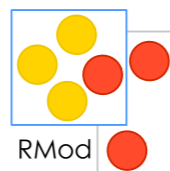




Application Development with Pharo

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

Inria



**Université
de Lille**

We want to develop in Pharo

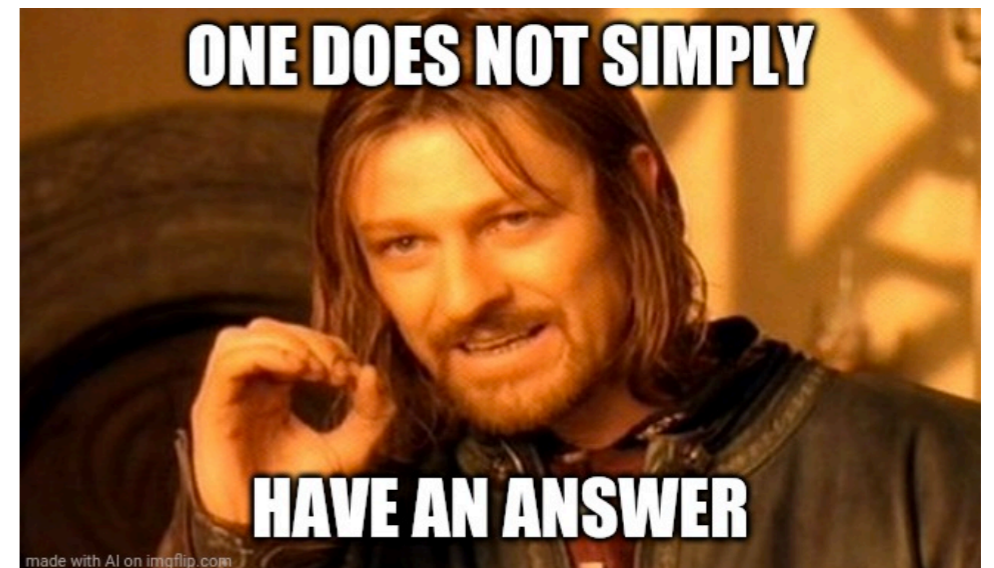
- Cool Tools
- Iterative Process
- Fun & Addictive



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



[astares/Seaside-Bootstrap5](https://github.com/astares/Seaside-Bootstrap5)



[ba-st/RenoirSt](https://github.com/ba-st/RenoirSt)

Phar 

pharojs.org



[ba-st/Willow](https://github.com/ba-st/Willow)

Web Applications



seaside 

www.seaside.st



[astares/Seaside-Bootstrap5](https://github.com/astares/Seaside-Bootstrap5)



[ba-st/RenoirSt](https://github.com/ba-st/RenoirSt)

Phar 

pharojs.org



[ba-st/Willow](https://github.com/ba-st/Willow)

Easy... we are cool here

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



**Native VM for RaspberryPi
ARM32 / ARM64**



pharo-iot/PharoThings



SquareBracketAssociates/Booklet-APharoThingsTutorial

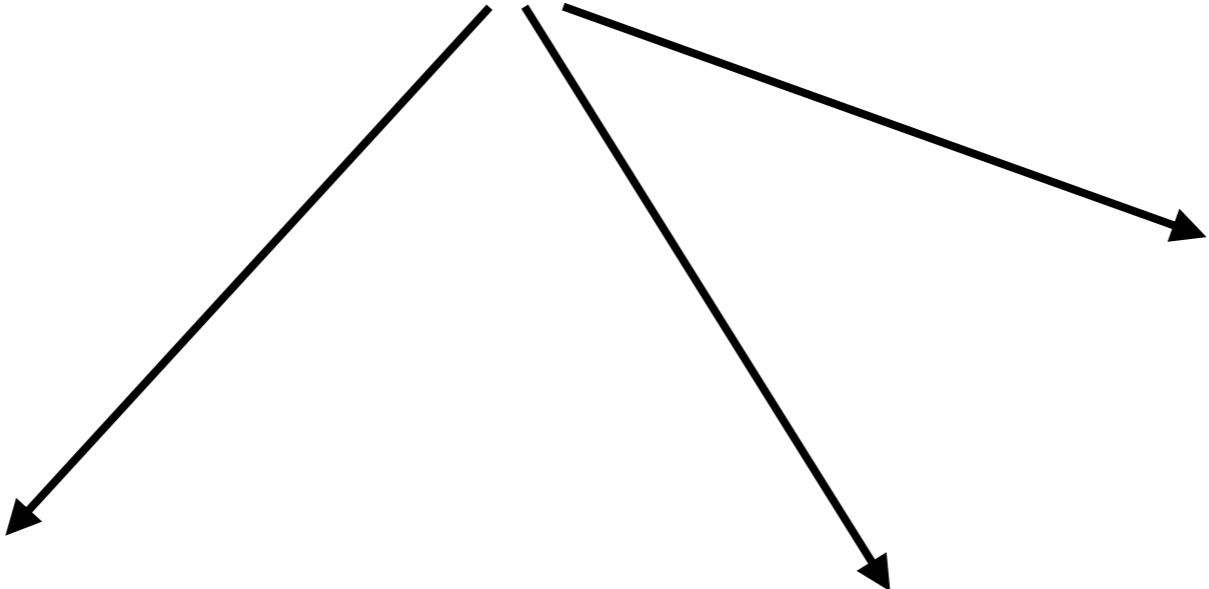
Desktop Applications



pharo-spec/Spec

Backends

Morphic



pharo-spec/Spec-Gtk

Integrated in Pharo



pharo-graphics/Spec-Brick

We used it daily

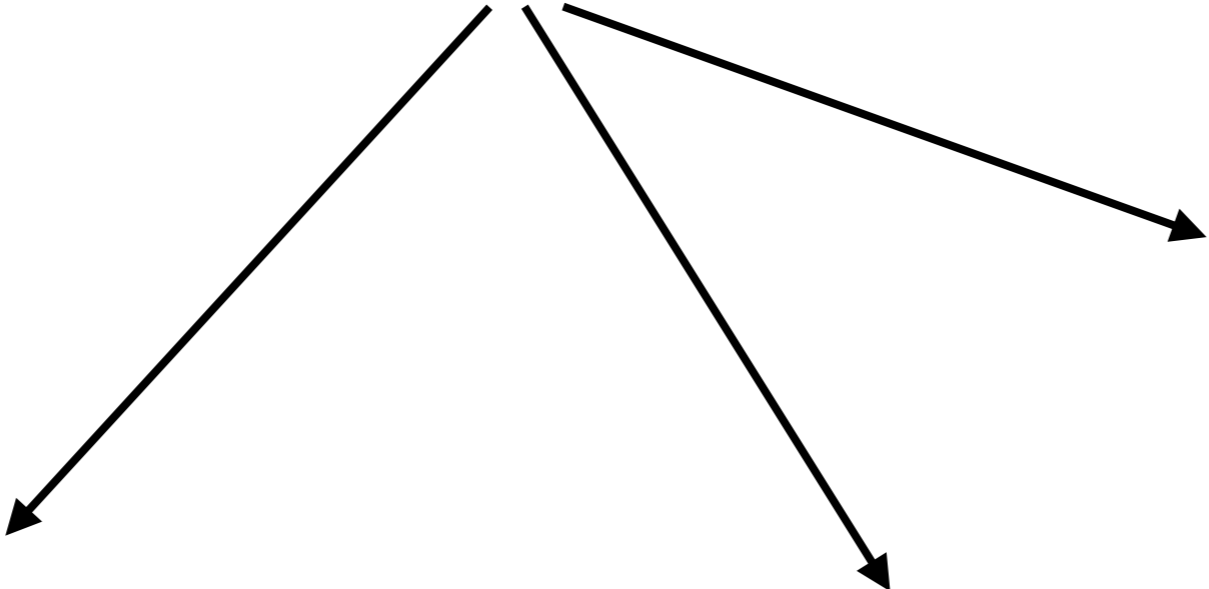
Desktop Applications



pharo-spec/Spec

Backends

Morphic



pharo-spec/Spec-Gtk

Without Morphic



pharo-graphics/Spec-Brick

Loadable in a minimal Image

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





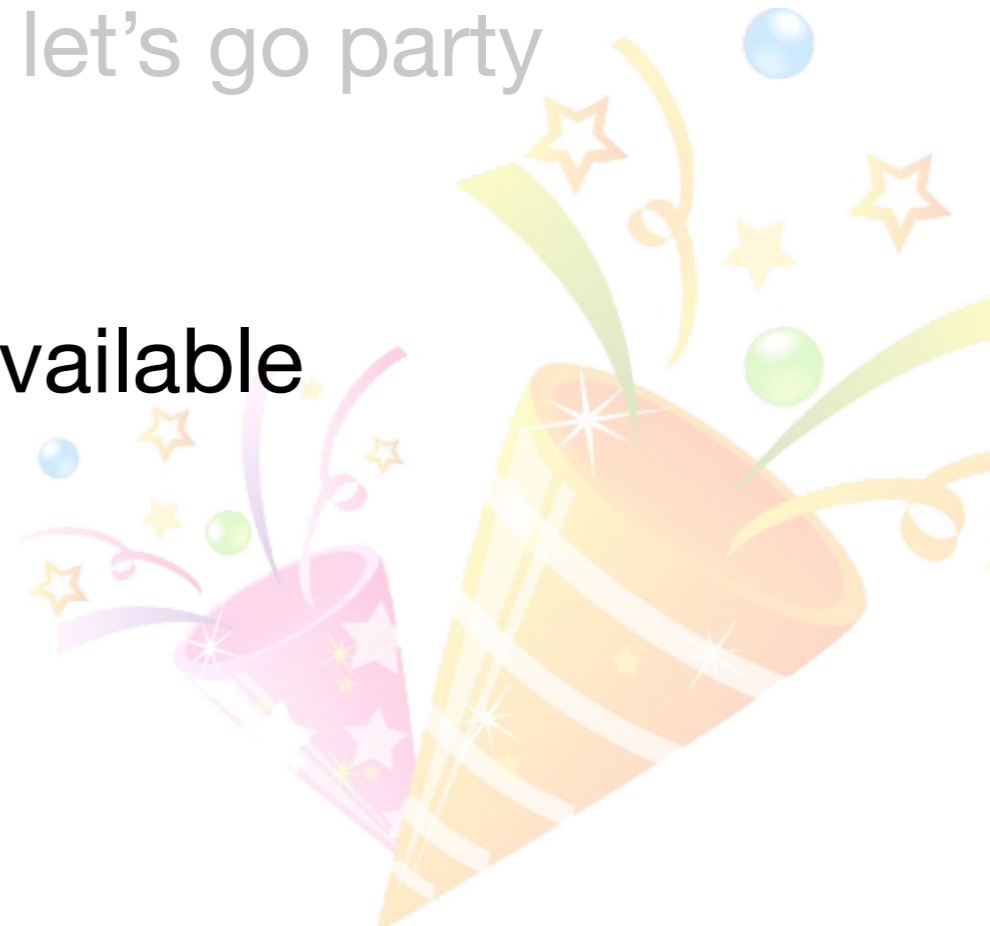
We have a solution developed...




...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

We will see these

- Deployment
- Errors Handling



Making a Release Version

Let's create an image



Making a Release Version

Let's create an image




Pharo
Image




My
AppCode

Do we want to
distribute the whole
Pharo Image?

Making a Release Version

Let's create an image

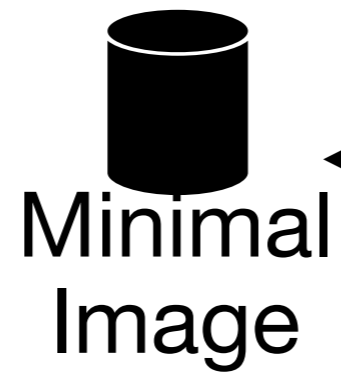


Do we want to
distribute the whole
Pharo Image?

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

Making a Release Version

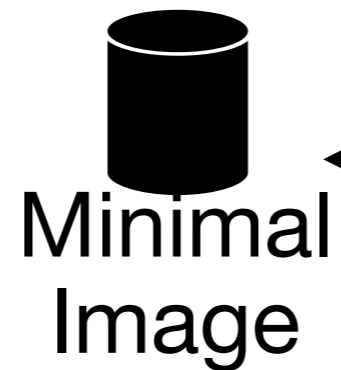
Let's create an image



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

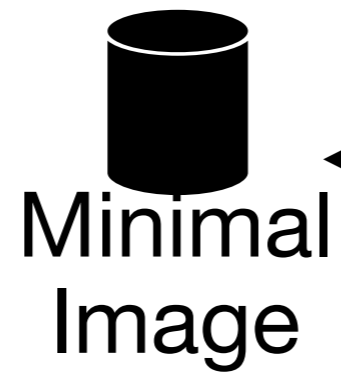
The minimal Image is generated during the bootstrap

Making a Release Version

Let's create an image



Ask If you are interested...



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

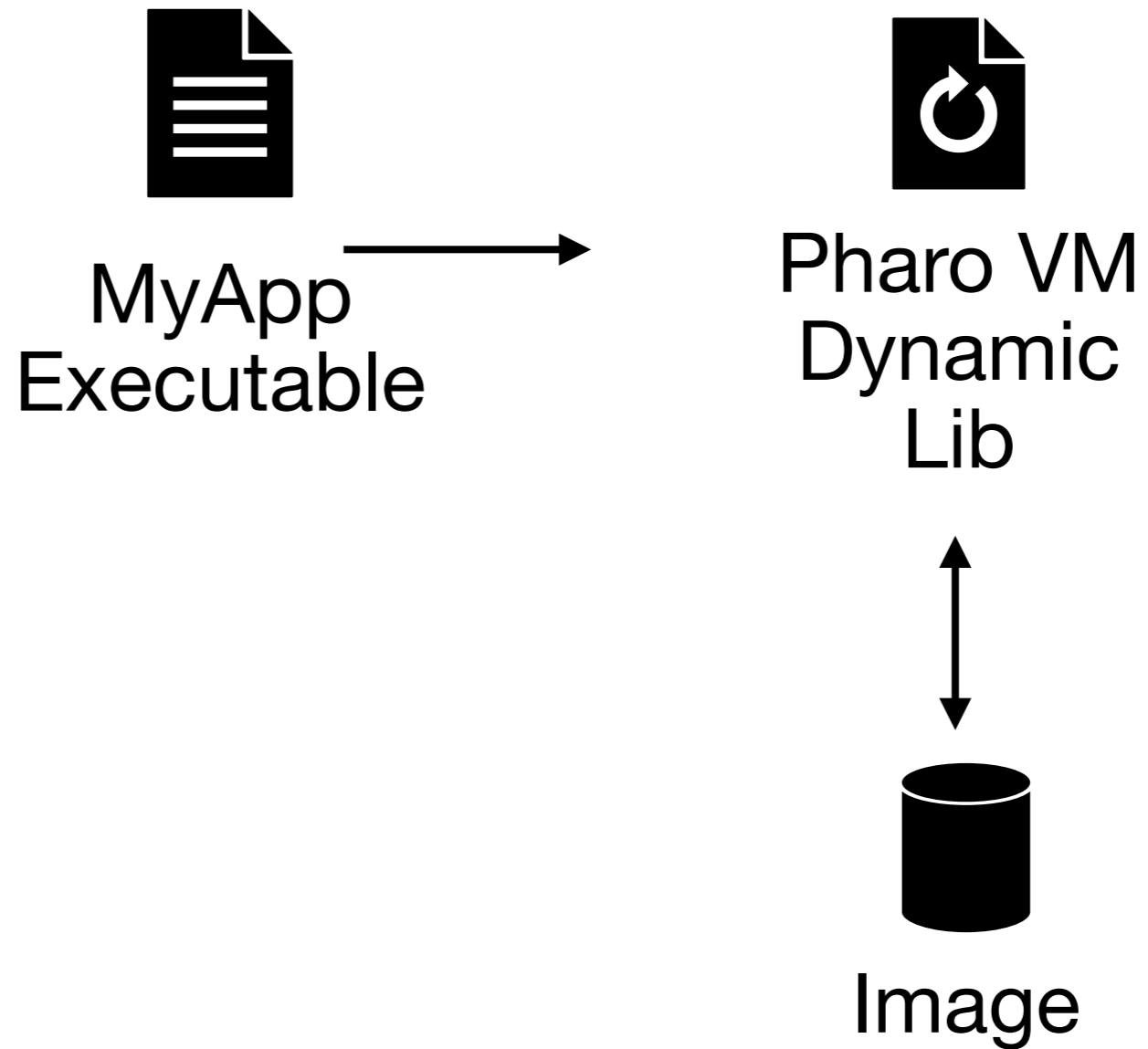
- My Own Icons
- My Own information
- Built using Pharo VM as a library



Branding

Proposed Architecture

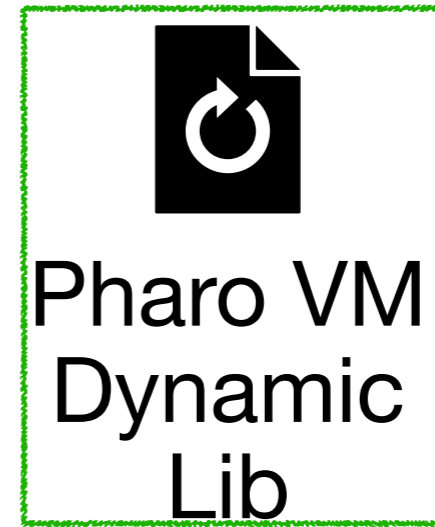
- Small



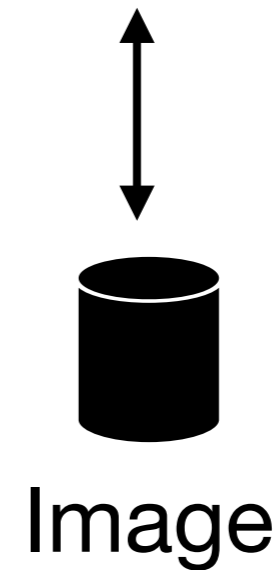
Branding

Proposed Architecture

- Small



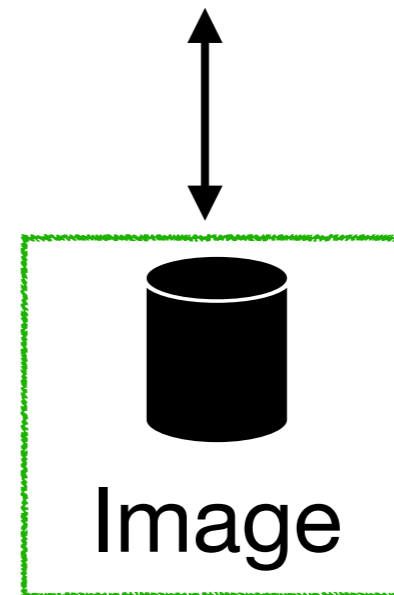
Built
by
Pharo



Branding

Proposed Architecture

- Small

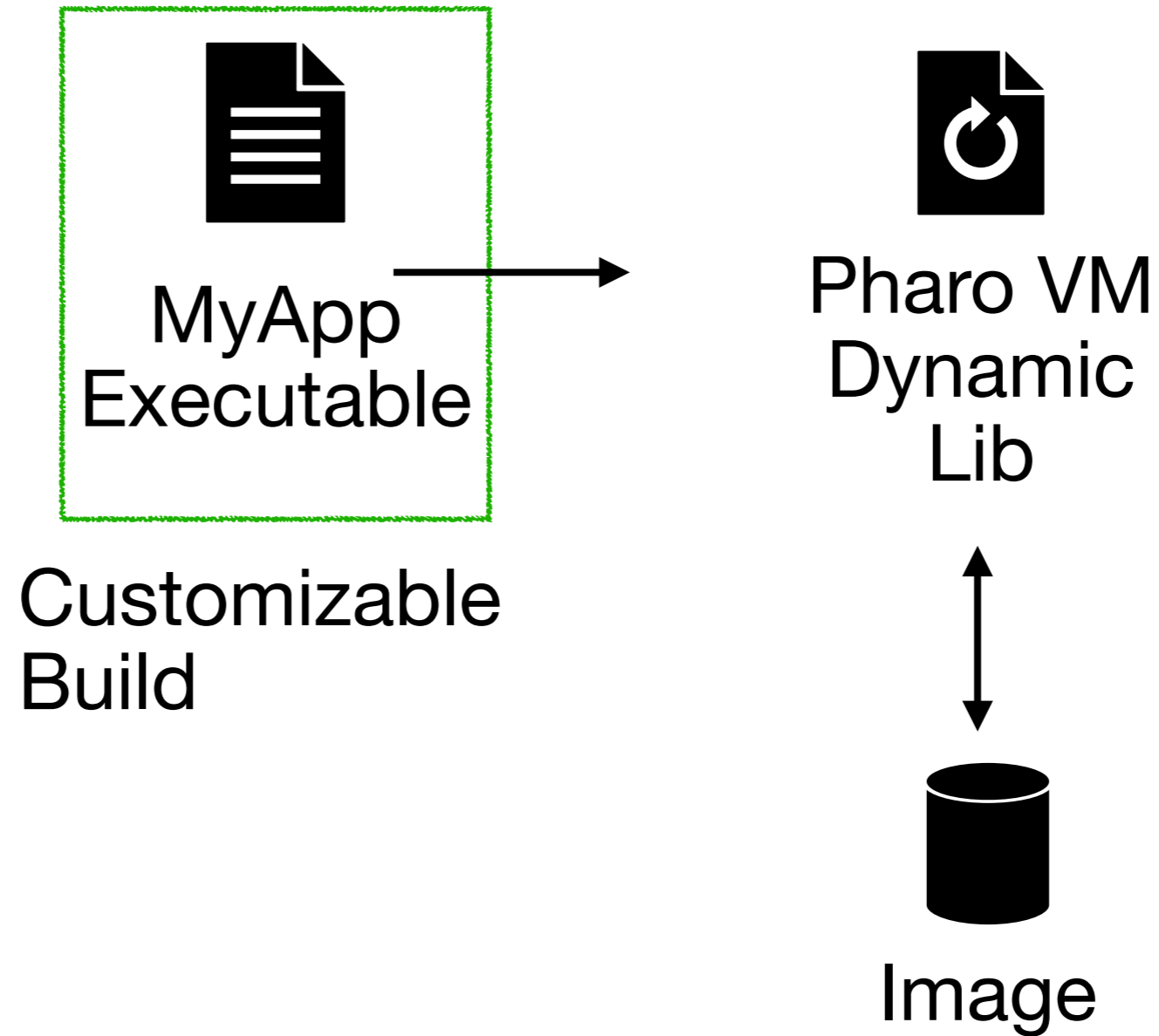


Generated as usual

Branding

Proposed Architecture

- Small

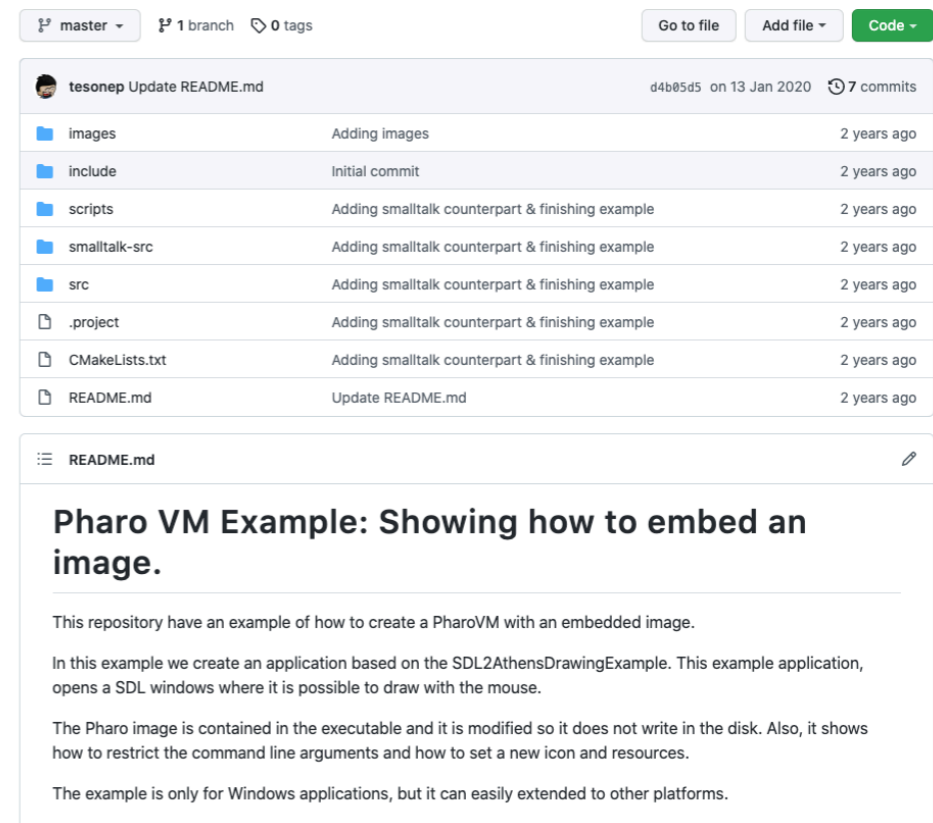


How to Implement it...

- 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
 */
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 func
 * 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

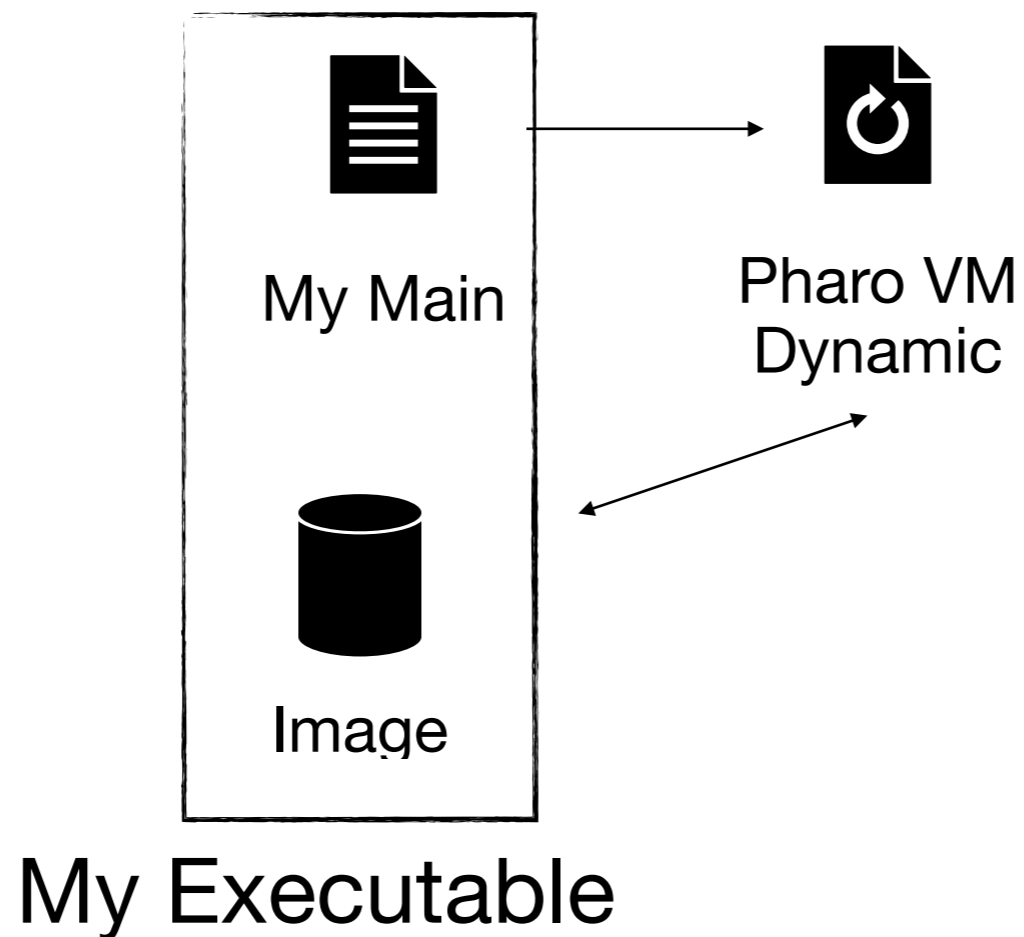
- Applications should be signed
- Signing should be done by the developer
- All executing code should be signed

We have to assure
that our applications
is not tainted

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

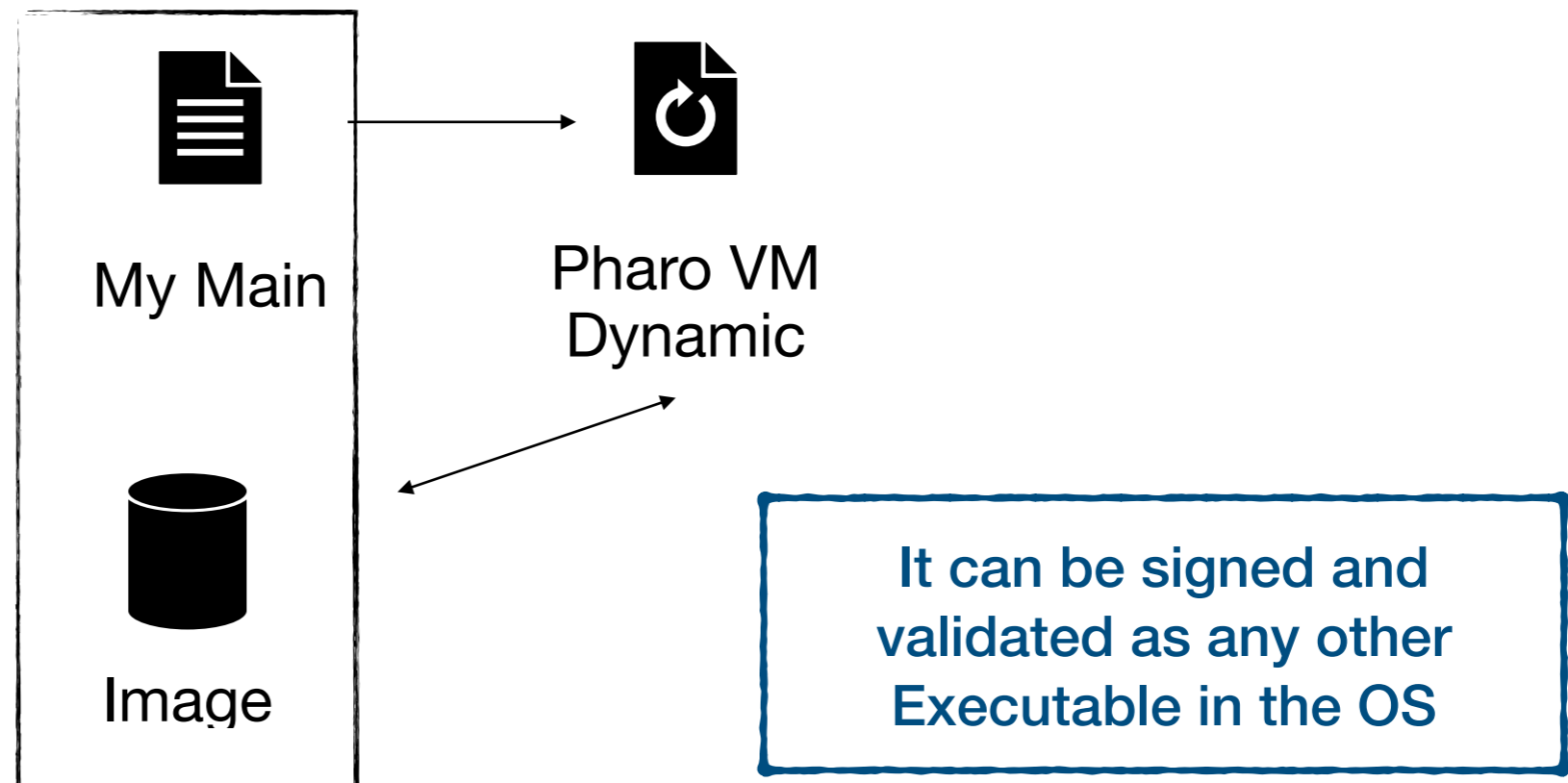
Alternative 1: Embedding as a Resource

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



Alternative 1: Embedding as a Resource

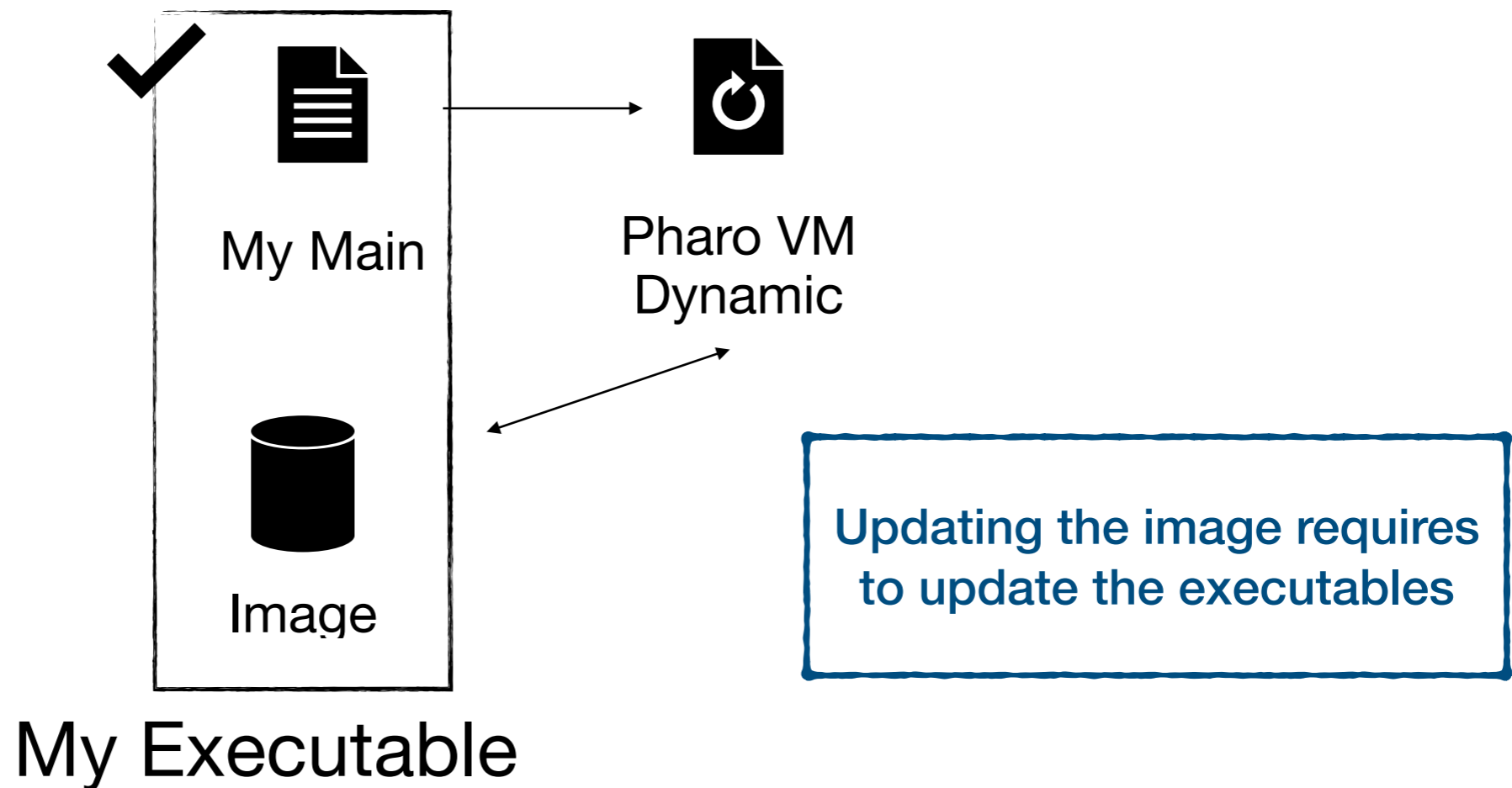
- If the image not change we can embed it as a resource.



My Executable

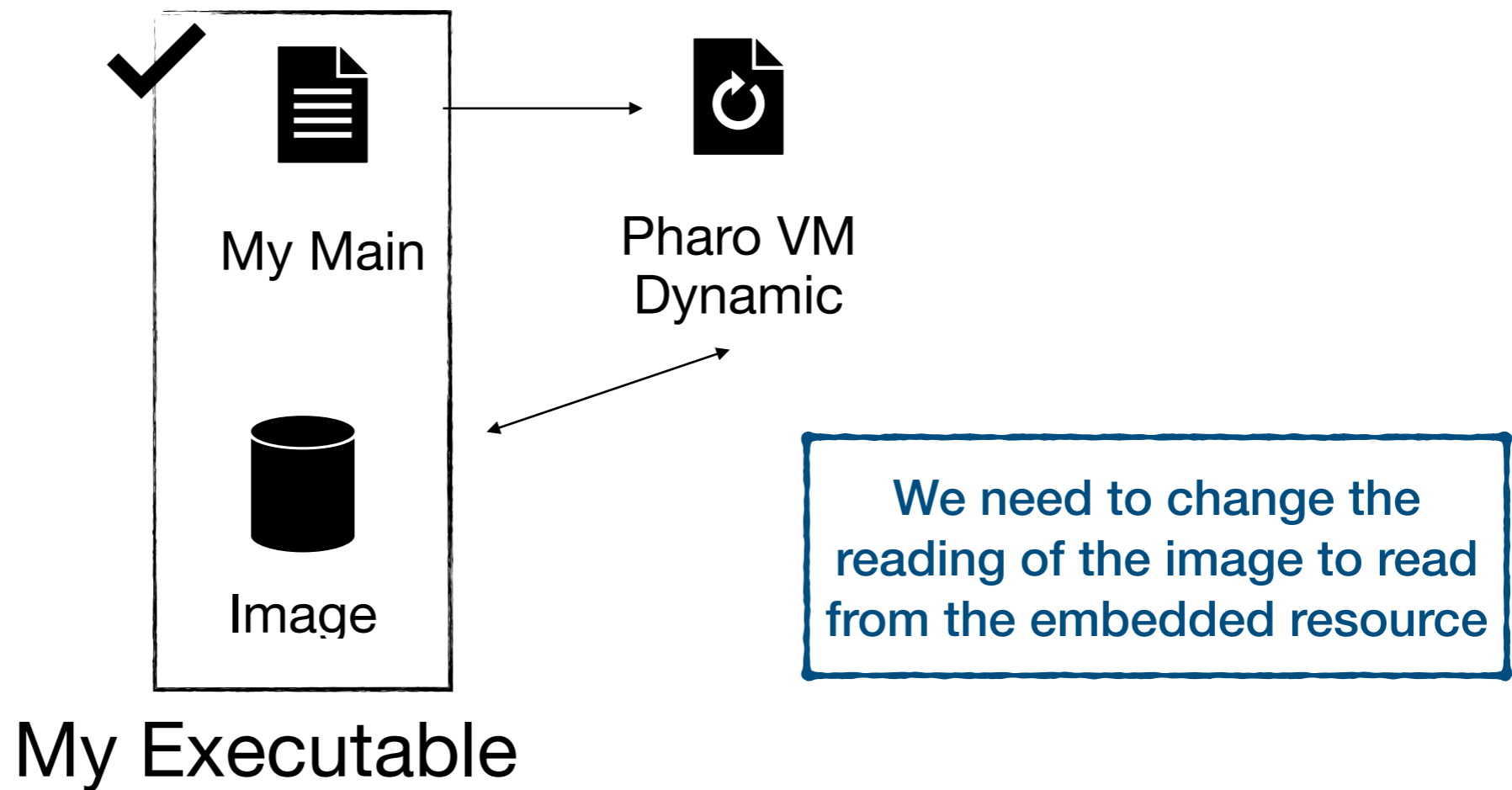
Alternative 1: Embedding as a Resource

- 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

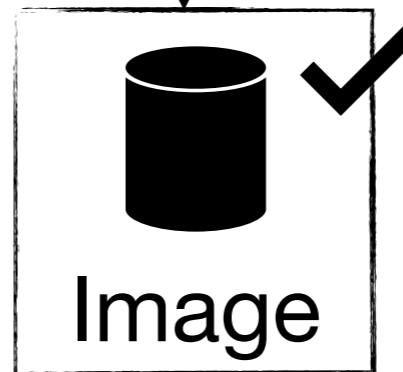
Proposed Architecture



MyApp
Executable



Pharo VM
Dynamic
Lib



Image

We can update the Image
downloading a new one

Alternative 2: We sign it outside the executable

Proposed Architecture

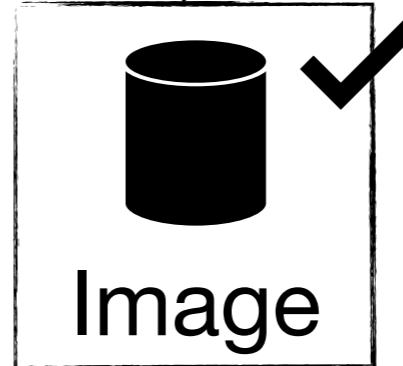


MyApp
Executable



Pharo VM
Dynamic
Lib

We need to update the read
to check the image
signature...



Image

Alternative 2: We sign it outside the executable

Proposed Architecture

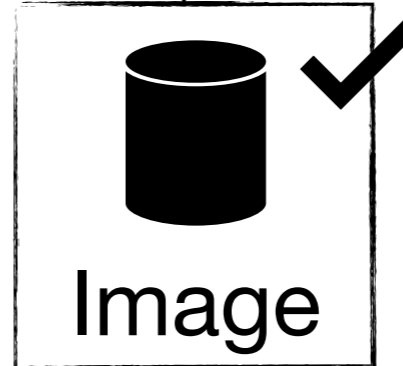


MyApp
Executable



Pharo VM
Dynamic
Lib

It is not validated by the OS... we need to validate it on loading



Image

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) embeddedFileAccess = {
    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(&embeddedFileAccess);
```

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;

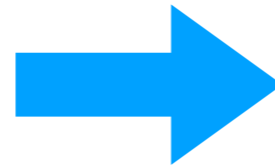
typedef _FileAccessHandler FileAccessHandler;
```

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(&embeddedFileAccess);
```


Distribution / Installation

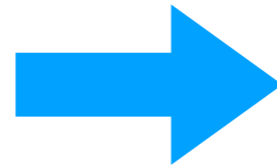
- Windows
- OSX



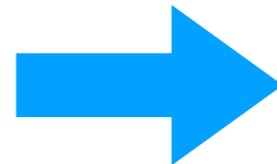
We have a normal application, we can distribute as usual

Distribution / Installation

- 
- Windows
 - OSX
 - Linux



We have a normal application, we can distribute as usual



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
Arch <ul style="list-style-type: none">⚠ aarch64🚚 x86_64 🔍 succeeded: 2 🔍 disabled: 3	xUbuntu_20.10 <ul style="list-style-type: none">🚚 aarch64 🔍 succeeded: 4 🔍 disabled: 1🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	Raspbian_11 <ul style="list-style-type: none">🚚 armv7l 🔍 disabled: 5	Fedora_35 <ul style="list-style-type: none">🚚 aarch64 🔍 disabled: 5🚚 armv7l 🔍 disabled: 5🚚 ppc64le 🔍 disabled: 5🚚 <u>x86_64</u> 🔍 succeeded: 3 🔍 disabled: 2	Debian_Testing <ul style="list-style-type: none">🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1
Debian_10 <ul style="list-style-type: none">🚚 i586 🔍 disabled: 2 🔍 excluded: 3🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	xUbuntu_21.04 <ul style="list-style-type: none">🚚 aarch64 🔍 succeeded: 4 🔍 disabled: 1🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	Raspbian_9.0 <ul style="list-style-type: none">🚚 aarch64 🔍 succeeded: 3 🔍 failed: 1 🔍 disabled: 1🚚 armv7l 🔍 succeeded: 3 🔍 failed: 1 🔍 disabled: 1	Fedora_32 <ul style="list-style-type: none">🚚 aarch64 🔍 disabled: 5🚚 armv7l 🔍 disabled: 5🚚 ppc64le 🔍 disabled: 5🚚 x86_64 🔍 succeeded: 3 🔍 disabled: 2	Fedora_36 <ul style="list-style-type: none">🚚 aarch64 🔍 disabled: 5🚚 armv7l 🔍 disabled: 5🚚 ppc64le 🔍 disabled: 5🚚 x86_64 🔍 succeeded: 2 🔍 disabled: 3
Debian_11 <ul style="list-style-type: none">🚚 i586 🔍 disabled: 5🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	xUbuntu_21.10 <ul style="list-style-type: none">🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	openSUSE_Leap_15.2 <ul style="list-style-type: none">🚚 <u>x86_64</u> 🔍 succeeded: 4 🔍 disabled: 1	Fedora_33 <ul style="list-style-type: none">🚚 aarch64 🔍 disabled: 5🚚 armv7l 🔍 disabled: 5🚚 ppc64le 🔍 disabled: 5🚚 x86_64 🔍 succeeded: 3 🔍 disabled: 2	Fedora_34 <ul style="list-style-type: none">🚚 aarch64 🔍 disabled: 5🚚 armv7l 🔍 disabled: 5
Debian_9.0 <ul style="list-style-type: none">🚚 i586 🔍 disabled: 2 🔍 excluded: 3🚚 x86_64 🔍 succeeded: 3	xUbuntu_22.04 <ul style="list-style-type: none">🚚 x86_64 🔍 succeeded: 2 🔍 disabled: 3	openSUSE_Leap_15.3 <ul style="list-style-type: none">🚚 x86_64 🔍 succeeded: 4 🔍 disabled: 1	Raspbian_10 <ul style="list-style-type: none">🚚 aarch64 🔍 succeeded: 4 🔍 disabled: 1🚚 armv7l 🔍 succeeded: 4 🔍 disabled: 1	
		openSUSE_Tumbleweed <ul style="list-style-type: none">🚚 i586 🔍 disabled: 5🚚 x86_64 🔍 succeeded: 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
Arch	xUbuntu_20.10	Raspbian_11	Fedora_35	Debian_Testing
aarch64	aarch64 question mark succeeded: 4	armv7l question mark disabled: 5	aarch64 question mark disabled: 5	x86_64 question mark succeeded: 4
x86_64 question mark succeeded: 2	question mark disabled: 1	Raspbian_9.0	armv7l question mark disabled: 5	question mark disabled: 1
question mark disabled: 3	x86_64 question mark succeeded: 4	aarch64 question mark succeeded: 3	ppc64le question mark disabled: 5	Fedora_32
Debian_10	question mark disabled: 1	question mark failed: 1	<u>x86_64</u> question mark succeeded: 3	aarch64 question mark disabled: 5
i586 question mark disabled: 2	xUbuntu_21.04	question mark disabled: 1	question mark disabled: 2	armv7l question mark disabled: 5
question mark excluded: 3	aarch64 question mark succeeded: 4	armv7l question mark succeeded: 3	Fedora_36	ppc64le question mark disabled: 5
x86_64 question mark succeeded: 4	question mark disabled: 1	question mark failed: 1	aarch64 question mark disabled: 5	x86_64 question mark succeeded: 3
question mark disabled: 1	x86_64 question mark succeeded: 4	question mark disabled: 1	armv7l question mark disabled: 5	question mark disabled: 2
Debian_11	question mark disabled: 1	openSUSE_Leap_15.2	ppc64le question mark disabled: 5	Fedora_33
i586 question mark disabled: 5	xUbuntu_21.10	<u>x86_64</u> question mark succeeded: 4	x86_64 question mark succeeded: 2	aarch64 question mark disabled: 5
x86_64 question mark succeeded: 4	x86_64 question mark succeeded: 4	question mark disabled: 1	question mark disabled: 3	armv7l question mark disabled: 5
question mark disabled: 1	question mark disabled: 1	openSUSE_Leap_15.3	Raspbian_10	ppc64le question mark disabled: 5
Debian_9.0	xUbuntu_22.04	x86_64 question mark succeeded: 4	aarch64 question mark succeeded: 4	x86_64 question mark succeeded: 3
i586 question mark disabled: 2	x86_64 question mark succeeded: 2	question mark disabled: 1	question mark disabled: 1	question mark disabled: 2
question mark excluded: 3	question mark disabled: 3	openSUSE_Tumbleweed	armv7l question mark succeeded: 4	Fedora_34
x86_64 question mark succeeded: 3		i586 question mark disabled: 5	question mark disabled: 1	aarch64 question mark disabled: 5
		x86_64 question mark succeeded: 2	armv7l question mark succeeded: 4	armv7l question mark 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

For Non-OpenSource easy to
run in-house

The packages can be
integrated in default
repositories

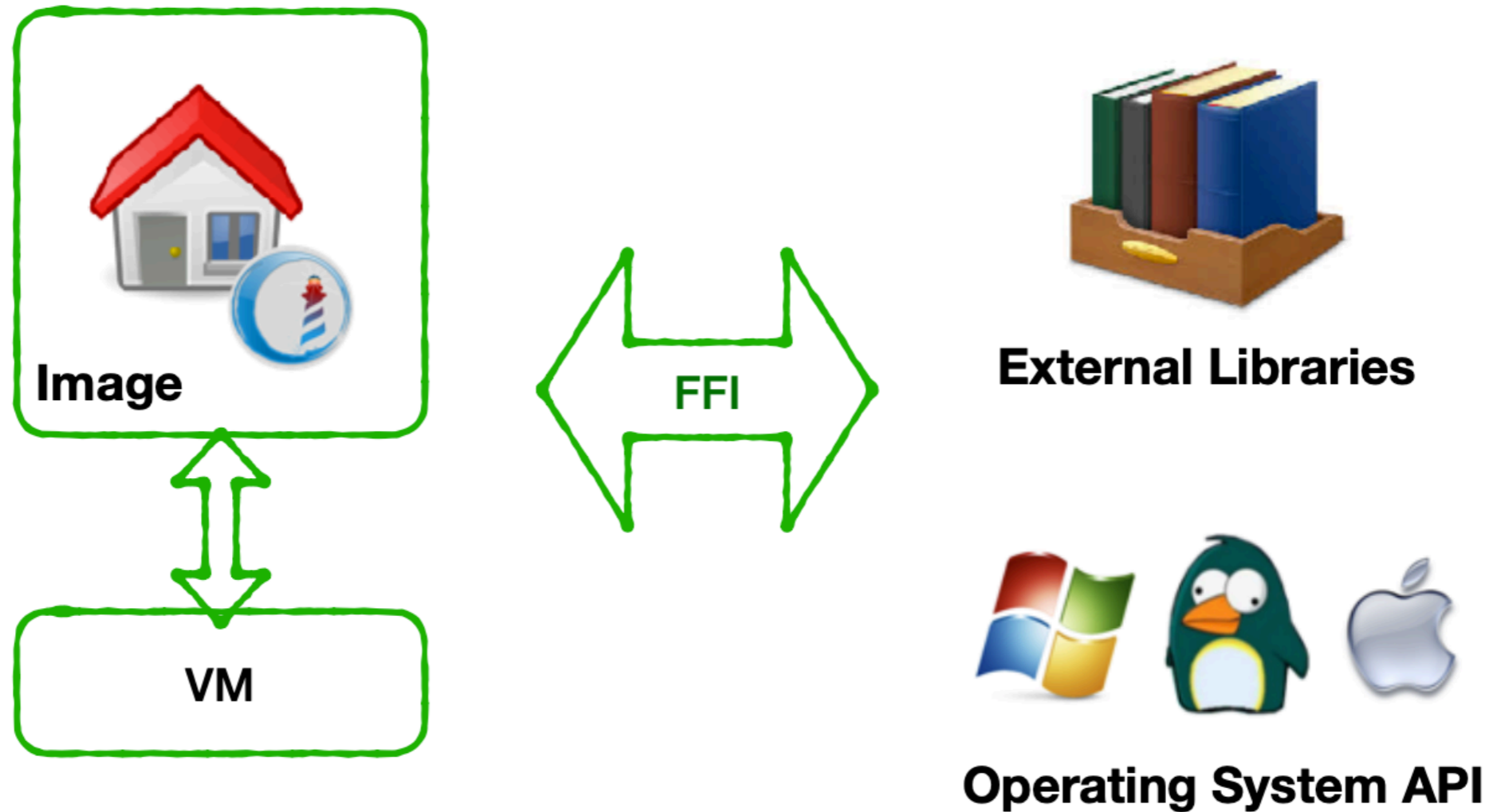
It provides a native repository
supporting updates



Bonus: Remember FFI

FFI: Opening the game to external libraries

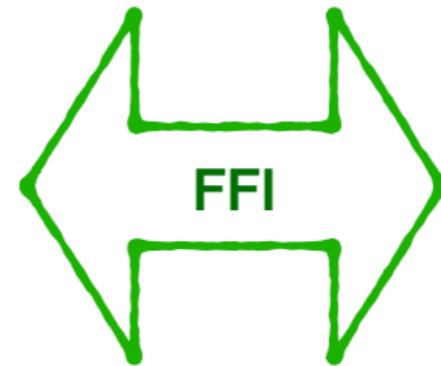
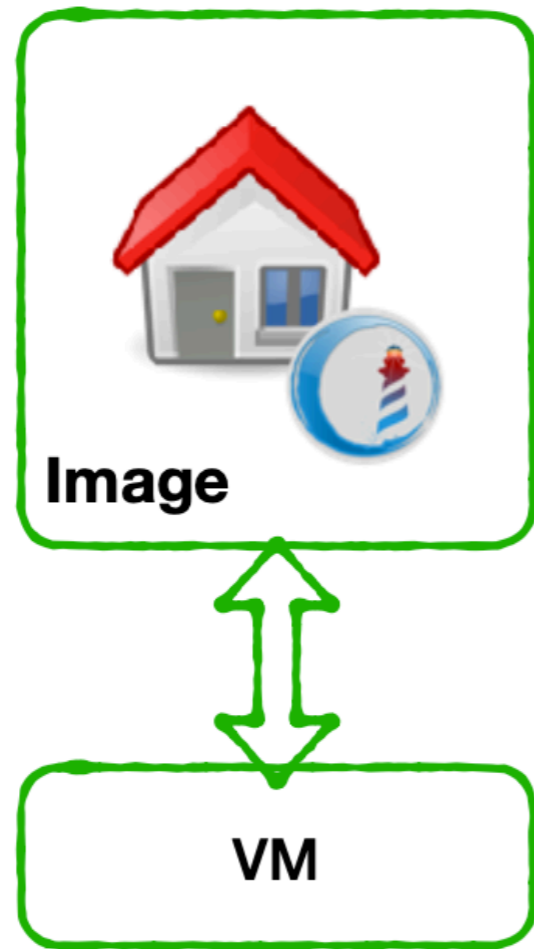
Let's not reinvent the wheel



We can communicate with anything that has a C API

FFI: Opening the game to external libraries

Let's not reinvent the wheel



External Libraries



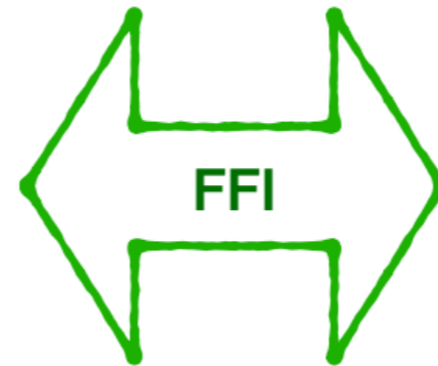
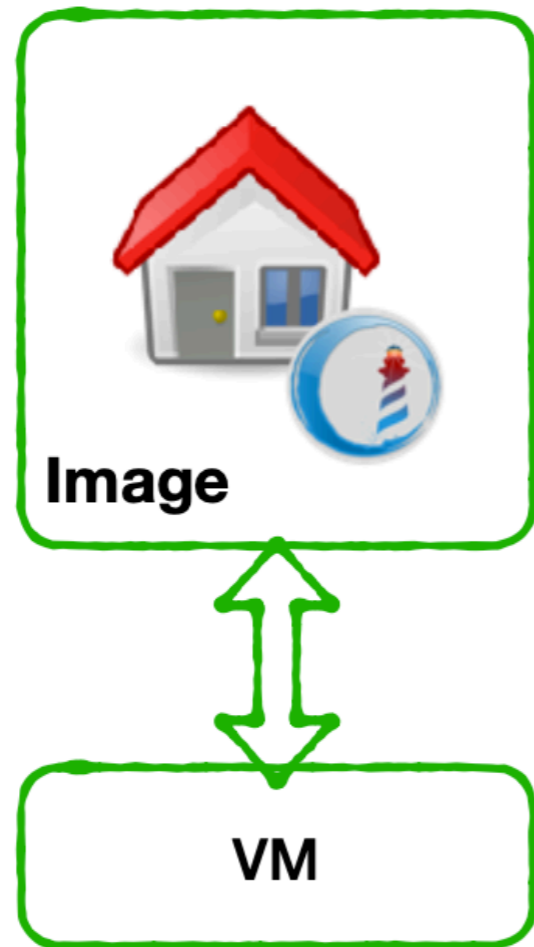
Operating System API

Reusing Libraries

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

Making a Release Version

Branding

Verification

Distribution

Deployment

Errors Handling

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