Runtime and Package Management

An update from SPI

Overview

- Package Management Wishlist
- Technology Overview / Current State
- Why We Are Here / Next Steps

Package Management Wishlist

“Life is really simple, but we insist on making it complicated.”

— Confucius

User Experience

Don't die cryptically when there are conflicts / issues
Do provide solver tracing for debugging

Don't introduce barriers for dev environments and one-off testing
Do provide "hackable" runtimes, allow developers to produce environments for small batch testing and fast iteration

Don't require long build or setup times to establish environments
Do implement fast, lightweight environments for build, and run

Don't require all variants to be built at release time
Do recognize source packages and build new variants on-demand

Package Definitions

Don't have complex version ordering semantics
Do have number-based version ordering

Don't require all releases to go directly into production
Do support pre and post release workflows

Don't distill all packages to a version number
Do enable describing all compatibility vectors

Don't expect all software to be cleanly separated
Do support bundled/embedded software in one package (eg DCC)

Runtime Environment

Don't require all processes to share a single runtime
Do support per-process configuration of the software

Don't make identifying the current runtime software tedious
Do provide a single, predictable filesystem tree

Don't over-isolate from the host system
Do support simple access to common resources

Don't require environments to be rebuilt from scratch
Do support natural platform definition, incremental updating

Storage and Distribution

Don't require re-transfer, duplicate storage for incremental updates
Do deduplicate artifacts at the file level

Don't require mothership access in all cases
Do support localization and export/bake workflows

Consumption and Stability

Don't recommend unsafe deletion of packages

Do have safe deprecation workflows for yanking

Don't allow previously successful solves to fail or ignore situations that a human can reason as successful

Do implement a backtracking solver (that is exhaustive)

Don't recommend dynamic environments in production or workflows that can cause unexpected changes to the production environment

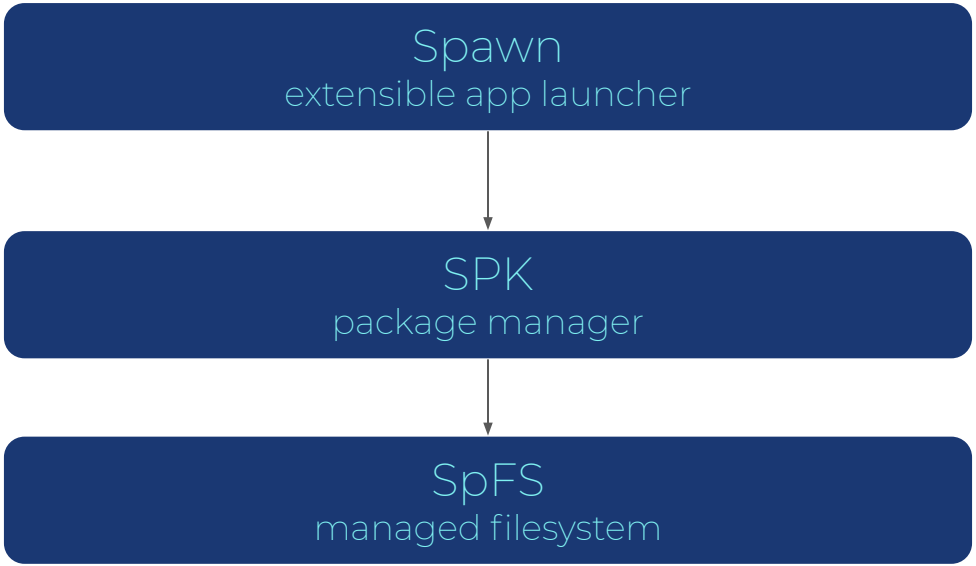
Do support deterministic storing and identification of environments

Technology Overview / Current State

“Go back?" he thought. "No good at all! Go sideways? Impossible! Go forward? Only thing to do! On we go!"

— J.R.R. Tolkien, The Hobbit, or There and Back Again

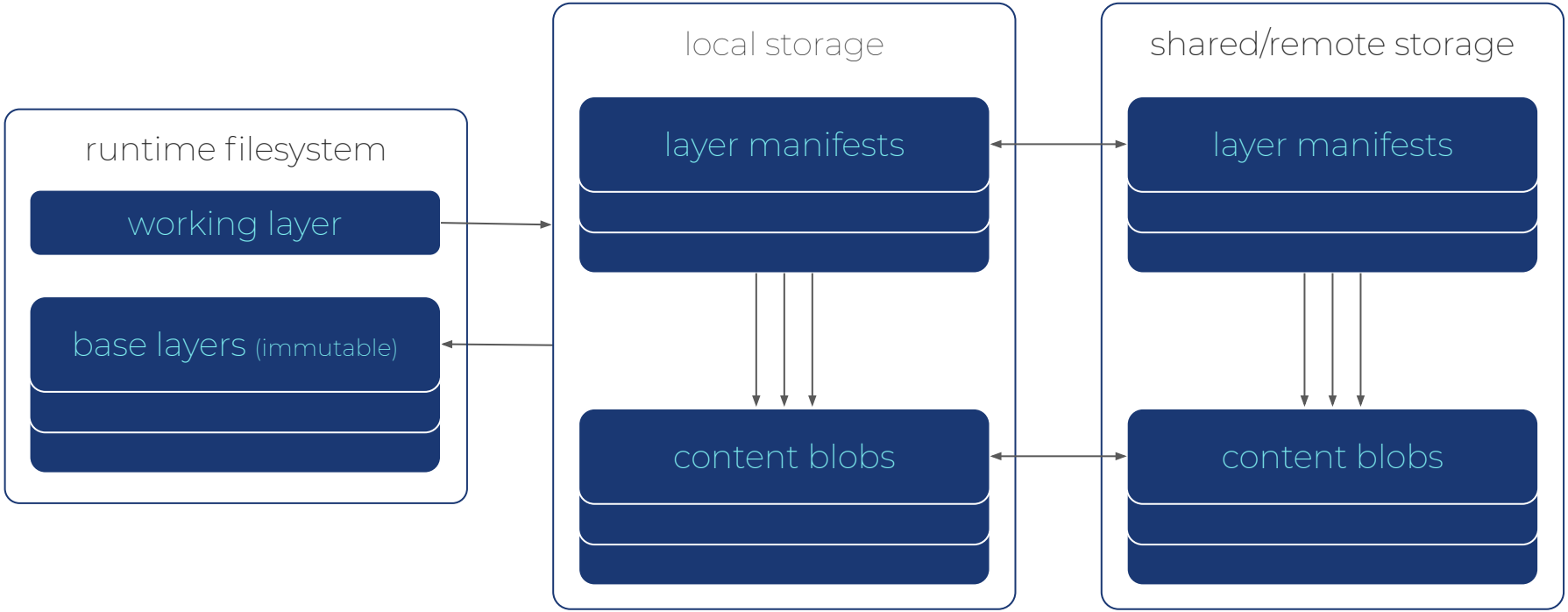
Components


Spawn
extensible app launcher
SPK
package manager
SpFS
managed filesystem

SpFS - Managed Filesystem

- Manages the contents of **`/spfs`** per-process
- Uses **overlayfs** to isolate base content from edits
- Efficient digest-based storage
- Conceptually Docker + Git

SpFS - Managed Filesystem



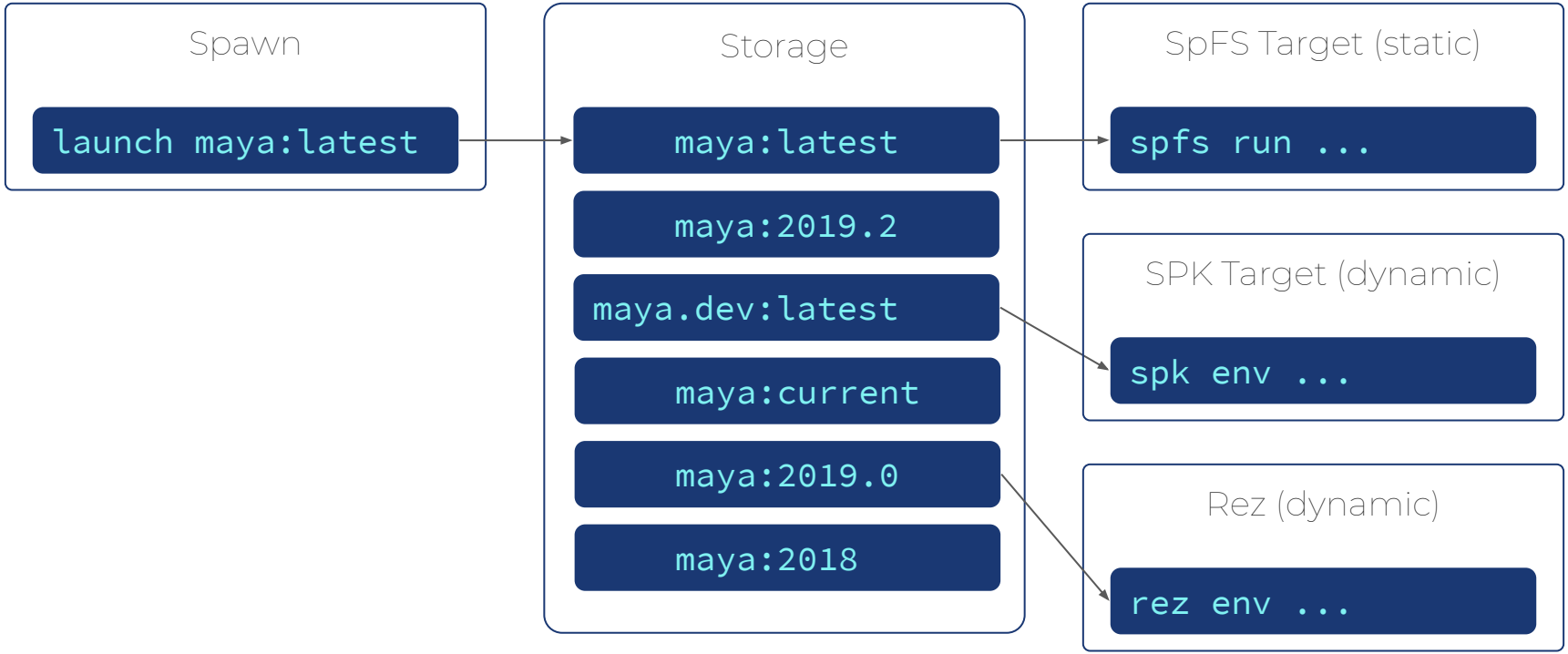
SPK - Package Manager for SpFS

- Each package is stored as an SpFS layer
- Packages are described in yaml
- Backtracking, traceable solver
- Variant support, with source packages and on-demand building
- Compatibility beyond the version number
- Deprecation workflows
- etc...

Spawn - Application Launcher

- Focused on storage/organization of launch targets
- As well as migration / testing of multiple runtime systems
- First-class representation of development environments
- Supports conversion between systems, is extendable

Spawn - Application Launcher



Recap

- Software platforms that are isolated just enough from the host
- Fully described compatibility semantics with debuggable solver
- Cloud-friendly software publish and distribution
- User-friendly launch and version management solution

Why We Are Here / Next Steps

Stronger Together...

- Recognize that these are problems that we all face
- Often these tools take a backseat to production work, and don't get the resources that they need
- We've seen that other studios have at least thought about similar solutions

The Pitch

- We are marching ahead with this software as we speak…
- And are willing to open source what we have
- We can also provide our current roadmap as a starting point for the community, and lead the development effort going forward
- We’re looking for (eventual) commitment from others before we do that…
- As we don’t want to be solely responsible for maintaining this project if it becomes public (it’s easier to do that internally)
- If there is interest, we can provide a version for initial evaluation