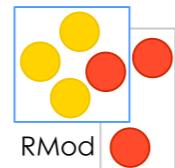




Application Development with Pharo

P. Tesone - G.Polito - 23/08/2022 - ESUG22



We want to develop in Pharo

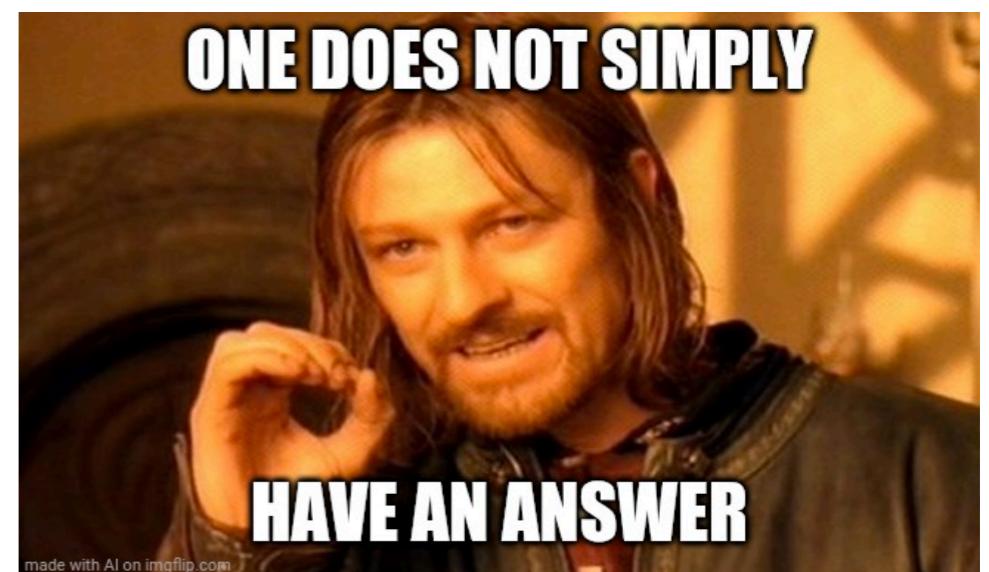
- Cool Tools
- Iterative Process
- Fun & Addictive



We want Pharo Everywhere

Not applications are the same

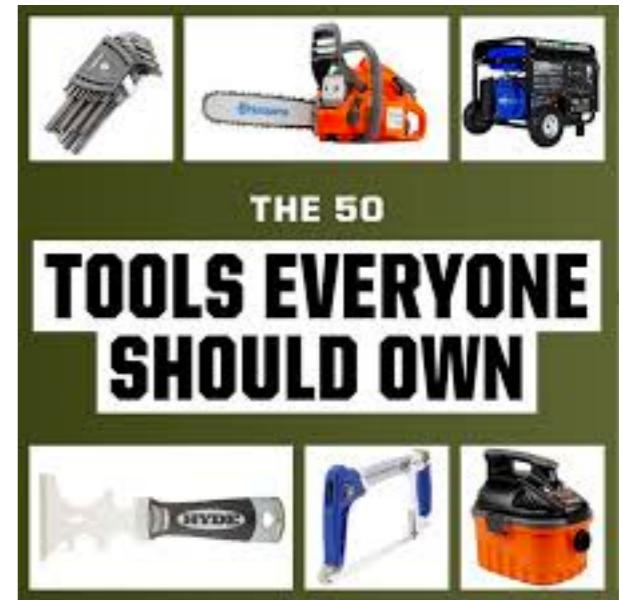
- Different User Interactions
- Different Technologies
- Not Two Applications Are the Same...



Pharo has a Rich Ecosystem



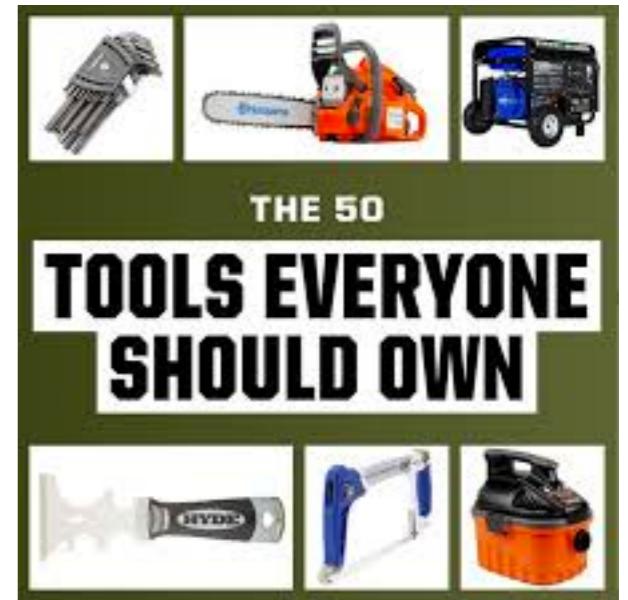
- Tools
- Frameworks
- Language Support



Pharo has a Rich Ecosystem



- Tools
- Frameworks
- Language Support



Let's See Case by Case

Web Applications



seaside *

www.seaside.st

PharJS

pharojs.org



[astares/Seaside-Bootstrap5](#)



[ba-st/Willow](#)



[ba-st/RenoirSt](#)

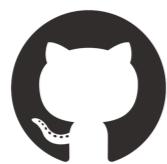
Web Applications



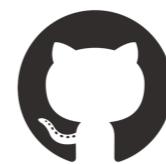
www.seaside.st



pharojs.org



[astares/Seaside-Bootstrap5](#)



[ba-st/Willow](#)



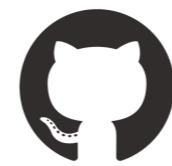
[ba-st/RenoirSt](#)



Rest Servers / Rest Clients



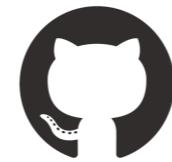
svenvc/zinc



ba-st/Superluminal



ba-st/Stargate



zeroflag/Teapot

<https://books.pharo.org/enterprise-pharo/>
<https://books.pharo.org/booklet-Zinc/>

IOT Applications

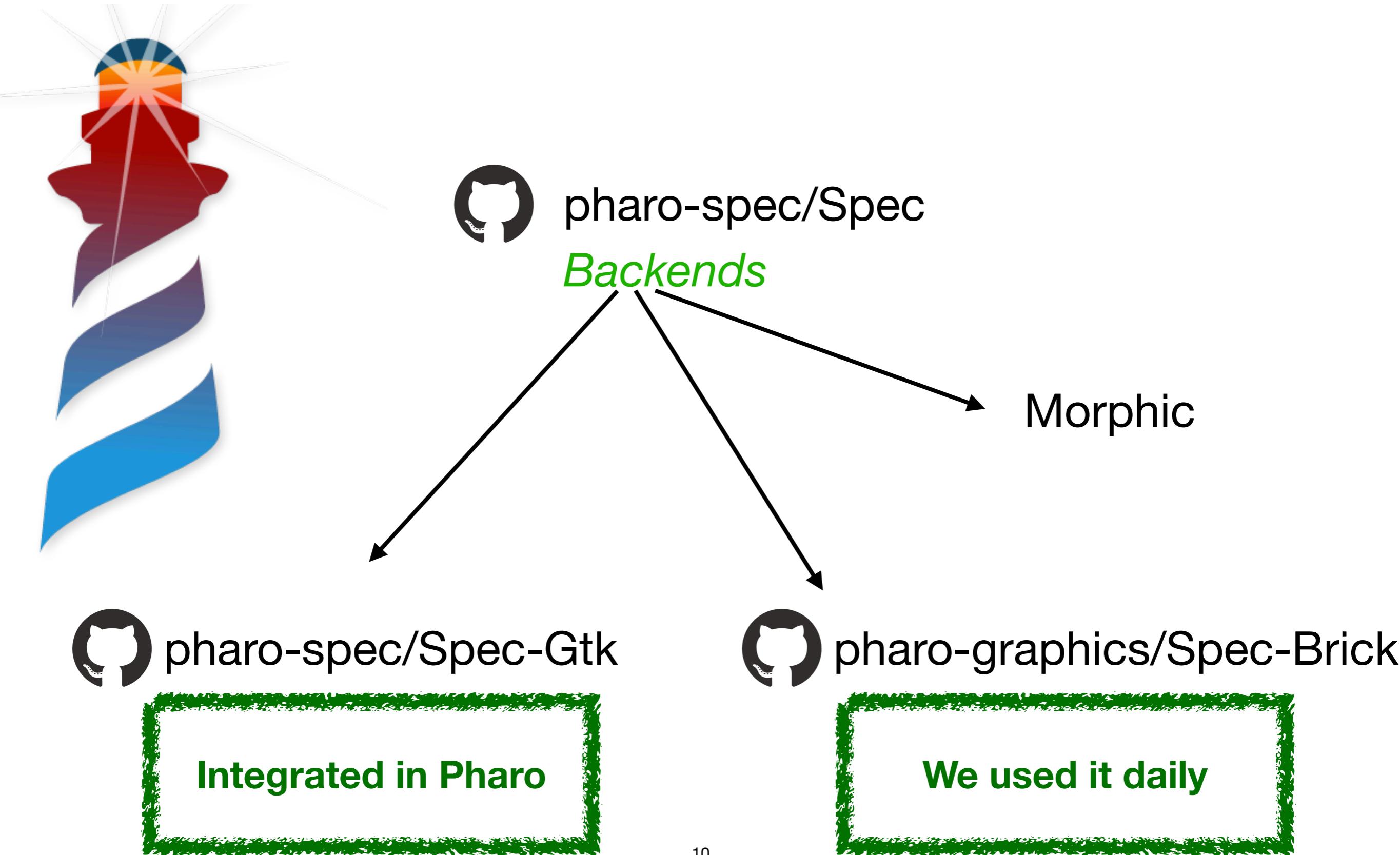


pharo-iot/PharoThings

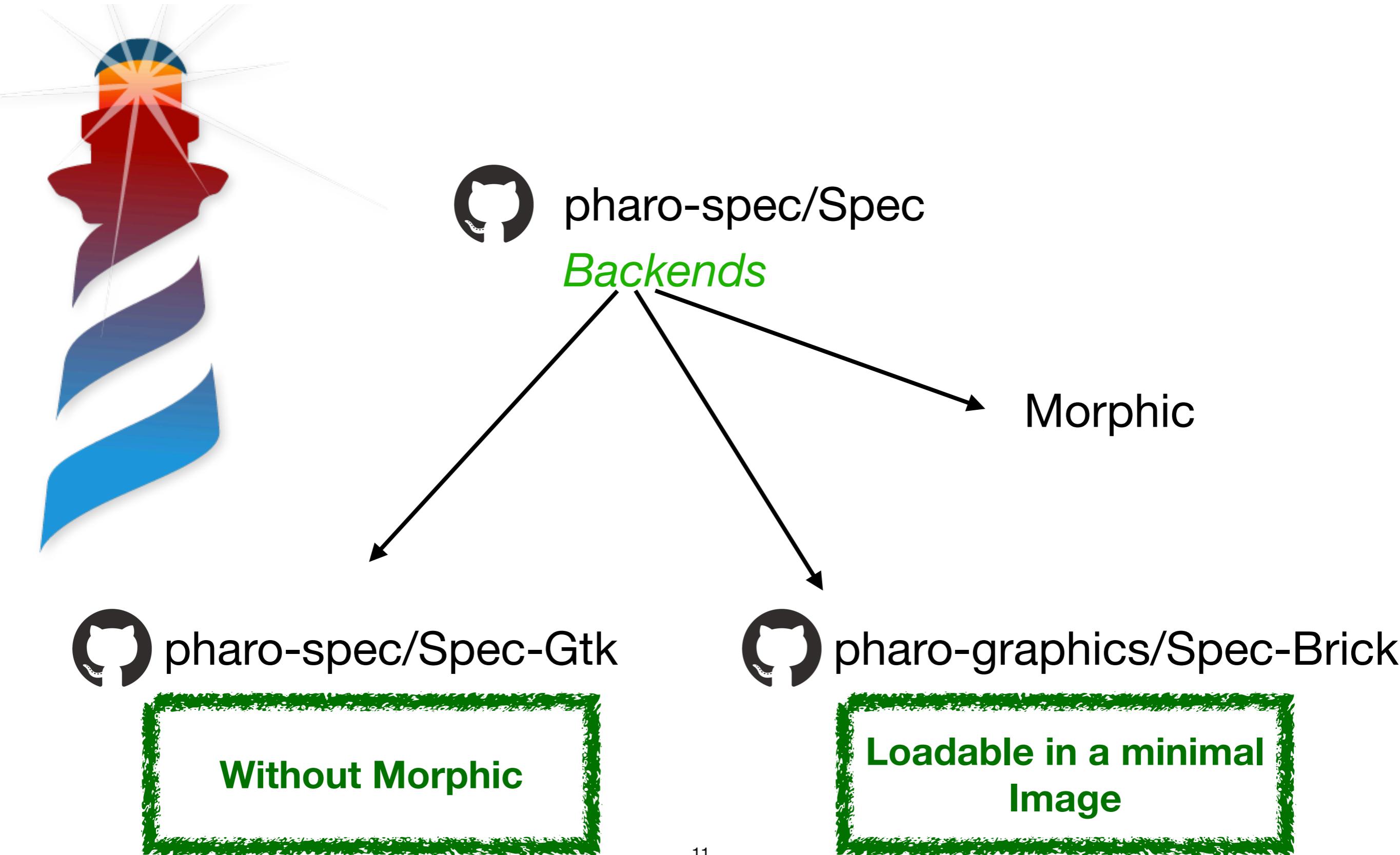


SquareBracketAssociates/Booklet-APharoThingsTutorial

Desktop Applications



Desktop Applications



Command Line Applications

- Extended Support for:
 - VTerm Colors
 - Command Line Parsing
 - Headless mode
 - VM Without dependencies on the Graphic UI



pharo-contributions/clap-st

Loaded in Pharo



We have our application developed...



We are done... let's go party



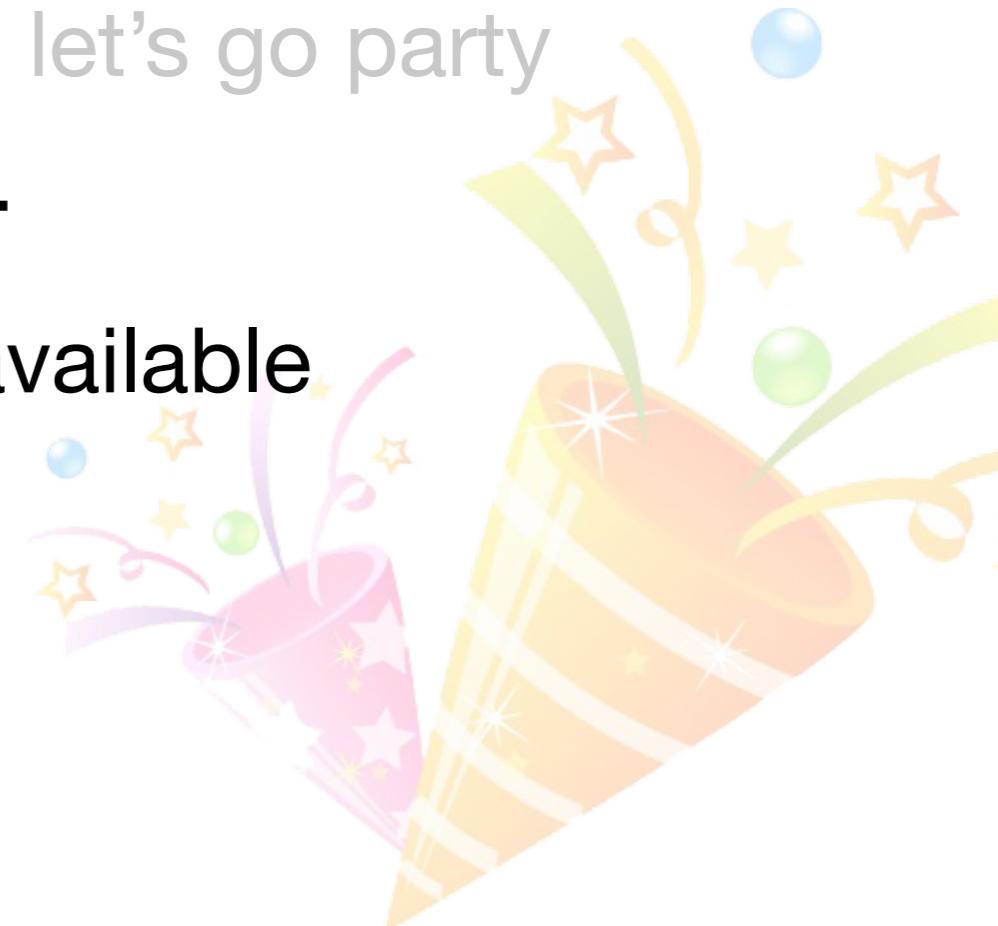
We have the application developed...



...let's go party

Not So Fast...

We need to make it available
to the users



Once the Application is developed...

- 
- We Need to think on:
 - Making a Release Version
 - Branding
 - Verification
 - Distribution
 - Deployment
 - Errors Handling

Once the Application is developed...

- 
- We Need to think on:
 - Making a Release Version
 - Branding
 - Verification
 - Distribution
 - Deployment
 - Errors Handling

We will not see these...



Once the Application is developed...

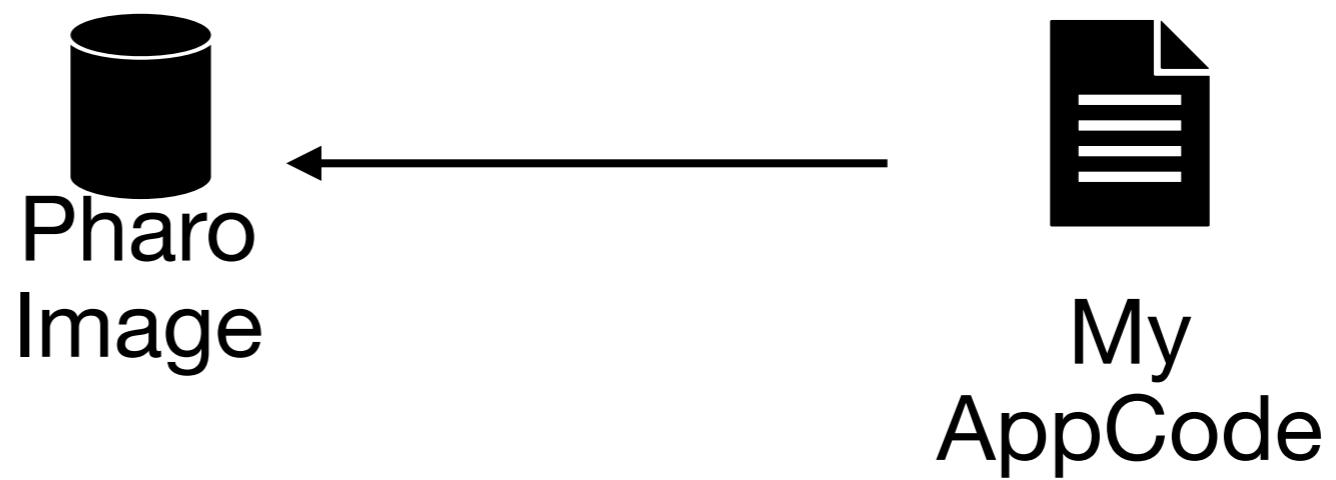
- We Need to think on:

- Making a Release Version
- Branding
- Verification
- Distribution
- Deployment
- Errors Handling

We will see these

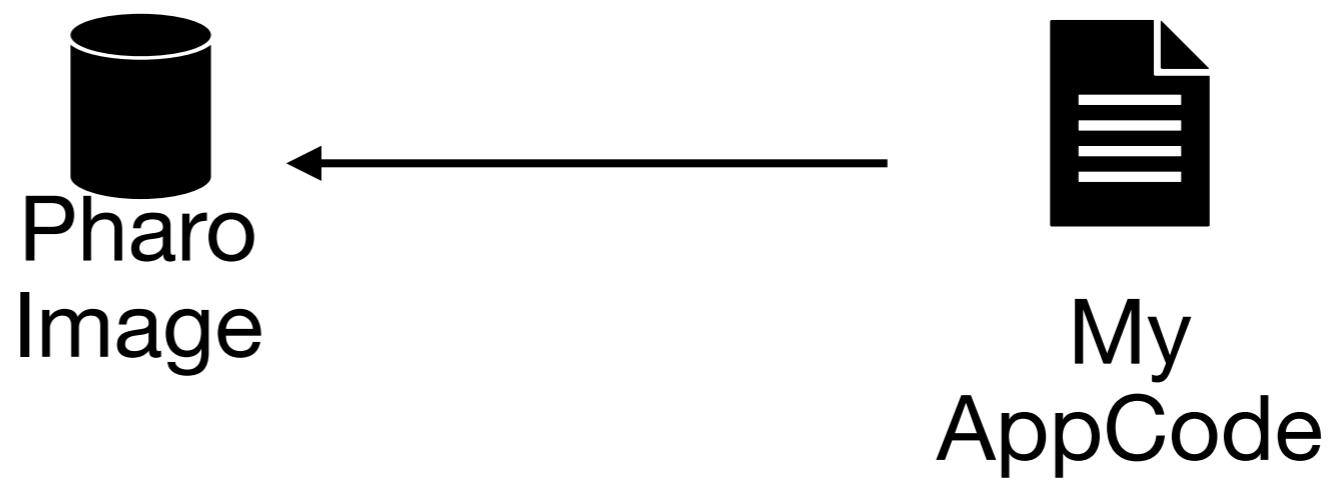
Making a Release Version

Let's create an image



Making a Release Version

Let's create an image



Do we want to
distribute the whole
Pharo Image?

Making a Release Version

Let's create an image

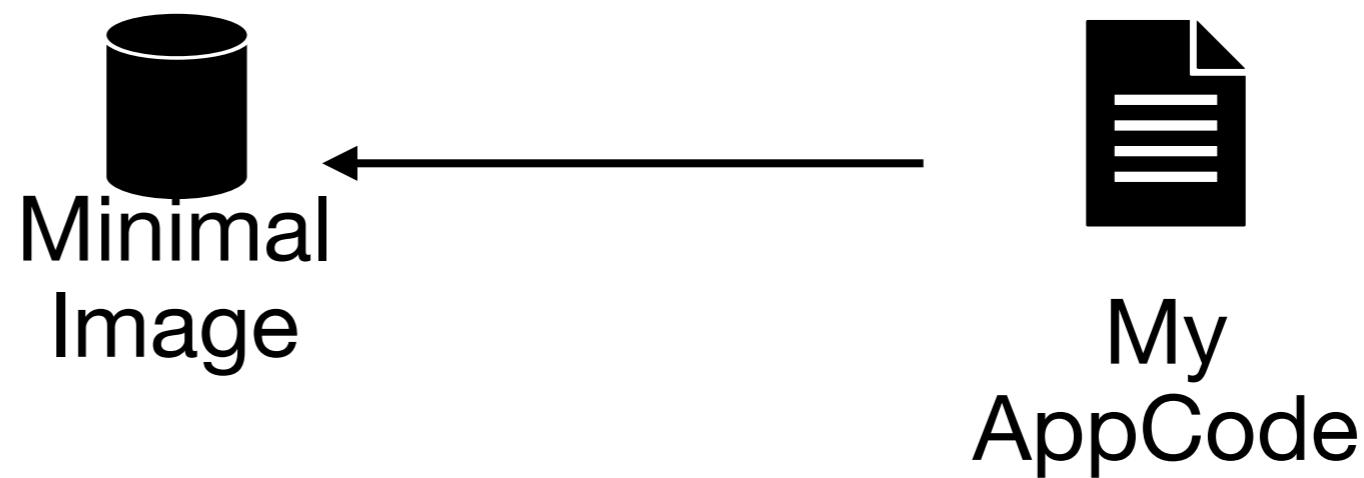


Do we want to
distribute the whole
Pharo Image?

Tools?...
UI?...
Debugger?

Making a Release Version

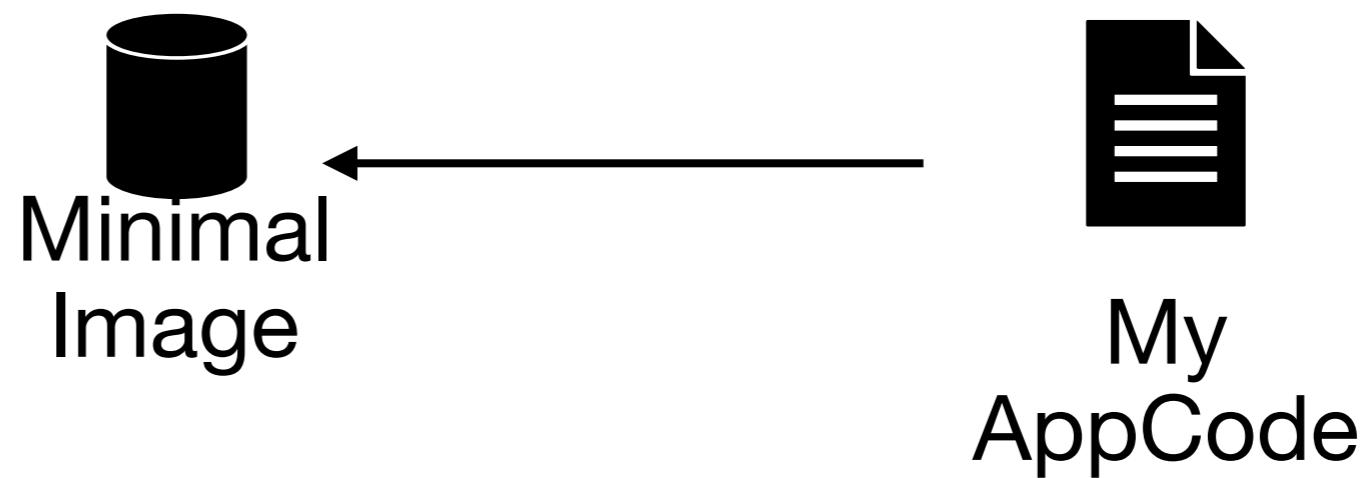
Let's create an image



We can use the
minimal Image and
load what we need

Making a Release Version

Let's create an image

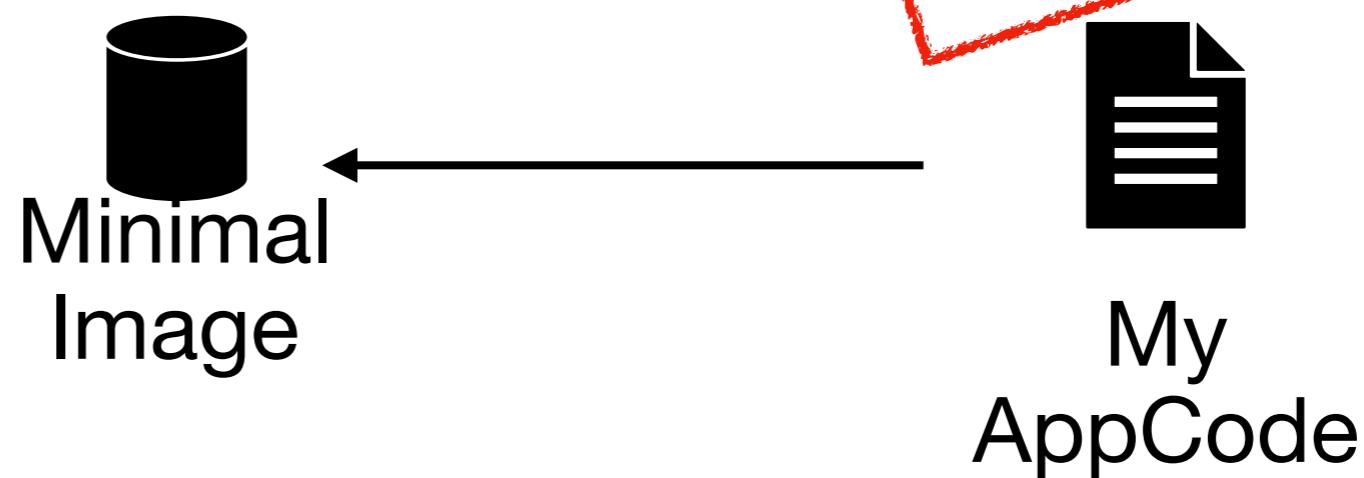


We can use the
minimal Image and
load what we need

The minimal Image is
generated during the
bootstrap

Making a Release Version

Let's create an image



We can use the minimal Image and load what we need

The minimal Image is generated during the bootstrap

Branding

Make your App look like it is your App

- Icons
- Resources (App Metadata)
- My App Executable
- The remaining stuff:
 - Main window open or not,
 - application title,
 - additional windows,
 - about dialog,
 - etc...



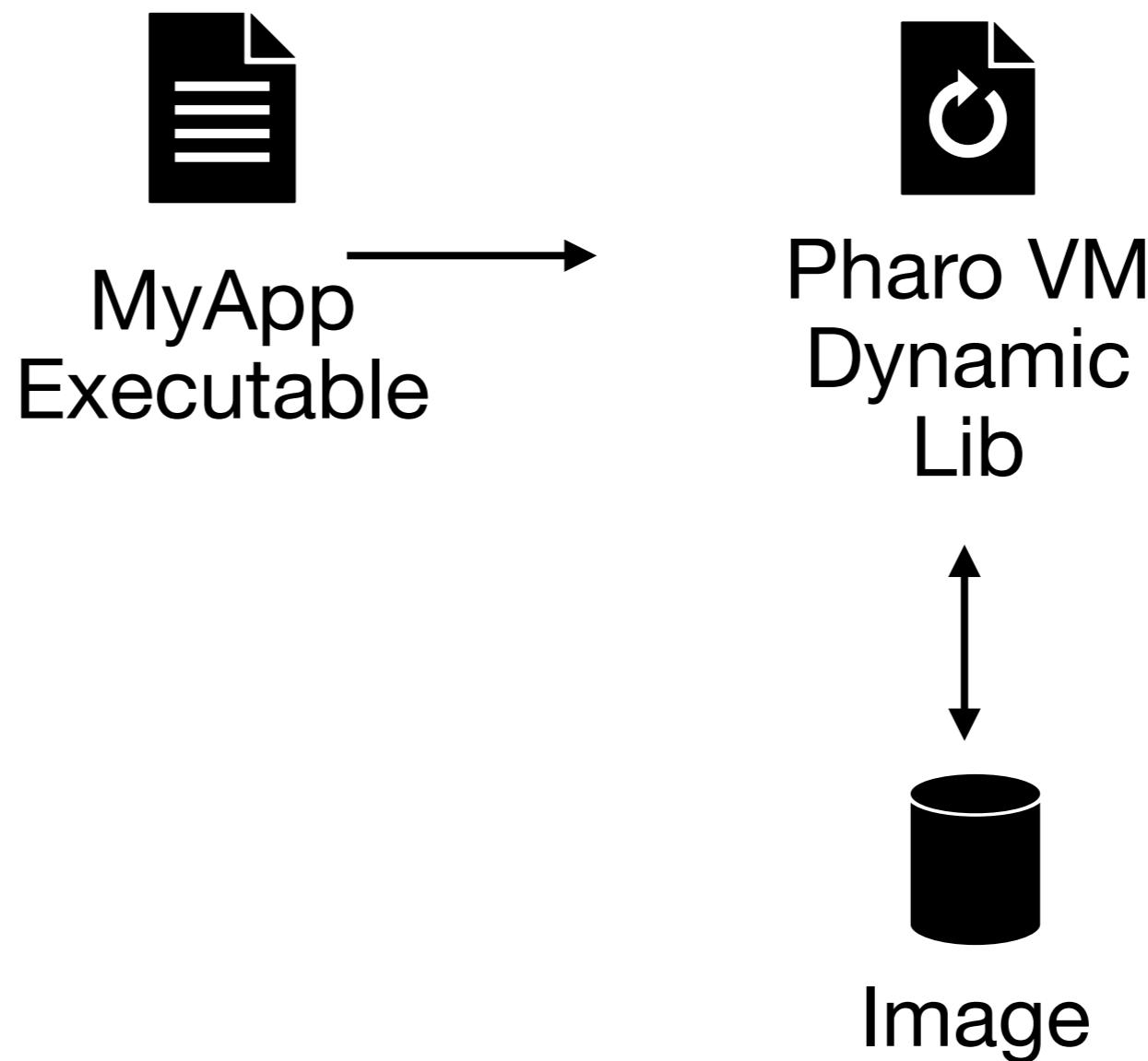
My APP as a Thin Layer

- 
- My Own Icons
 - My Own information
 - Built using Pharo VM as a library

Branding

Proposed Architecture

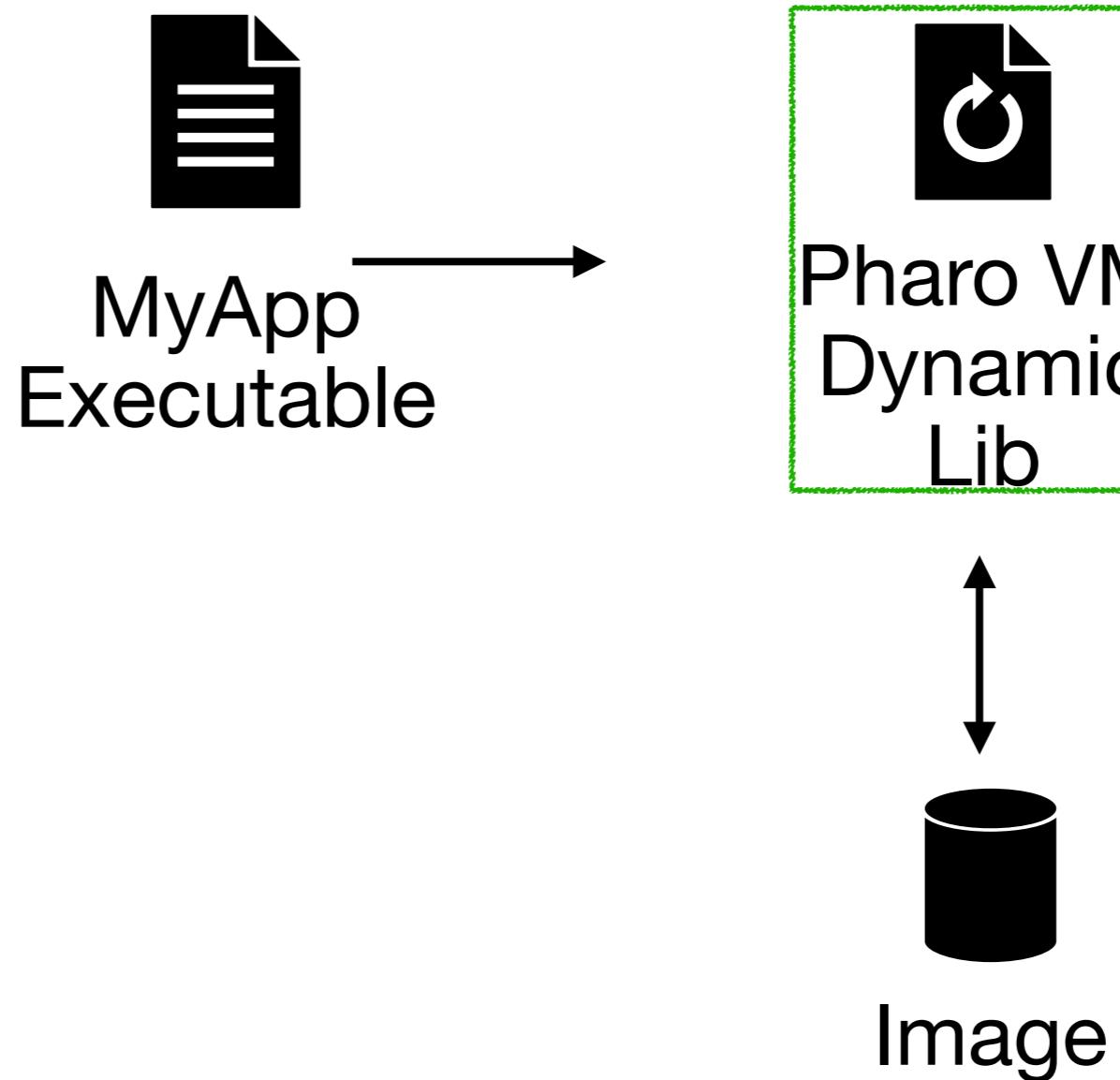
- Small



Branding

Proposed Architecture

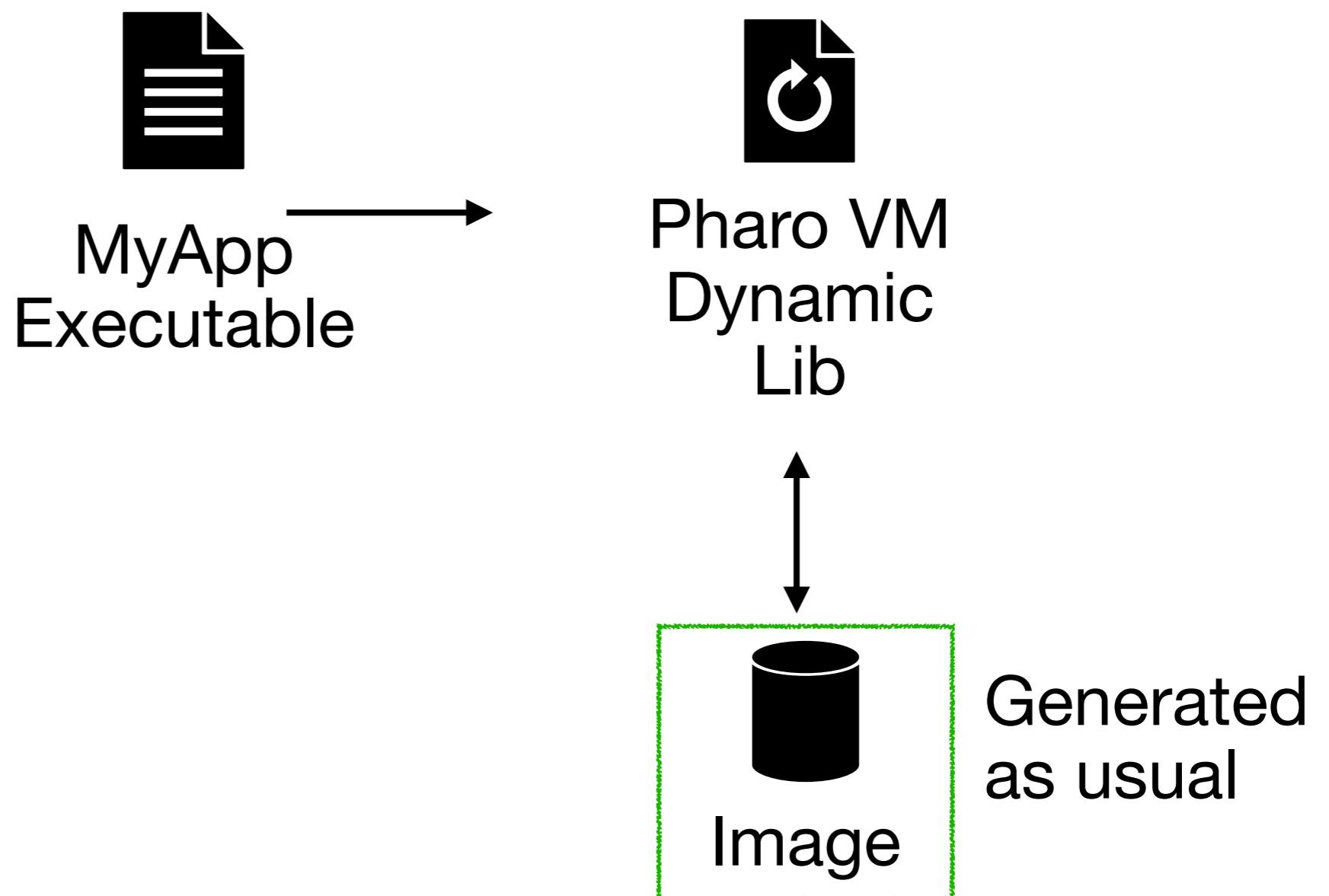
- Small



Branding

Proposed Architecture

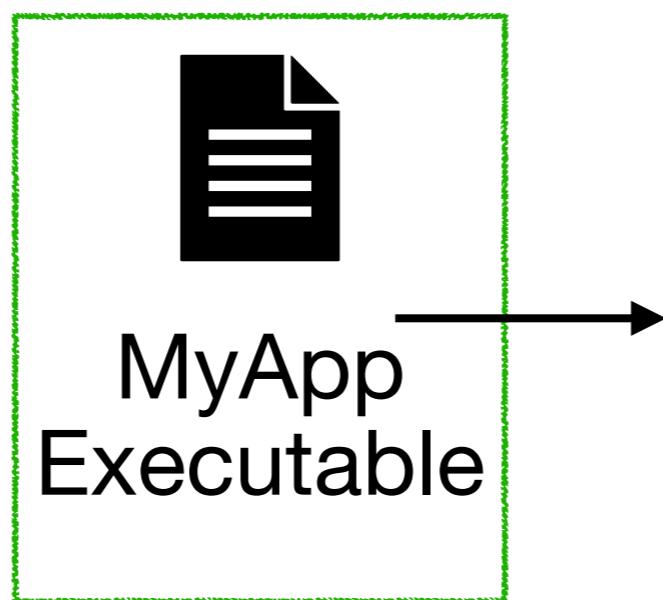
- Small



Branding

Proposed Architecture

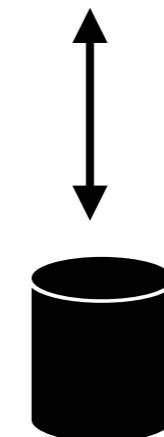
- Small



Customizable
Build



Pharo VM
Dynamic
Lib



Image



How to Implement it...

- A Simple CMake Script and some simple files



[tesonep / pharo-vm-embedded-example](#)

master 1 branch 0 tags Go to file Add file Code

tesonep	Update README.md	d4b05d5 on 13 Jan 2020 7 commits
images	Adding images	2 years ago
include	Initial commit	2 years ago
scripts	Adding smalltalk counterpart & finishing example	2 years ago
smalltalk-src	Adding smalltalk counterpart & finishing example	2 years ago
src	Adding smalltalk counterpart & finishing example	2 years ago
.project	Adding smalltalk counterpart & finishing example	2 years ago
CMakeLists.txt	Adding smalltalk counterpart & finishing example	2 years ago
README.md	Update README.md	2 years ago

README.md

Pharo VM Example: Showing how to embed an image.

This repository have an example of how to create a PharoVM with an embedded image.

In this example we create an application based on the `SDL2AthensDrawingExample`. This example application, opens a `SDL` windows where it is possible to draw with the mouse.

The Pharo image is contained in the executable and it is modified so it does not write in the disk. Also, it shows how to restrict the command line arguments and how to set a new icon and resources.

The example is only for Windows applications, but it can easily extended to other platforms.

My Thin App

60 lines of code with comments

- Just a Main Function



```
/*
 * I am creating a VMParameters with the information
 * that I want to send to the image.
 */
VMParameters parameters = {};
parameters.processArgc = 4;
parameters.processArgv = (const char**)args;
parameters.environmentVector = env;

/**
 * I have to set the first argument correctly as this one is used
 * by the VM to detect if launched from the console or from the desktop.
 */
args[0] = argv[0];

parameters.imageFileName = "Pharo.image";
parameters.isDefaultImage = true;
parameters.defaultImageFound = true;

/*
 * The set of arguments to pass to the image.
 */
char* args[] = {"", "Pharo.image", "embeddedExample", "--embedded"};
```

```
/*
 * I pass "made up" parameters to the VM to handle them.
 * In this case to handle the logic of the '--logLevel' parameter we have to call this function
 * To give the VM the opportunity of parsing the log parameter
 */
vm_parameters_parse(4, (const char**)args, &parameters);

/*
 * I force the vm to start in a non interactive Session.
 * As the VM tries to detect if launched from the console or from the desktop.
 * In an interactive session the image opens a window with the Pharo World.
 */
parameters.isInteractiveSession = false;

int exitCode = vm_main_with_parameters(&parameters);
vm_parameters_destroy(&parameters);
return exitCode;
```

Some Resources

- 
- In Windows:
 - A Resource file with icon information & Metadata of the application (Developer, version, etc)
 - In OSX:
 - A PList with information about the icons, file associations and metadata of the application.

What else...



```
cmake .  
make
```

- Downloads Pharo VM
- Build Thin Executable
- Integrate Resources

Verification

- 
- Applications should be signed
 - Signing should be done by the developer
 - All executing code should be signed

Verification

- 
- Applications should be signed
 - Signing should be done by the developer
 - All executing code should be signed

We have to assure
that our applications
is not tainted

Verification

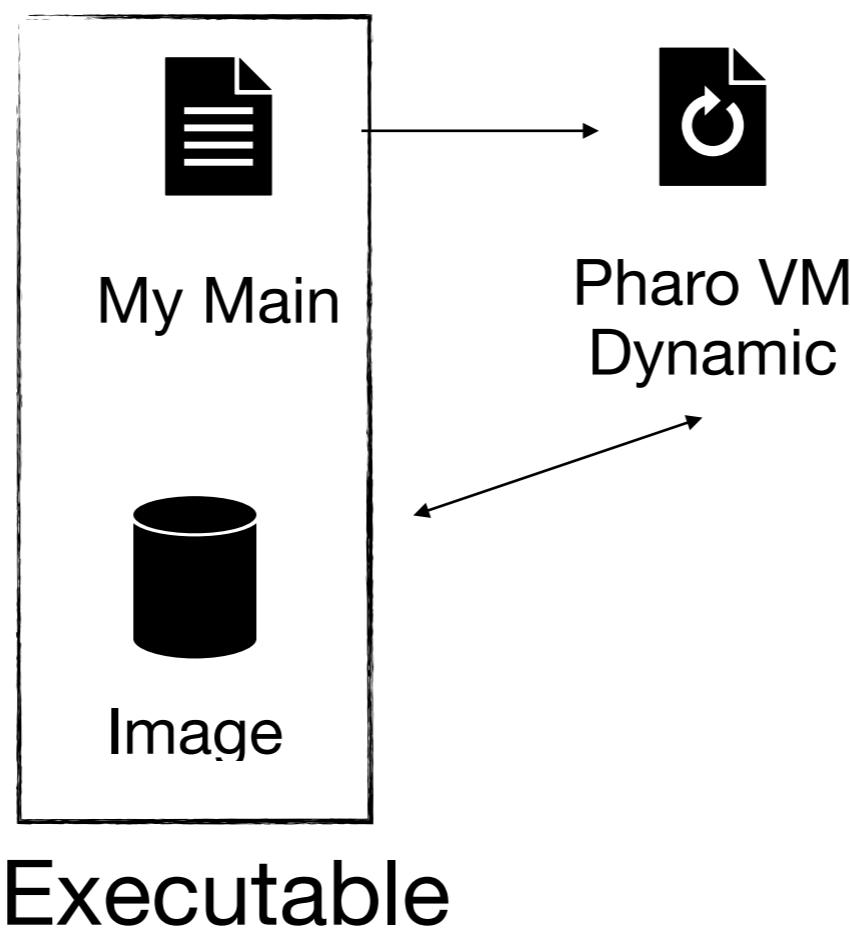
- 
- Applications should be signed
 - Signing should be done by the developer
 - All executing code should be signed

We have to assure
that our applications
is not tainted

What we do with the
image?
The image is
executable code...

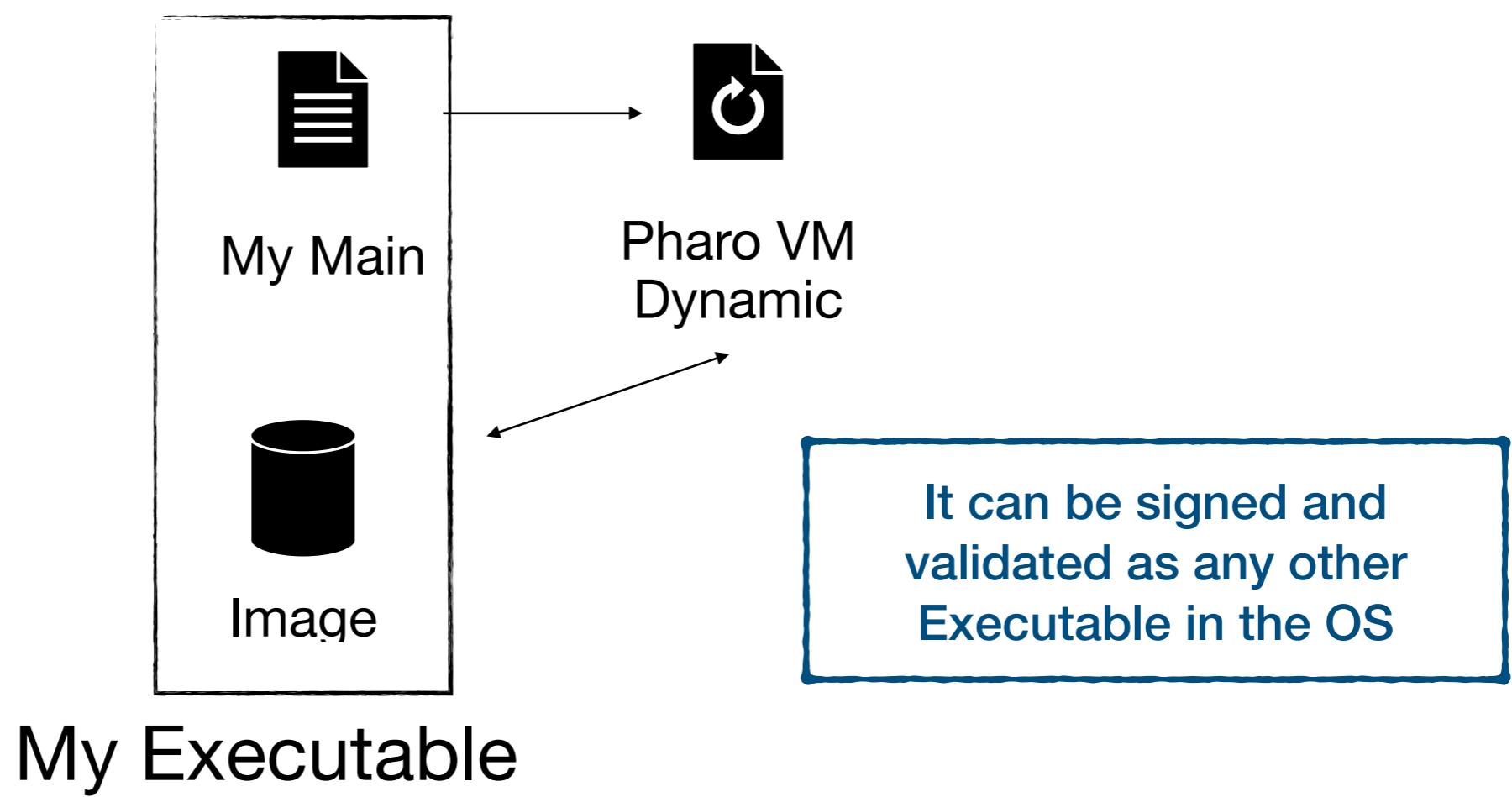
Alternative 1: Embedding as a Resource

- If the image not change we can embed it as a resource.



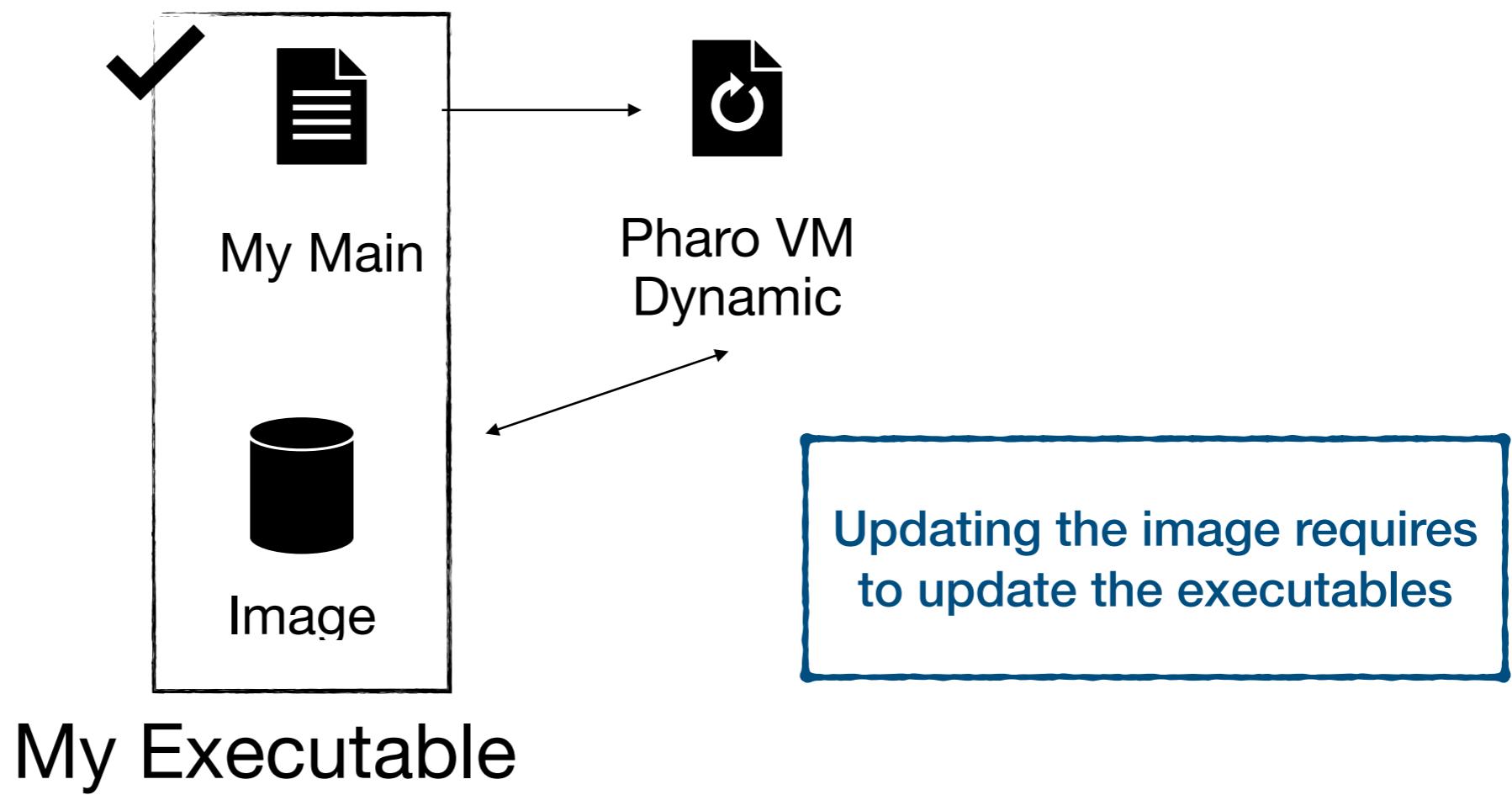
Alternative 1: Embedding as a Resource

- If the image not change we can embed it as a resource.



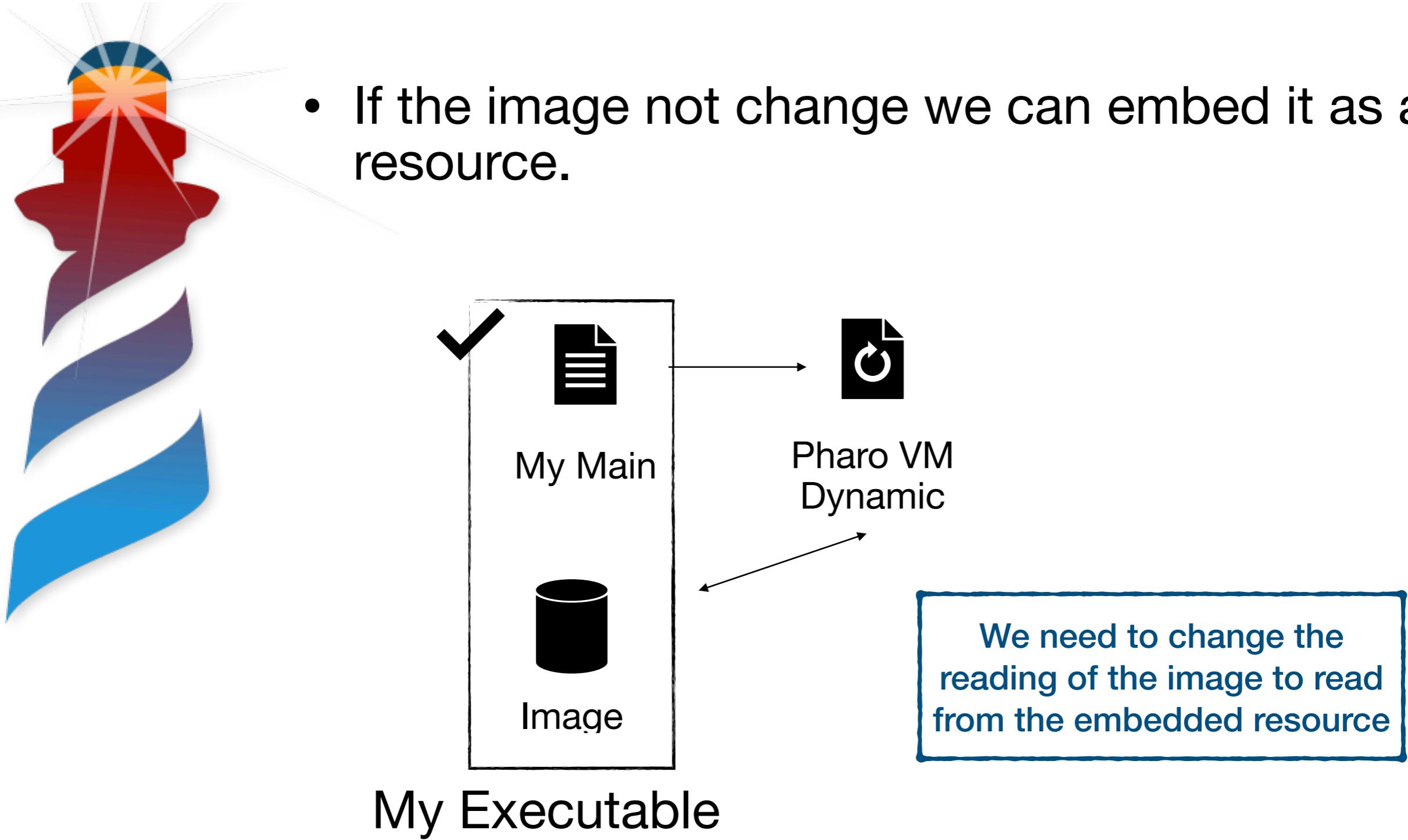
Alternative 1: Embedding as a Resource

- If the image not change we can embed it as a resource.

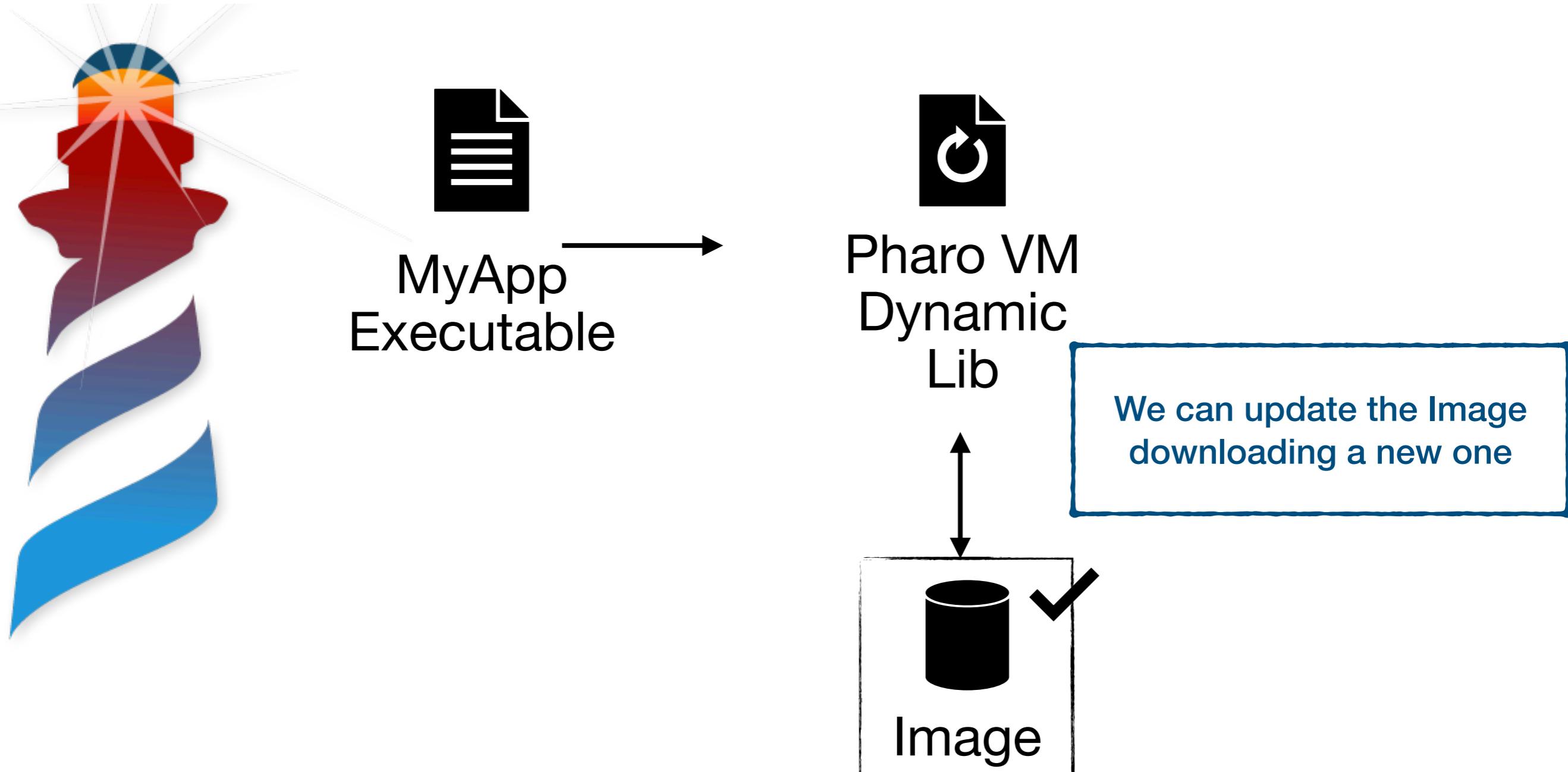


Alternative 1: Embedding as a Resource

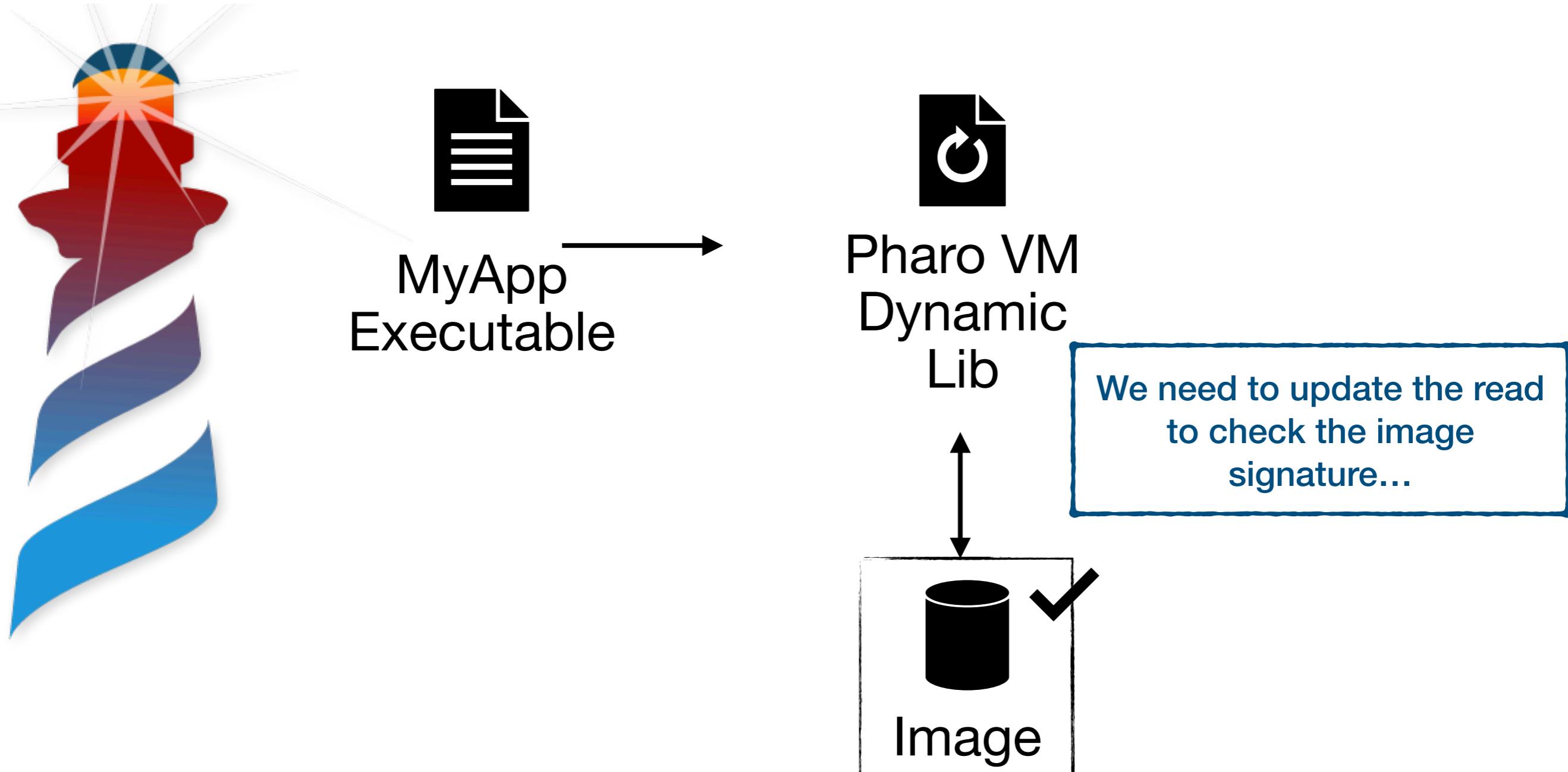
- If the image not change we can embed it as a resource.



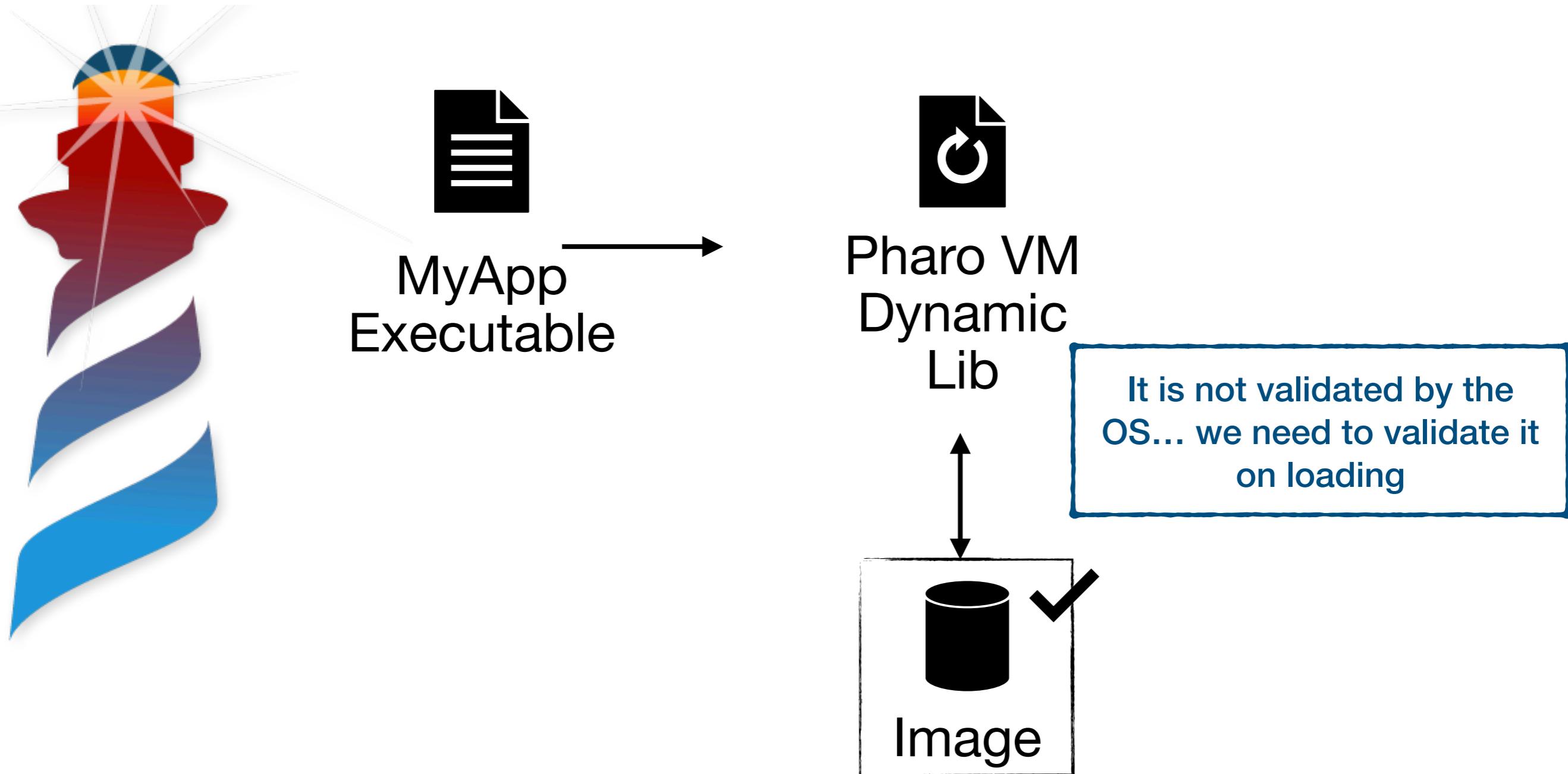
Alternative 2: We sign it outside the executable Proposed Architecture



Alternative 2: We sign it outside the executable Proposed Architecture



Alternative 2: We sign it outside the executable Proposed Architecture



Implementing Alternative 1

Same Repository...



[tesonep / pharo-vm-embedded-example](#)

```
typedef struct {
    sqInt (*imageFileClose)(sqImageFile f);

    sqImageFile (*imageFileOpen)(const char* fileName, char *mode);
    long int (*imageFilePosition)(sqImageFile f);
    size_t (*imageFileRead)(void * ptr, size_t sz, size_t count, sqImageFile f);

    int (*imageFileSeek)(sqImageFile f, long int pos);
    int (*imageFileSeekEnd)(sqImageFile f, long int pos);
    size_t (*imageFileWrite)(void* ptr, size_t sz, size_t count, sqImageFile f);
    int (*imageFileExists)(const char* aPath);
    void (*imageReportProgress)(size_t totalSize, size_t currentSize);
} _ FileAccessHandler;

typedef _ FileAccessHandler FileAccessHandler;
```

```
EXPORT(FileAccessHandler) embedded FileAccess = {
    embeddedImageFileClose,
    embeddedImageFileOpen,
    embeddedImageFilePosition,
    embeddedImageFileRead,
    embeddedImageFileSeek,
    embeddedImageFileSeekEnd,
    embeddedImageFileWrite,
    embeddedImageFileExists
};
```

```
/**
 * I will replace the access to the file with the ones in the embeddedImage.c file
 * This functions handles the reading of the image from the resources
 */
setFileAccessHandler(&embedded FileAccess);
```

Implementing Alternative 1

Same Repository...

 [tesonep / pharo-vm-embedded-example](#)

```
typedef struct {
    sqInt (*imageFileClose)(sqImageFile f);

    sqImageFile (*imageFileOpen)(const char* fileName, char *mode);
    long int (*imageFilePosition)(sqImageFile f);
    size_t (*imageFileRead)(void * ptr, size_t sz, size_t count, sqImageFile f);

    int (*imageFileSeek)(sqImageFile f, long int pos);
    int (*imageFileSeekEnd)(sqImageFile f, long int pos);
    size_t (*imageFileWrite)(void* ptr, size_t sz, size_t count, sqImageFile f);
    int (*imageFileExists)(const char* aPath);
    void (*imageReportProgress)(size_t totalSize, size_t currentSize);
} _FileAccessHandler;

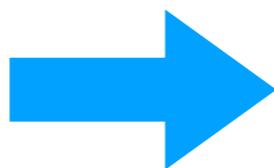
typedef _FileAccessHandler FileAccessHandler;
```

We implement the verification
of the image in the same
fashion...

```
/**
 * I will replace the access to the file with the ones in the embeddedImage.c file
 * This functions handles the reading of the image from the resources
 */
setFileAccessHandler(&embedded FileAccess);
```

Distribution / Installation

- 
- Windows
 - OSX

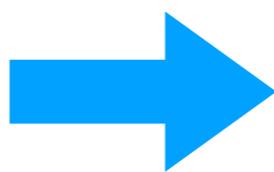


We have a normal application, we can distribute as usual

Distribution / Installation

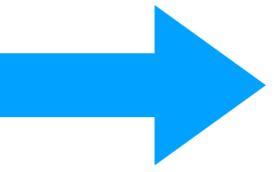


- Windows
- OSX



We have a normal application, we can distribute as usual

- Linux



OBS

Open Build Service

<https://openbuildservice.org/>

- Attack Distributions Differences
 - Build for many architectures at the same time.
- Our Package: ***devel:languages:pharo:latest/pharo9***



Open Build Service

<https://openbuildservice.org/>



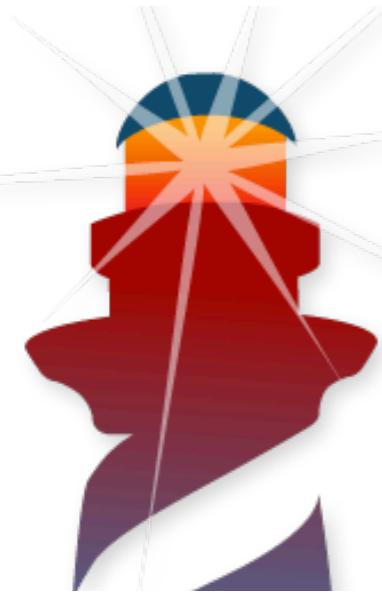
- Attack Distributions Differences
 - Build for many architectures at the same time.

Our Package: *devel:languages:pharo:latest/pharo9*

Arch	xUbuntu_20.10	Raspbian_11	Fedora_35	Debian_Testing
⚠ aarch64	📦 aarch64 ? succeeded: 4 ? disabled: 1	📦 armv7l ? disabled: 5	📦 aarch64 ? disabled: 5	📦 x86_64 ? succeeded: 4 ? disabled: 1
📦 x86_64	📦 x86_64 ? succeeded: 4 ? disabled: 1	📦 aarch64 ? succeeded: 3 ? failed: 1 ? disabled: 1	📦 armv7l ? disabled: 5 📦 ppc64le ? disabled: 5	Fedora_32
Debian_10	xUbuntu_21.04	📦 armv7l ? succeeded: 3 ? failed: 1 ? disabled: 1	📦 x86_64 ? succeeded: 3 ? disabled: 2	📦 aarch64 ? disabled: 5
📦 i586	📦 aarch64 ? succeeded: 4 ? disabled: 1	openSUSE_Leap_15.2	Fedora_36	📦 armv7l ? disabled: 5
📦 x86_64	📦 x86_64 ? succeeded: 4 ? disabled: 1	📦 x86_64 ? succeeded: 4 ? disabled: 1	📦 ppc64le ? disabled: 5	📦 ppc64le ? disabled: 5
Debian_11	xUbuntu_21.10	openSUSE_Leap_15.3	📦 x86_64 ? succeeded: 2 ? disabled: 3	📦 x86_64 ? succeeded: 3 ? disabled: 2
📦 i586	📦 x86_64 ? succeeded: 4 ? disabled: 1	📦 x86_64 ? succeeded: 4 ? disabled: 1	Raspbian_10	Fedora_33
📦 x86_64	xUbuntu_22.04	openSUSE_Tumbleweed	📦 aarch64 ? succeeded: 4 ? disabled: 1	📦 aarch64 ? disabled: 5
Debian_9.0	📦 x86_64 ? succeeded: 2 ? disabled: 3	📦 i586 ? disabled: 5	📦 armv7l ? succeeded: 4 ? disabled: 1	📦 armv7l ? disabled: 5
📦 i586	📦 x86_64 ? succeeded: 2 ? disabled: 3	📦 x86_64 ? succeeded: 2	Fedora_34	📦 ppc64le ? disabled: 5
📦 x86_64				📦 x86_64 ? succeeded: 3 ? disabled: 2

Open Build Service

<https://openbuildservice.org/>



We are using... great for
creating packages

- Attack Distributions Differences
 - Build for many architectures at the same time.

Our Package: *devel:languages:pharo:latest/pharo9*

Arch	xUbuntu_20.10	Raspbian_11	Fedora_35	Debian_Testing
aarch64	aarch64 succeeded: 4 disabled: 1	armv7l disabled: 5	aarch64 disabled: 5	x86_64 succeeded: 4 disabled: 1
x86_64	x86_64 succeeded: 4 disabled: 1	Raspbian_9.0	armv7l disabled: 5	Fedora_32
Debian_10	xUbuntu_21.04	aarch64 succeeded: 3 failed: 1 disabled: 1	ppc64le disabled: 5	aarch64 disabled: 5
i586	aarch64 succeeded: 4 disabled: 1	armv7l succeeded: 3 failed: 1 disabled: 1	x86_64 succeeded: 3 disabled: 2	armv7l disabled: 5
x86_64	x86_64 succeeded: 4 disabled: 1	openSUSE_Leap_15.2	Fedora_36	ppc64le disabled: 5
Debian_11	xUbuntu_21.10	x86_64 succeeded: 4 disabled: 1	aarch64 disabled: 5	x86_64 succeeded: 3 disabled: 2
i586	x86_64 succeeded: 4 disabled: 1	openSUSE_Leap_15.3	armv7l disabled: 5	Fedora_33
x86_64	xUbuntu_22.04	x86_64 succeeded: 4 disabled: 1	ppc64le disabled: 5	aarch64 disabled: 5
Debian_9.0	x86_64 succeeded: 2 disabled: 3	openSUSE_Tumbleweed	x86_64 succeeded: 2 disabled: 3	armv7l disabled: 5
i586	i586 disabled: 5	i586 disabled: 5	Raspbian_10	ppc64le disabled: 5
x86_64	x86_64 succeeded: 3	x86_64 succeeded: 2	aarch64 succeeded: 4 disabled: 1	x86_64 succeeded: 3 disabled: 2
			armv7l succeeded: 4 disabled: 1	Fedora_34
				aarch64 disabled: 5
				armv7l disabled: 5

Open Build Service

<https://openbuildservice.org/>

- 
- Attack Distributions Differences
 - Build for many architectures at the same time.

Our Package: ***devel:languages:pharo:latest/pharo9***

Ideal for Open source
Packages... you can use
OpenSuse Infra

The packages can be
integrated in default
repositories

For Non-OpenSource easy to
run in-house

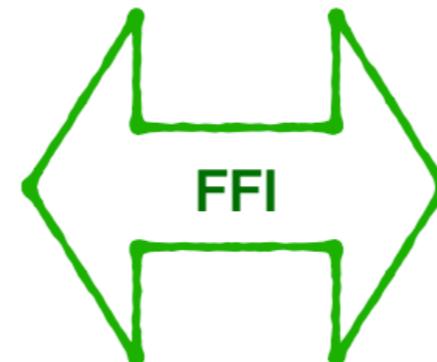
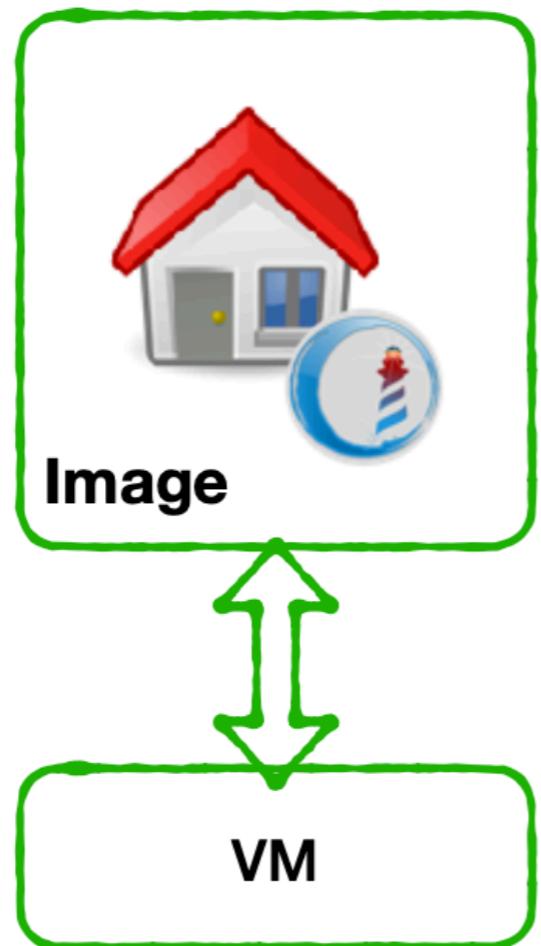
It provides a native repository
supporting updates



Bonus: Remember FFI

FFI: Opening the game to external libraries

Let's not reinvent the wheel



External Libraries

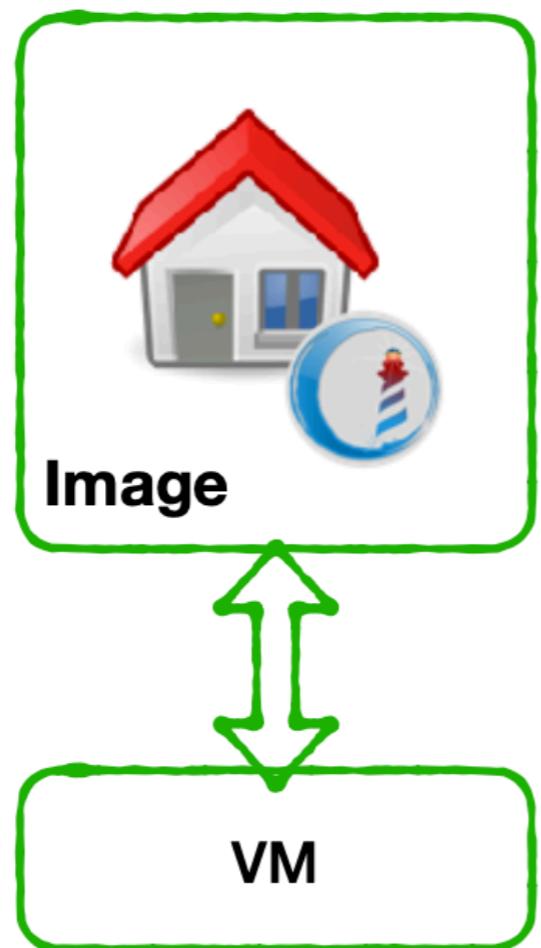


Operating System API

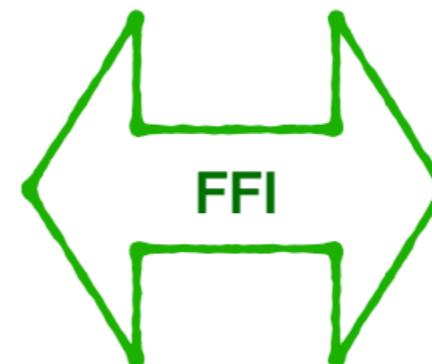
We can communicate with anything that
has a C API

FFI: Opening the game to external libraries

Let's not reinvent the wheel



Reusing Libraries



External Libraries

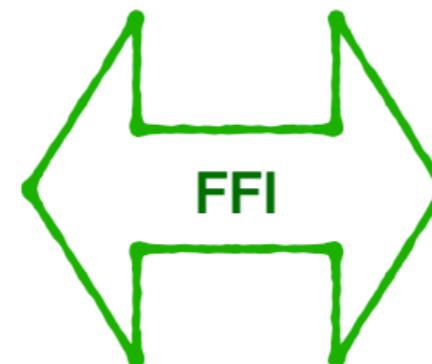
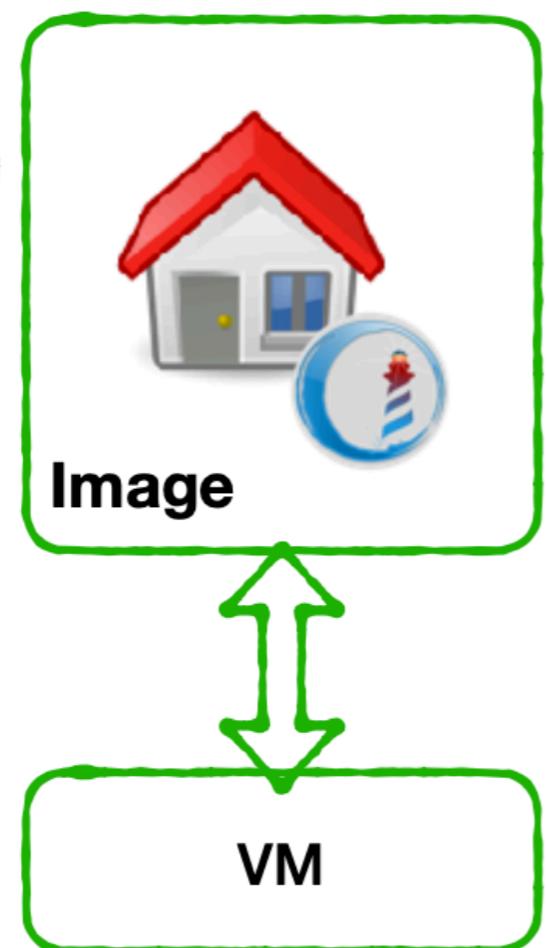


Operating System API

Interacting with
the OS

FFI: Opening the game to external libraries

Let's not reinvent the wheel



External Libraries



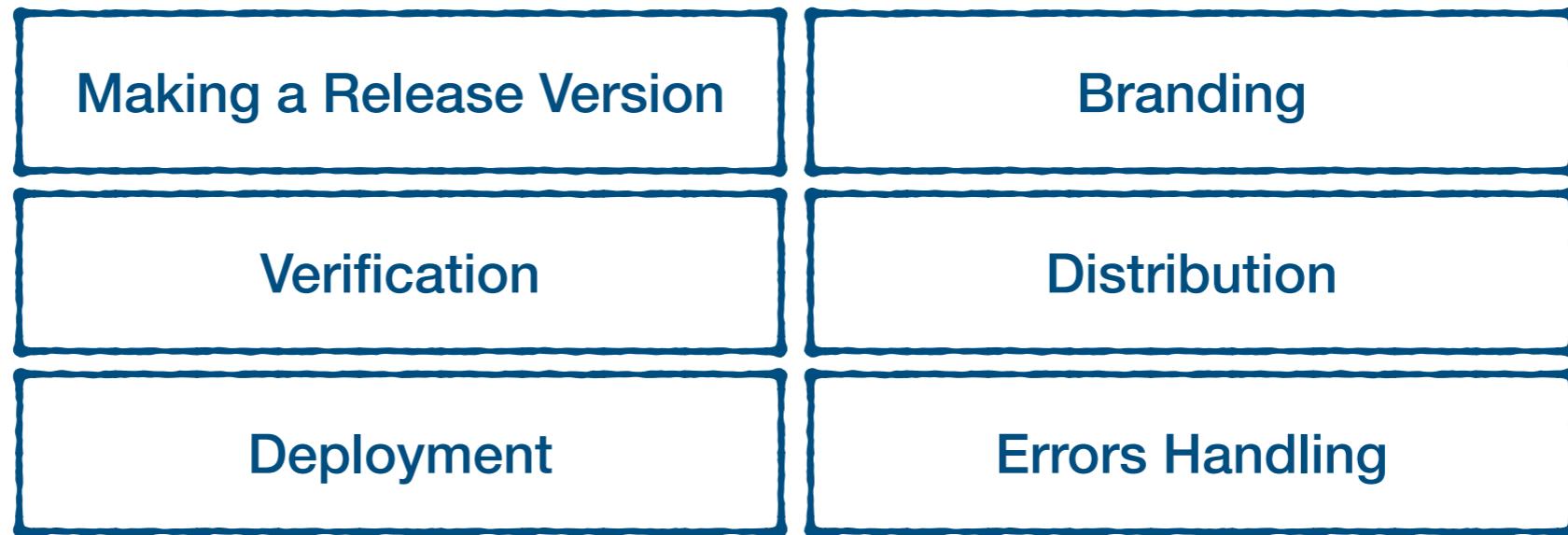
Operating System API

Integrated in
Pharo

<https://books.pharo.org/booklet-uffi/>



Application Development with Pharo



Pablo Tesone - Guille Polito
pablo.tesone@inria.fr
@tesonep