Integrating Artifactory into a Simple Build

Chris B. Behrens
Software Architect

@chrisbbehrens

Artifactory in Automated Builds

The simplest way – a store for your dependencies

Protects against dependency confusion and primary feed downtime

We'll use the nuget package manager

Jenkins, my very favorite build server

In its own container

Demo

An existing build for our Globomantics software project

Look at the instructions at the JFrog website to integrate with a build

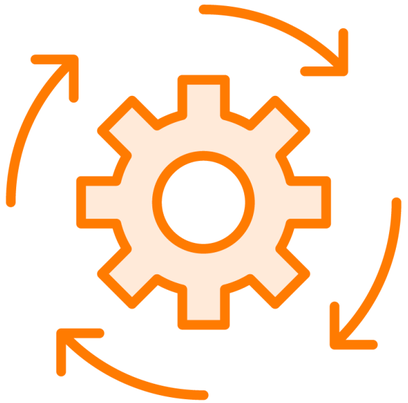
Apply those instructions with our build

In a secure way

The Problems with What We've Got

Secrets don’t belong in version control

Secrets Don’t Belong in a Number of Places



Secrets don't belong in build artifacts, either

Especially not in the logs

Logs get handed around when there's a problem

A big question about configuration:

Centralized or distributed?

“a single, unified place”, the registry

Ugh

The Problem with Central Config

Different applications have different opinions

This leads to DLL Hell

Conflicts in which version of a DLL should be present and used

.NET Binding Redirects

How This Problem Bears on Our Work

NuGet is centrally configured (like most package managers)

Raw commands are executed against the centralized config

And Add Source will fail if the source exists

- Check whether the source already exists in the file, or
- Use a local config file pulled from version control

I *hate* centralized config

Action items

- Inject the API key into the build
- Target a local config instance

The Problem

The instructions don't work on Linux

.NET is cross-platform...mostly

No encryption on Linux

We need a special command

--store-password-in-clear-text

The Package That Broke the Internet

An Internet of Things project

Using D3.js

The software was not building

I drilled into the logs

The feed was up, the package was down

Left-pad

Our dependency chain was deep and wide

Checking out Left-pad

No published packages

A missing link in the dependency chain

This affected a *lot* of people

Unless you had a copy cached...

Your build was dead

The Three Jerks

A new instant messenger called Kik

This conflicts with an npm package of the same name

The company sends a cease and desist

The Second Jerk

The developer ignores the letter

Instead of telling them to pound sand, they remove the package

The Third Jerk

Azer pulls down all his packages

But a good part of the Internet depends on the package

This is why you shouldn't unpublish

And he shouldn't have, either

https://bit.ly/408PKJn

The Resolution

Unwinding the whole mess was hard

Eventually, npmjs did the (almost) unthinkable – they took ownership of left-pad and re-published it

Three people being jerks

Azer was the least to blame...maybe I would do the same?

I sure wouldn't trust npmjs.org to safeguard my open-source project

Where Artifactory Fits in All This

Where this fits in terms of the build

We'd be hitting our downstream Artifactory feed

So left-pad would be *cached*

Down for everyone else...

No problem for us, though

This may never happen again

The Punchline

left-pad.js

```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

Package Stores

The Ideal Build Environment

Our packages are coming *from Artifactory...*

Where are they going?

The ideal build is *ephemeral*

*"*Existing for a short time
and then disappearing"

We don't want secure assets sitting around

And we don't want the results of the previous build jacking with the current one

Should we clear the local cache?

Ephemerality of Workspaces

Fresh build, or deltas?

Maybe a problem with your repo

I want a completely clean workspace

Packages, especially when they're versioned, should be static

I don't want to understate the amount of work

You shouldn't build on the primary server

And the dynamic agents are *really* ephemeral

Mount a shared volume for the packages

https://bit.ly/3Ft0KrN

A *custom agent* with Azure DevOps

Run an Artifactory instance inside your network

Summary

A simple build with package restoration from Artifactory

The package that broke the Internet

How an artifact store protects us from that

How all this fits in a package ecosystem