Artifact Repositories

Ádám Révész, Attila Ulbert, Norbert Pataki, Zoltán Gera

Eötvös Loránd University,
Budapest, Hungary

Introduction to
Cloud Technologies

Contents

Introduction

All of our service builds end with an artifact...

Motivation

- We cannot always start our pipelines with building every microservice and configuration.
- Sometimes we cannot (or don't want to) pass the source code to the client.
- We cannot rely on third party package repositories when it's comes down to production. Don't want to get in trouble with some deleted GitHub account caused package outage (see NPM, 2015).

Definition

Artifact: tangible produced during software development. An artifact is a piece of any kind:

- library
- piece of software
- design files
- binaries
- software packages
- configurations
- ...

Public repositories

- You may already used at least one of them:
  - Maven – Java world
  - npmjs.org – NodeJS Package Manager
  - Docker Hub – Docker
  - Pypi – Python package index
  - Hackage – Haskell Cabal repository
  - Hex – Erlang repository
  - ...

Public repositories

Benefits:

- Free to use
- Easy integration with social developer repositories like GitHub and public CI tools, like Travis CI
- Wide wariaty of packages

Downsides:

- No warranty for uptime
- No warranty for persisting packages – What will we do when one of our dependencies disappears?
- Not optimal for private use.

Private repositories

- It can be outsourced, but most of the time it's self-hosted due to legal reasons.
- Most of artifactory softwares support multiple package types and package managers (eg Cabal, npm, Docker Image, etc.).
- Caches requested public packages, so when npmjs.org is down, or your favourite Docker Image is deleted from Docker Hub, you can still access them in your private repo.