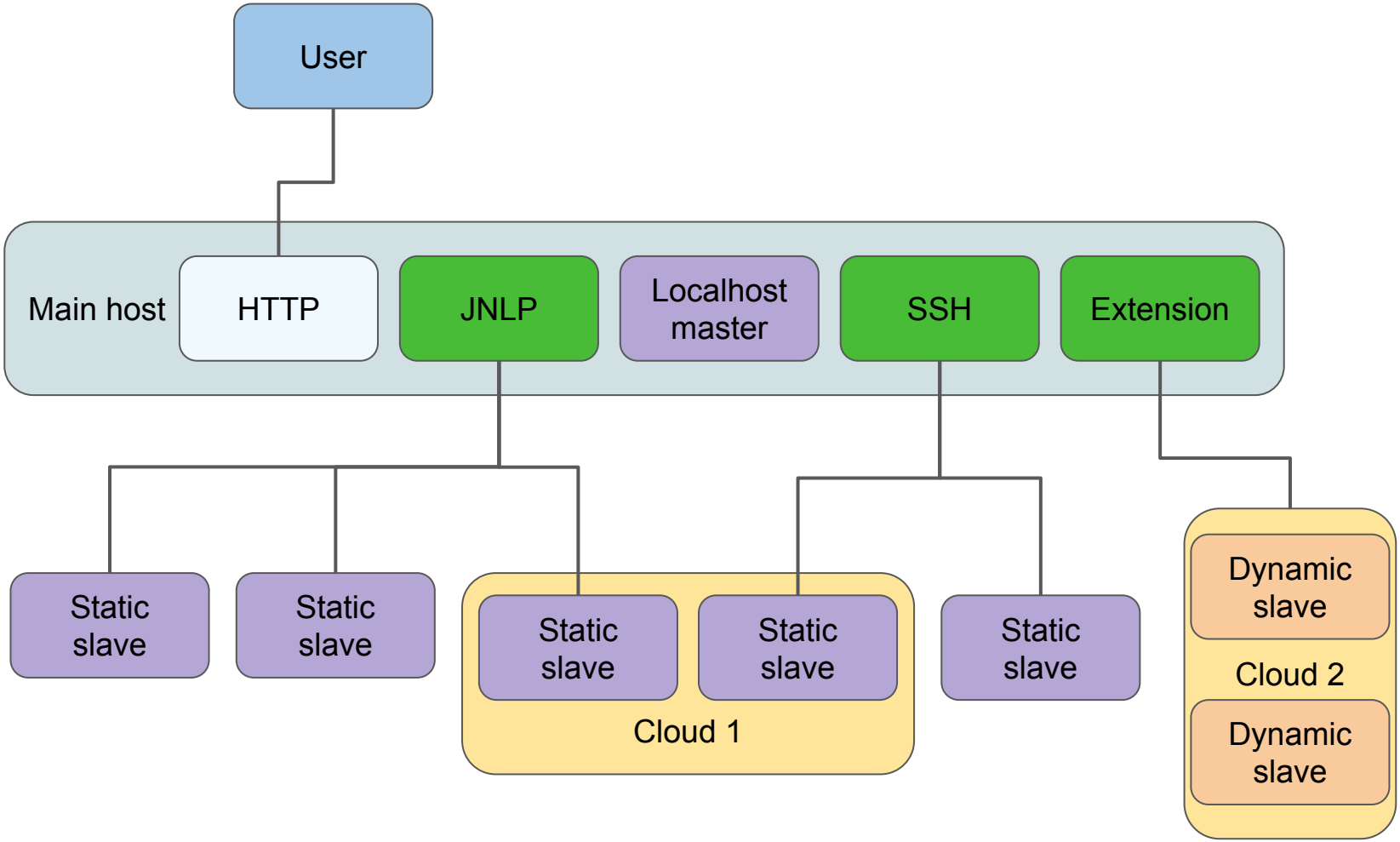
- **OpenSource** - следовательно, бесплатен
- **Многофункционален** - имеет большое количество видов использования автоматизаций
- Имеет возможность **расширения**
- Использует встроенные синтаксисы: **groovy** и **pipeline**

Что такое Jenkins?

Некоторые основные возможности системы:

- **Создание** конвейера **автоматизации**
- Создание и **управление** пользователями
- Синхронизация с **AD**
- Ограничение прав пользователей на разных вложенностях сущностей
- Установка **плагинов**
- Настройка **Custom Tool**
- Использование динамических и статических окружений

Архитектура Jenkins



Сущности Jenkins

Freestyle Job

Freestyle Job - простейшее описание рутины в виде последовательно выполняемых заданий.

- Задача может быть описана удобным способом
- Часть заданий можно описать в репозитории

Groovy

Groovy - объектно-ориентированный язык программирования разработанный, как дополнение к **java**.

- Apache 2 **License**
- Может быть языком **статической** типизации
- Может быть языком **динамической** типизации
- Имеет **встроенный** синтаксис для работы с массивами, списками и регулярками
- Считается смесью **java** и **python**, **ruby** и **smalltalk**

Больше о самом языке на их официальном сайте

Groovy

В контексте **Jenkins'a**, **groovy** может нас заинтересовать следующими объектами:

- Package **jenkins**
  - jenkins.*
  - jenkins.model.*
- Package **hudson**
  - hudson.*
  - hudson.model.*

Для полноценного использования можно ознакомится с содержимым официальной страницы

```
Jenkins.instance.computers.each{
  println "${it.displayName} ${it.hostName}"
}
```

Pipeline Job

Pipeline Job - описание рутины в отдельных файлах с **pipeline** на собственном синтаксисе. Удобство в большей визуализации различных этапов сборки и отдельной обработки каждой стадии (**stage**) при работе с шагами (**steps**). Хранится в **репозитории** или в самом **Jenkins.**

Может быть описано в двух видах:

- **Declarative** Pipeline
- **Scripted** Pipeline

Pipeline Syntax

Declarative Pipeline - верхнеуровневое описание того, что необходимо сделать. Имеет свой собственный синтаксис и описывается в файлах с именованием **Jenkinsfile**.

Пример **Declarative** Pipeline:

```
pipeline {
      agent any
      stages {
            stage('Build'){
                  steps{
                  }
            }
            stage('Test'){
                  steps{
                  }
            }
      }
}
```

Pipeline Syntax

Scripted Pipeline - описание стадий конвейера с использованием собственного синтаксиса, но с дополнениями в виде **groovy-скриптов**.

Пример **Scripted** Pipeline:

```
node {
      stage('Build'){
      }
      stage('Test'){
      }
      if (currentBuild.currentResult == SUCCESS){
            stage('Test'){
            }
      }
}
```

Pipeline Syntax

- **Jenkins** был написан в тесной связи с **groovy**
- **Groovy** нужно было изолировать от использования системных вызовов - был создан **Scripted** Pipeline
- Чтобы не принуждать всех пользователей изучать **groovy** - был создан **Declarative** Pipeline с упрощенным интерфейсом

Полное описание работы с pipeline можно найти в документации

Multibranch Pipeline

Multibranch Pipeline - вид Pipeline, который умеет запускать сборки по действиям в рамках одного репозитория.

- Умеет разделять виды действий в репозитории
- Фильтровать имена branches

Дополнения

Blue Ocean

Blue Ocean - новый UI для Jenkins. Призван улучшить читаемость происходящего и сделать более качественную визуализацию прохождения этапов сборки.

Изначально, вырос из концепции **Blue Ocean Strategy**: Нужно рассматривать проблему более широко, чем просто смотреть туда, где она возникла.

Ansible

Ansible - добавляет возможность запускать ansible через синтаксис Declarative Pipeline. Фактически, избавляет от необходимости вызова оболочки для вызова плейбука.

Пример синтаксиса можно посмотреть на официальной странице плагина:

```
ansiColor('xterm') {
    ansiblePlaybook(
        playbook: 'path/to/playbook.yml',
        inventory: 'path/to/inventory.ini',
        credentialsId: 'sample-ssh-key',
        colorized: true)
}
```

Docker

Docker - позволяет осуществлять сборки на docker-контейнерах. Также, позволяет собирать собственные образы и выкладывать их в Registry. Особенности настройки и работы можно посмотреть на странице.

Monitoring

Monitoring - плагин, позволяющий отслеживать java-процессы через **JavaMelody**. Можно отслеживать как процессы **master** так и **agents**.

Позволяет:

- Проверять **работоспособность** потоков
- Смотреть созданные объекты в **heap**
- Вызывать **GC**
- Собирать **heapdump**

Итоги

Итоги

- **Master** - основной процесс **Jenkins**, он же исполнитель на ноде
- **Agent** - **java** процесс на отдельном хосте для выполнения **job**
- **Job** - описание рутинного рабочего процесса
- **Job** может быть нескольких видов, основные
  - **Freestyle**
  - **Pipeline**
- Существует огромный набор расширений функционала
- Инструмент обладает гибкостью в настройке
- Полностью бесплатен

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Алексей Метляков

Алексей Метляков