Native Dependency Bootstrapping Phase 1 One-Page Overview

Intent

The intent of this doc is to describe the plan for the first phase towards native bootstrapping. Once we've reached agreement on this doc, we will have an actionable plan towards accomplishing phase 1 of native dependency bootstrapping. Items in this doc which are listed as "Out of Scope" are NOT out of scope for native bootstrapping, they are out of scope for phase 1. After phase 1 is accomplished, we will have more experience or lessons learned towards accomplishing the next phases of native toolset bootstrapping.

It is possible that the landscape will change w.r.t. available technologies, and we should be ready to consider re-evaluating the plan and choosing those technologies if they are mature enough for adoption. I think that the work here in phase 1 will not be entirely thrown away should that event occur (bootstrapping unsupported platforms will likely require a similar mechanism as we define here).

Scope

- CLI and Pre-CLI toolsets
- Xcopy deployable native dependencies

Out of scope for phase 1 (9/21)

- Non-Xcopy deployable dependencies.
  - ie, docker, Visual Studio, NodeJS, etc…
- Toolset detection
  - Detecting if a tool (and which version) is already installed, as well as which one should be used by the build (pre-installed vs repo specified).
- Native tool install scripts
  - We will only be providing an example installer for a particular tool (CMake) both as a sample for teams to follow, and as a proof of concept. Additional installers will be written on an as-needed basis.

Goals

- Cloning a repo and building deploys versioned native dependencies (within scope) which are available for use in the repo's build.

Plan Overview

- Create zips / tarballs of a couple of xcopy deployable native dependencies (like cmake) which are stored and publicly accessible in Azure blob storage. These will be proof of concept, and not an exhaustive list of native toolset dependencies.
- Determine how repos will define a dependency version list for native dependencies which fits into the dependency description spec or modifies it where necessary.
- Create Powershell and bash scripts (these are in-box available tools) which are capable of understanding the dependency list and downloading / extracting versioned dependencies so that they are available in the repo from a well-known (or defined) location.

Alternatives considered

- dependencies distributed as Nupkg's - Why not use nupkg's as the distribution mechanism? Nupkg's have a couple of desirable attributes including being versioned, tfm / runtime awareness, well-known format. There are a couple of downsides though.
  - All implementations of the dependency are packaged into a single unit. ie, you can't just download the Windows dependency from the package unless you package it separately. You would always bring down the Windows, Linux variants, and OSX implementations for a native dependency.
  - Native dependencies are not (yet) supported by CLI. There is some work in progress by CLI team to support global tools with native dependencies (and also repo tools), but that work is not yet available and not expected until late in the .NET Core 2.2 timeframe. We could create global tools with managed console apps that wrap native dependencies, but there is an investment there and it's throw-away work (ie, not something customers currently expect / do).
  - If we used a nupkg format, we wouldn't actually be able to use NuGet to download and extract them, they would be purely used as zip files.
- X-Plat choco - beyond the scope of what's necessary for xcopy deployable native dependencies.
- MSI's, debian packages, etc - beyond the scope of what's necessary for xcopy deployable native dependencies.

Current status

Done

- Repos list their native toolset dependencies via global.json
- We already have a working and tested version of:
  - Windows bootstrapping scripts
  - Windows common library scripts
  - Windows install sample (CMake)

Pending

- Move the current storage location for the native assets, and write guidance for teams on where to upload them in the future. They are currently stored in the native-assets container in the dotnetfeed storage account and need to be moved to the storage account in use by OS onboarding. Tracked by: #757.
- Write the Linux infrastructure scripts and CMake sample. Tracked by #749
- Update scripts and documentation to add details and remove inconsistencies. Tracked by #756.
- Handle CLI as a bootstrapped native tool. Tracked by #151.

Was this helpful?