Hands-On Angular Tooling

Pôle Java
Demi journée technique

Hello, we are valtech_

Nous créons de la **Valeur** par la **Technologie**.

valtech_

Objectifs

- Découvrir les outils utilisés en Javascript et AngularJS
  - node et NPM
  - Les frameworks de test (karma, jasmine, protractor)
  - bower
  - grunt / gulp
  - yeoman
- Faire une application
- Déployer l'application dans un serveur NGinx sous docker

Agenda

- La plate-forme Java
- L'écosystème

valtech.

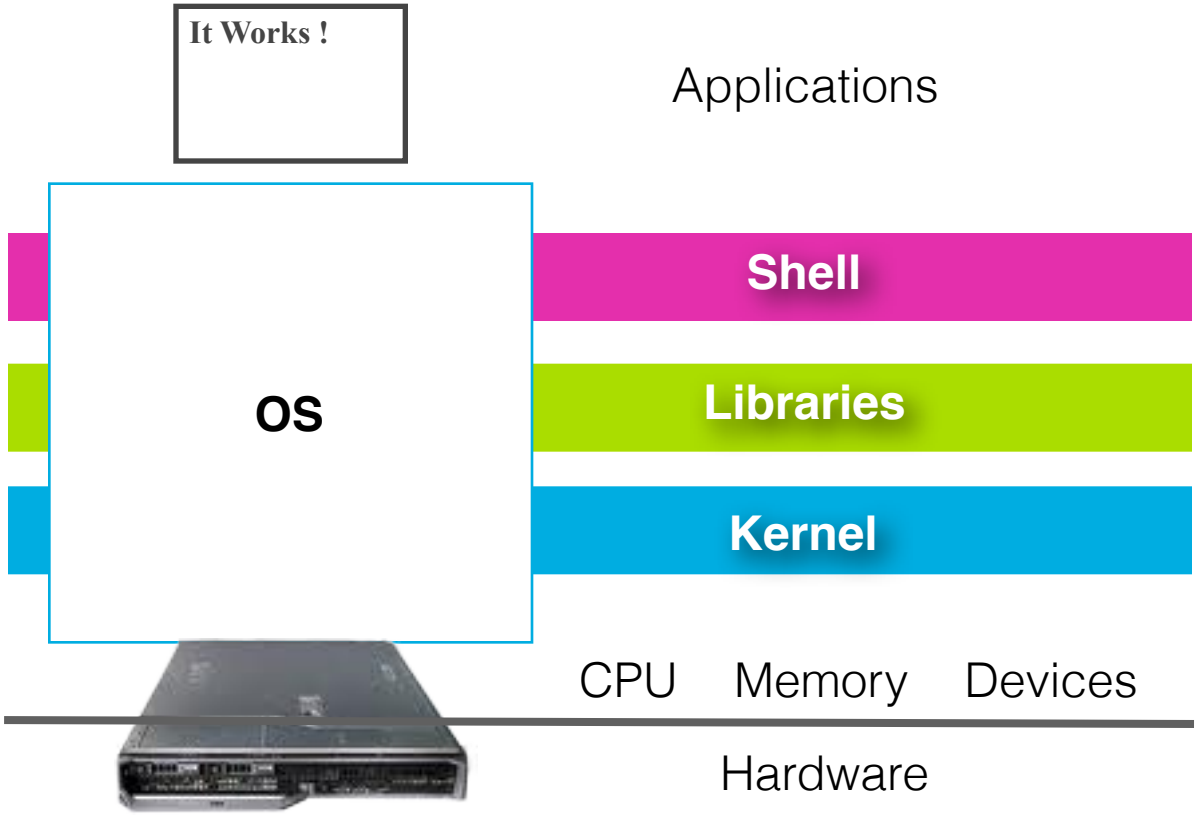
La VM

valtech.

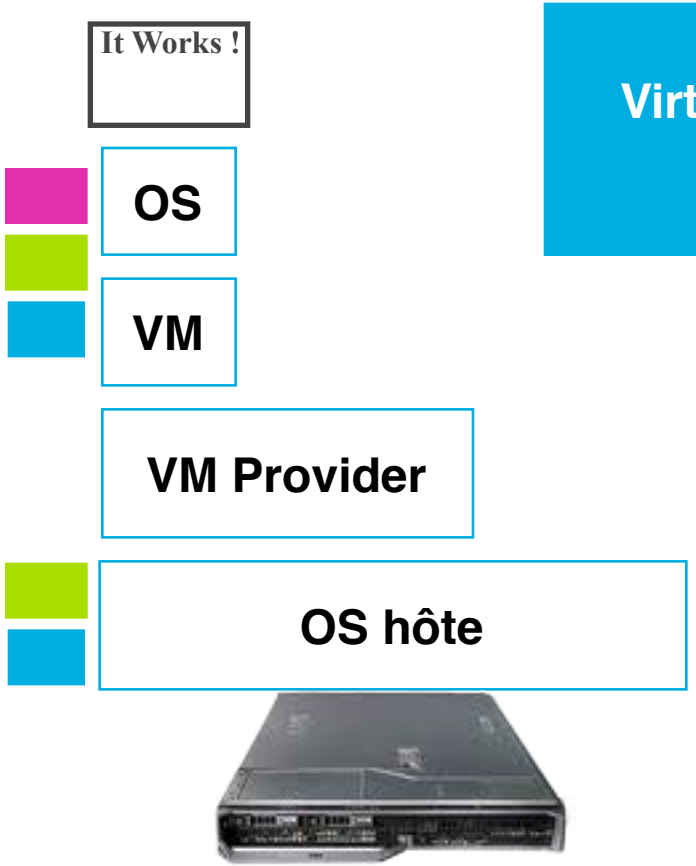
Vagrant et VirtualBox

- VirtualBox est un provider de VM (Virtual Machine)
- Vagrant est un outil de gestion de VM en ligne de commande

Les couches d'un OS



Les types de virtualisation hardware

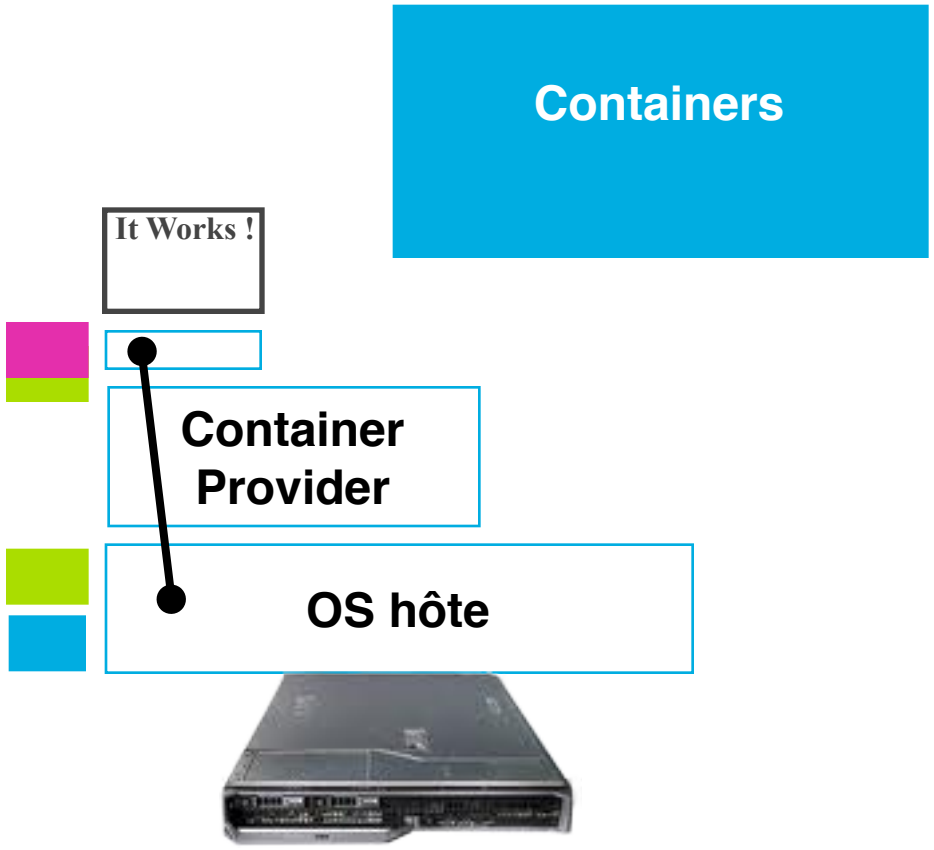


Virtual Machines (VM)

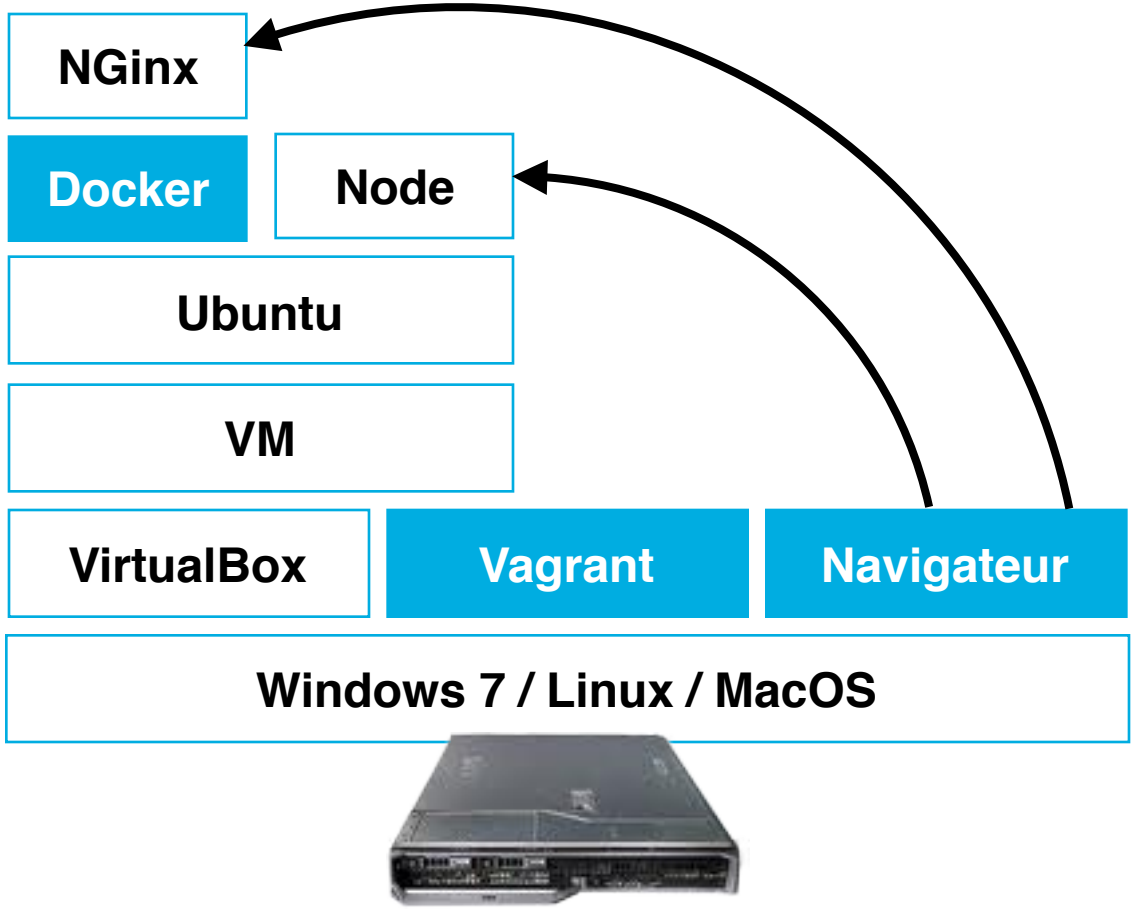
- Logiciel permettant d'émuler une machine sur un OS
- Le guest a un OS complet qui peut être différent de celui de l'hôte
- VMWare, VirtualBox

Les types de virtualisation hardware

- Logiciel permettant de compartimenter un OS pour qu'il apparaisse comme plusieurs instances
- Utilise le même kernel. Le guest n'a qu'une couche d'adaptation et les OS doivent être compatibles
- LXC, WPar Aix; Jail FreeBSD



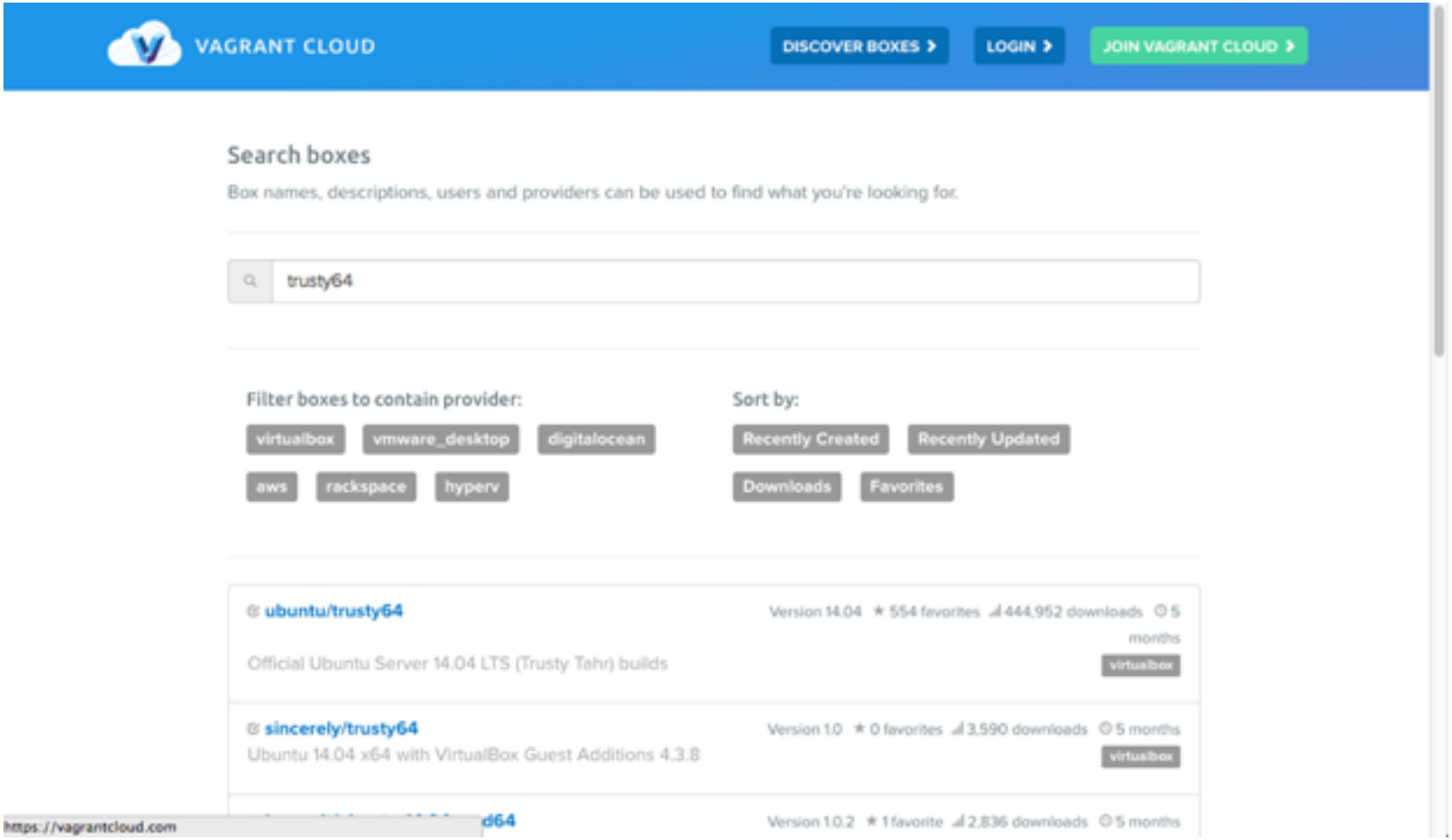
L’architecture pour ce cours



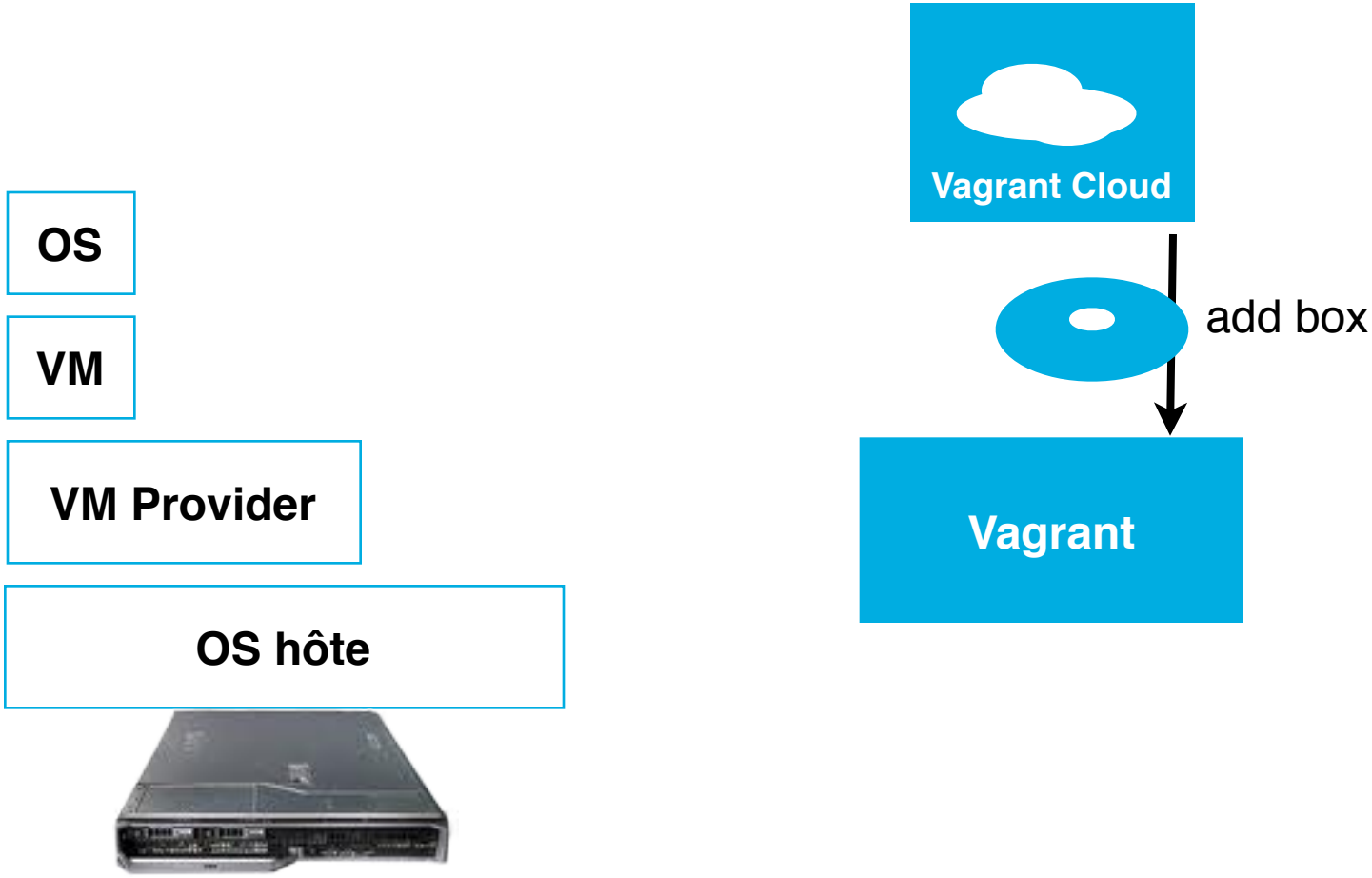
VMs versus Containers

- VMs
  - Pas de contraintes sur les OS
  - Tester des opérations système sur un OS vierge
  - Tester des logiciels sur des OS diffférents
- Containers
  - Les OS doivent être compatibles
  - Moins consommateur de ressources
  - Mêmes usages que la VM si OS compatibles
  - Possibilité d'émuler une plate-forme distribuée

Vagrant Cloud



Vagrant - obtenir l'image de la VM



Vagrantfile

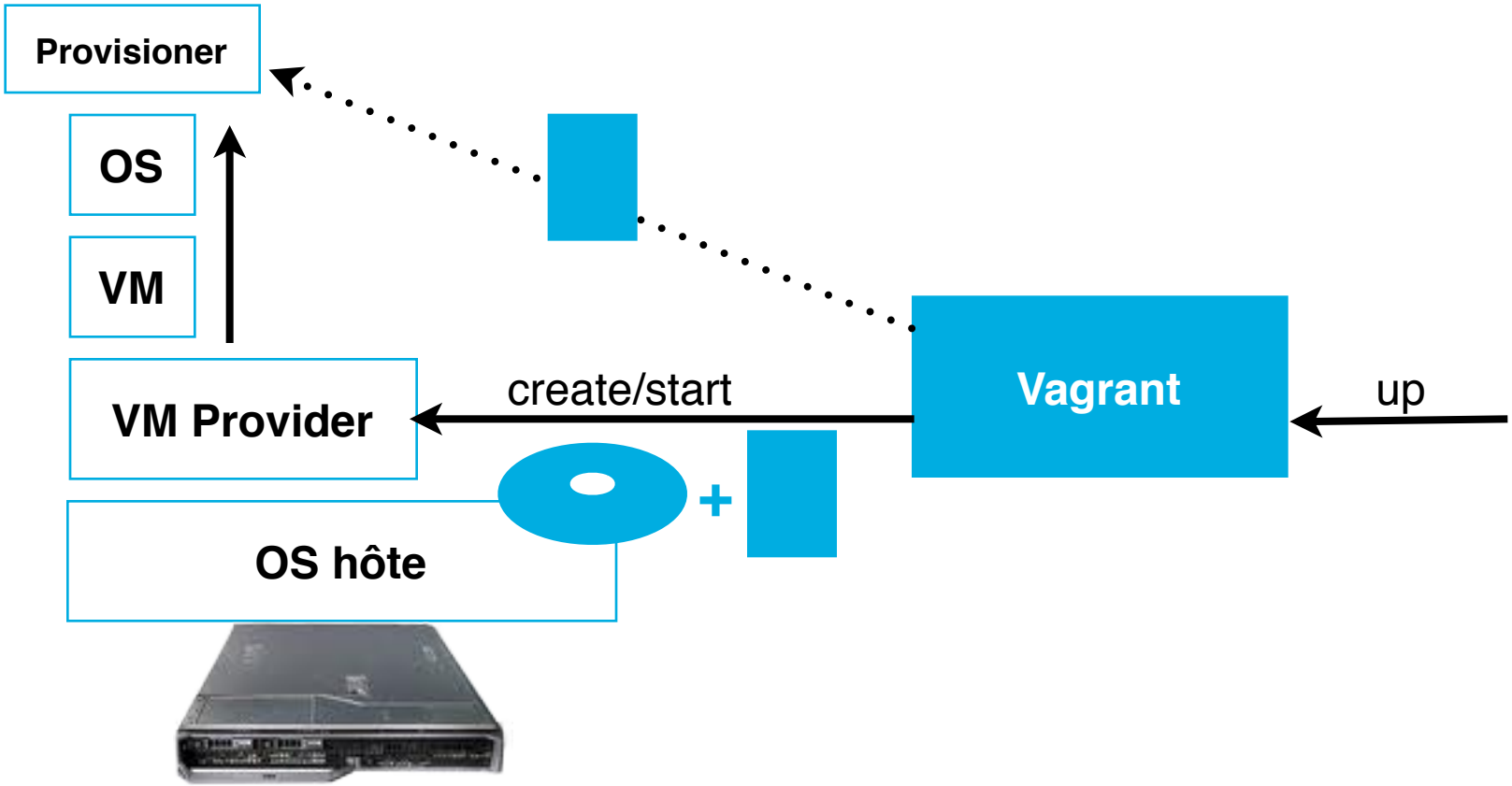

```
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  config.vm.box = "ubuntu/trusty64"

  # forward http-server port to guest
  config.vm.network "forwarded_port", guest: 8000, host: 8000

  config.vm.provider "virtualbox" do |v|
    v.memory = 1024
  end

  # run the installation script
  config.vm.provision "shell", path: "./install-env.bash"

end
```

valtech.

Vagrant - Principales commandes

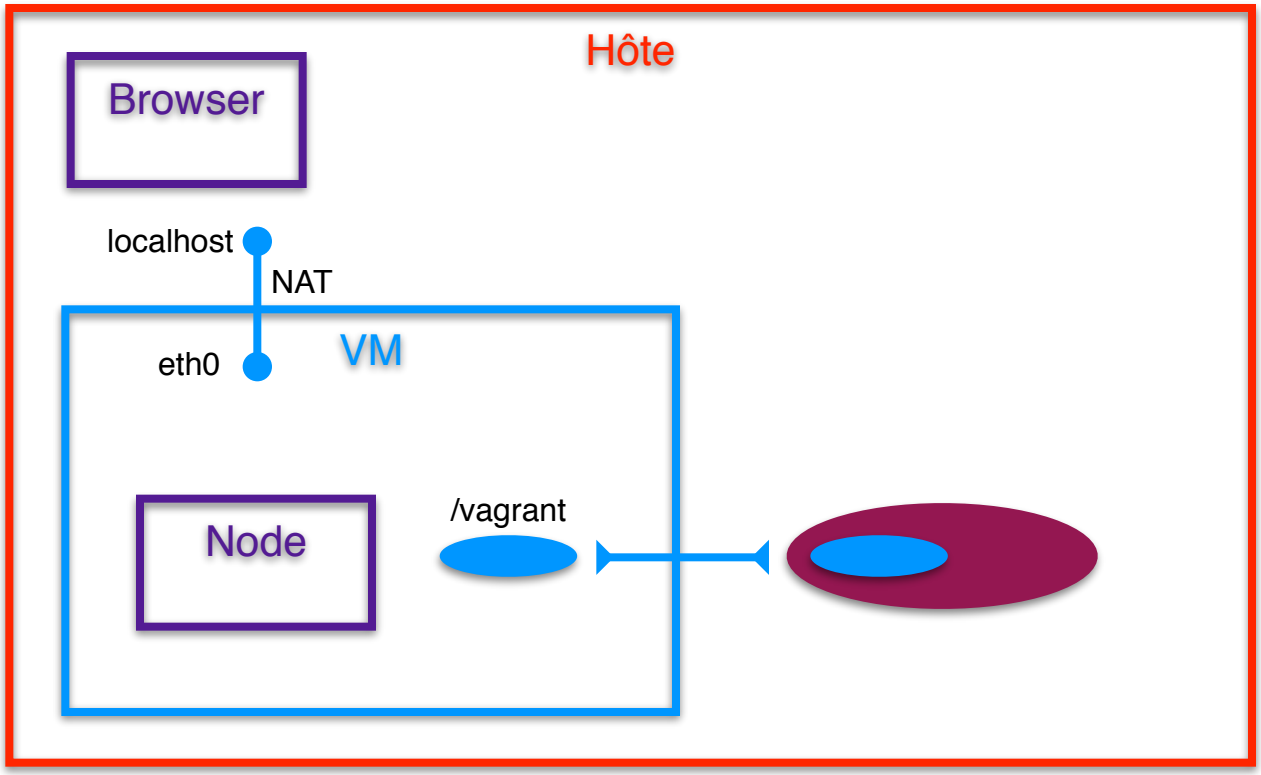
- vagrant add box <url de box>
  - ajoute la box au catalogue
- vagrant init <nom de box>
  - génère le vagrantfile
  - le Vagrantfile identifie la VM dans toutes les commandes
- vagrant up
  - construit la VM au premier run
  - démarre la VM
- vagrant ssh
  - se connecter sur la VM
- vagrant halt
- vagrant destroy

Vagrant - création de la VM



Communication avec la VM

- NAT et forward de port
- Montage de disque virtuel



Lab Vagrant

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab1

- Lab Vagrant
  - Démarrer la VM

Node JS

valtech.

Node

- **Node.js** est un **environnement d'exécution** en **Javascript**
- Site de référence : http://nodejs.org

- Utilisation interactive

```
$ node
> var name = 'Alice';
undefined
> console.log(name);
Alice
undefined
>
(^C again to quit)
>
```

Un programme Node

- Un exemple de programme

hello.js

```
var name = "Alice";

console.log( "Hello " + name);
```

- Lancer le programme

```
$ node hello.js
Hello Alice
```

Node est asynchrone

- Chaque fonction est **asynchrone**. Tout ce qui bloquerait le thread doit être exécuté en arrière-plan

- Par exemple, la lecture d'un fichier est asynchrone. Il faut passer une fonction qui sera exécutée lorsque les données seront disponibles (**callback**)

reader.js

```
var fs = require("fs");
fs.readFile("test.txt", 'utf-8', function (error, data) {
    console.log( "File content: " + data);
});
```

```
$ node reader.js
Would you tell me, please, which way I ought to go from here?
```

valtech.

Les modules Node

- Node utilise une architecture en **modules**
- Le module n'expose que certaines fonctions. Les fonctions non exposées sont en accès privé dans le module

- Un exemple de module

formula.js | circle.js

```
var PI = Math.PI;

module.exports.circumference = function (r) {
  return 2 * PI * r;
};
```

Pi est-il utilisable dans circle.js ?

formula.js | circle.js

```
var formula = require('./formula.js');
console.log( "Circle circumference: " + formula.circumference(10));
```

valtech.

Installer node et les composants

- Installer l'environnement
  - apt-get install nodejs
  - apt-get install npm
- Installer un module via le **package manager NPM**
  - npm install module_name
- Les **packages** sont des modules publiés sur NPM

valtech.

Lab Node et NPM

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab2

- Lab Node
  - Lancer node en interactif et taper quelques commandes
  - Ecrire un programme et le lancer avec Node
  - Ecrire un module l'appeler

NPM

NPM

- **NPM** est le package manager de **Node**
  - fabriquer le package
  - publier le package
  - installer le package
  - utiliser l'application correspondant au package

- Le **repository NPM** : https://www.npmjs.com/

- Quelques packages courants
  - Grunt
  - Karma
  - Bower
  - Less
  - Yo

Créer un package

- Créer un dossier contenant la description du package
- coder le module dans index.js (point d'entrée apr défaut)
- npm **publish**

La configuration

- La configuration npm se trouve dans le fichier package.json
- Ce fichier doit se trouver dans la racine du projet
- npm **init** crée le fichier

package.json

```
{
  "name": "formula",
  "version": "0.0.0",
  "description": "basic formula",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "BSD-2-Clause"
}
```

valtech.

Installer un package

- npm **install** *package_name*
  - par défaut : le package est installé dans **node_modules**
  - **—save** : il est ajouté au package.json
  - **—save-dev** : il est ajouté au package.json en dépendance dev
  - **—global** : le package est installé globalement (pas dans node_module) et il est disponible en ligne de commande

- Exemple

```
$ npm install jasmine-node --save-dev
```

valtech.

Gestion des packages

- npm **install**
  - installe tous les packages listés dans package.json
- npm **update**
  - met à jour tous les packages listés dans package.json

valtech.

Les commandes npm et run

- npm a une liste de commandes prédéfinies
  - npm list
  - npm install
  - npm test
  - npm addUser
  - npm publish

- npm **help** pour la liste des commandes

Les commandes npm

- des scripts de commande peuvent être définis dans **package.json**

package.json

```
  "main": "index.js",
  "scripts": {
    "hello": "echo Hello World!"
  },
  "author": "",
```

- npm **run** permet de lancer la commande scriptée

```
$ npm run hello

> formula@0.0.0 test /vagrant/npmlab
> echo Hello World!

Hello World!
```

Tester un package

- npm **test**

- Il existe plusieurs frameworks de test
  - QUnit
  - Mocha
  - Jasmine

La commande test

- La commandes test peut être redéfinie

package.json

```
  "main": "index.js",
  "scripts": {
    "test": "node_modules/.bin/jasmine-node spec"
  },
  "author": "",
```

- Lancer la commande scriptée

```
$ npm test

> formula@0.0.0 test /vagrant/npmlab
> node_modules/.bin/jasmine-node spec

.

Finished in 0.02 seconds
1 test, 1 assertion, 0 failures, 0 skipped
```

valtech.

Lab Node et NPM

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab3

- Lab NPM
  - Créer un package
  - Ecrire le module
  - Installer Jasmine en local
  - Ecrire un test
  - Lancer le test via une commande nom

valtech.

Bower

- **Bower** est un gestionnaire de dépendances pour le Web
  - framework et librairies
  - assets
  - utilitaires
- **Site de référence** : http://bower.io/

- Utilise Node et NPM

```
$ npm install -g bower
```

- S'intègre à d'autres outils (Grunt, Yeoman)

Les packages Bower

- Recherche de packages : http://bower.io/search/
  ou `$ bower search <package>`
- Installer un package

```
$ bower install <package>
```

- Lister les packages

```
$ bower list
```

- Pour initialiser le projet et créer le fichier bower.json

```
$ bower init
```

Le fichier de configuration

- La configuration Bower se trouve dans **bower.json**
- Ce fichier se trouve à la racine du projet
- bower **init** crée le fichier

bower.json

```
{
  "name": "bowerlab",
  "version": "0.0.0",
  "homepage": "",
  "license": "MIT",
  "private": true
}
```

valtech.

Installer un package Bower

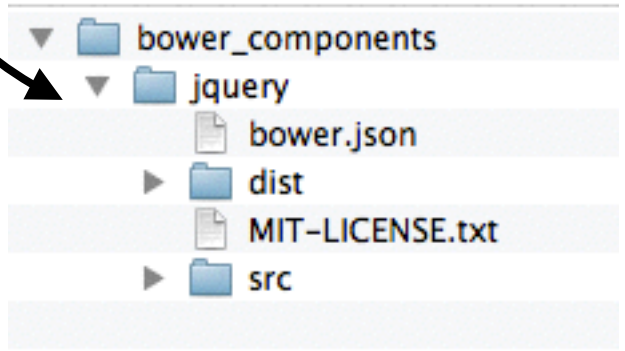
- Installer un package
  - package peut être un package référencé, un repo git ou une URL
  - exemples : bootstrap, jquery, angular-ui-sortable, d3, …
  - —save ajoute la dépendance au fichier bower.json

- Exemple

```
$ bower install —save jquery
```

bower.json

```
{
  "name": "bowerlab",
  "version": "0.0.0",
  "homepage": "",
  "license": "MIT",
  "private": true,
  "dependencies": {
    "jquery": "~2.1.3"
  }
}
```



valtech.

Lab Bower

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab4

- Lab Bower
  - Initialiser le projet Bower
  - Installer le package jquery
  - Lister les packages

Grunt

valtech.

Grunt

- **Grunt** est un outil d'**automatisation de tâches**
- Le site de référence : http://gruntjs.com/
- Des tâches prédéfinies sont fournies en **plugins**
  - JS Hint : compilation et vérification statique du Javacript
  - Less, Sass : gestion du CSS
  - Require JS : Gestion de modules Javascript
  - Uglify : minification
- implémenté en Node JS

L’installation

- L’installation se fait par nom

```
$ npm install -g grunt-cli
$ npm install grunt —save-dev
$ npm install grunt-contrib-jshint --save-dev
```

- **grunt-cli** installe en global la commande **grunt** que vous pourrez taper dans un shell

- le **package grunt** se trouve en local dans le projet. grunt-cli cherche simplement l’outil installé dans le projet depuis n’importe quel sous-répertoire du projet

- Les plugins sont installés de la manière manière

valtech.

La configuration

- **Gruntfile.js** (ou Gruntfile.coffee) décrit les **tâches**
- **package.json** liste les **plugins**
- Ces fichiers sont placés à la racine du projet

```
gruntlab/
  node_modules/
  scripts/
    hello..js
  Gruntfile.js
  jsi
  package.json
```

package.json

```
{
  "name": "gruntlab",
  "version": "0.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error\" && exit 1"
  },
  "author": "",
  "license": "BSD-2-Clause",
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.11.0"
  }
}
```

valtech.

Gruntfile.js

- un module node
- lister les tâches
- lister les packages requis
- décrire les configurations des packages
- lancer **grunt**

package.json | Gruntfile.js

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),

    jshint: {
      all: {
        src: [
          'Gruntfile.js',
          'scripts/{,*/}*.js'
        ]
      }
    }
  });

  // Load the plugin that provides the "jshint" task.
  grunt.loadNpmTasks('grunt-contrib-jshint');

  // Default task(s).
  grunt.registerTask('default', ['jshint']);

};
```

valtech.

Autres tâches

- grunt hello

```
  // Default task(s).
  grunt.registerTask('default', ['jshint']);

  // Hello task
  grunt.registerTask('hello', 'Say Hello', function() {
    grunt.log.write('Hello World!').ok();
  });

};
```

- grunt serve

```
grunt.registerTask('serve', 'Compile then start a connect web server', function (target) {
  if (target === 'dist') {
    return grunt.task.run(['build', 'connect:dist:keepalive']);
  }

  grunt.task.run([
    'clean:server',
    'wiredep',
    'concurrent:server',
    'autoprefixer',
    'connect:livereload',
    'watch'
  ]);
});
```

valtech.

Lab Grunt

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab5

- Lab Grunt
  - Initialiser le projet Grunt
  - Utiliser jshint
  - Ajouter une tâche hello

Gulp

- Gulp est un autre build manager basé sur Node
- La configuration est similaire, un peu plus simple

- http://gulpjs.com/

valtech.

Angular

- Angular est un framework Web en Javascript

- Site de référence : https://angularjs.org/

- MVC (Model-View-Controller)
- Two-Way Data Binding
- templates
- injection de dépendance
- utilisation des éléments DOM standards et directives custom
- testable unitairement

- Alternatives : Backbone, EmberJS, Meteor, Polymer …

Les modules Angular

- L'application a un module principal, l'application
- D'autres modules peuvent être rajoutés

- Attention : module Angular ≠ module Node

- Les contrôleurs, services, filtres sont rattachés à un module

```
angular.module('twinApp')
  .controller('MainCtrl', function ($scope) {
    $scope.awesomeThings = [
      'HTML5 Boilerplate',
      'AngularJS',
      'Karma'
    ];
  });
```

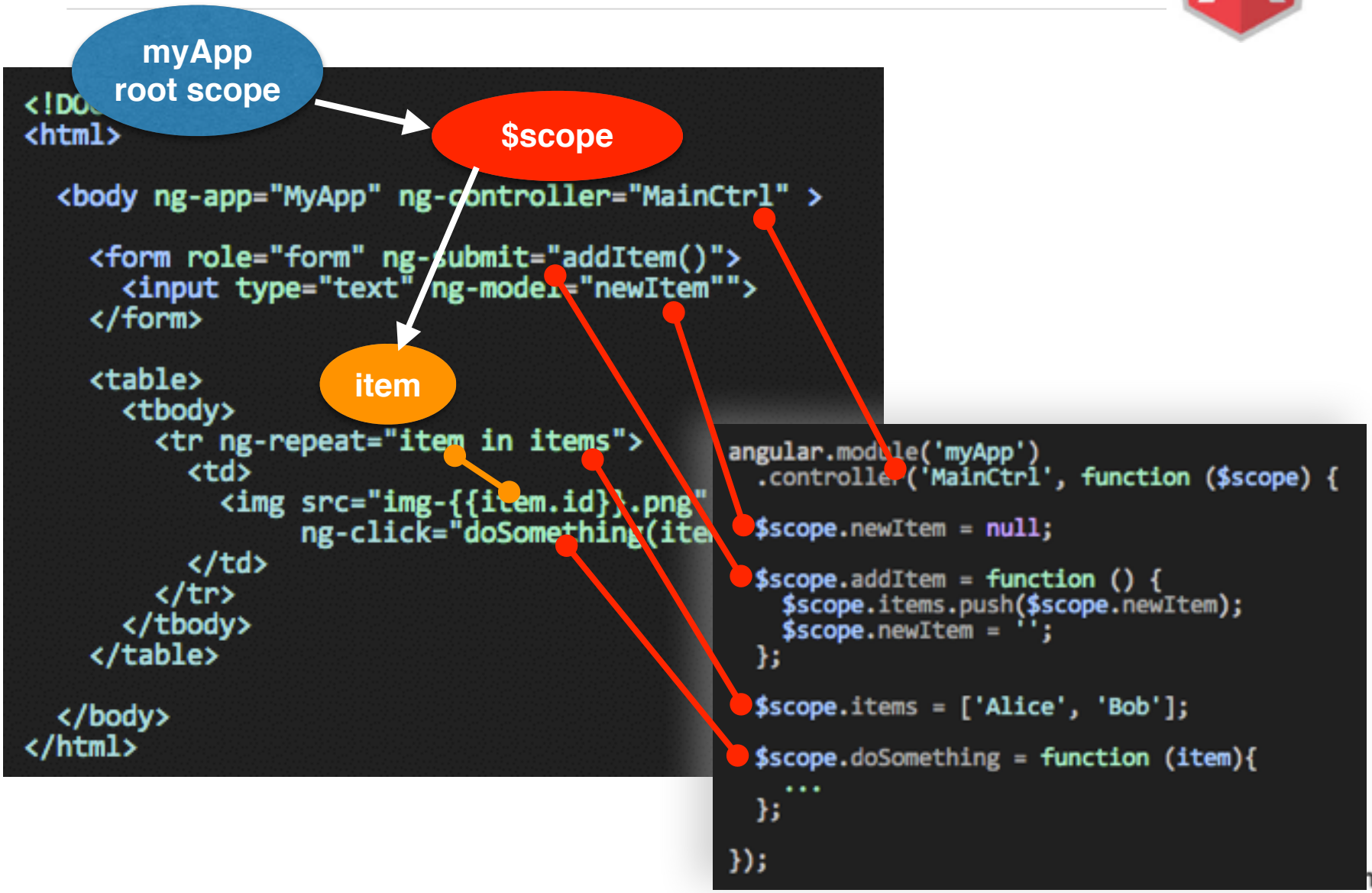
Les vues

- Templating
- Directives

```
<!DOCTYPE html>
<html>

  <body ng-app="MyApp" ng-controller="MainCtrl" >

    <form role="form" ng-submit="addItem()">
      <input type="text" ng-model="newItem"">
    </form>

    <table>
      <tbody>
        <tr ng-repeat="item in items">
          <td>
            <img src="img/{{item.id}}.png"
                 ng-click="doSomething(item)"/>
          </td>
        </tr>
      </tbody>
    </table>

  </body>
</html>
```

valtech.

Scopes et Two-ways data binding



Les routes

```
angular
  .module('mytodoApp', [
    'ngCookies',
    'ngResource',
    'ngRoute',
    'ngSanitize'
  ])
  .config(function ($routeProvider) {
    $routeProvider
      .when('/', {
        templateUrl: 'views/main.html',
        controller: 'MainCtrl'
      })
```

index.html

```
<!doctype html>
<html class="no-js">
  <body ng-app="mytodoApp">

      <div ng-view=""></div>

  </body>
</html>
```

main.html

```
<div class="container">

    <form role="form" ng-submit="addItem()">
      <input type="text" ng-model="newItem"">
    </form>

    <table>
      <tbody>
        <tr ng-repeat="item in items">
          <td>
            <img src="img-{{item.id}}.png"
                 ng-click="doSomething(item)"/>
          </td>
        </tr>
      </tbody>
    </table>

</div>
```

valtech.

Yeoman

- **Yeoman** est un outil de Web scaffolding (échafaudage)
- **Site de référence** : http://yeoman.io/

- il permet
  - de **générer un projet** javascript pré-configuré
  - utilisant bower et grunt/gulp
  - initialisé avec les principaux packages

- utilise différents modèles de projet (**generators**)
  - node : projet node
  - angular : projet angular front end avec grunt
  - angular-gulp : projet angular front end avec gulp
  - jhipster : projet angular + java Spring
  - cordova : projet mobile en cordova
  - …

Démarrer un projet angular

- Getting Started : http://yeoman.io/codelab.html
- Installer yo

```
$ npm install -g yo
$ yo —version
npm update yo
```

- Installer le générateur angular

```
$ npm install --global generator-angular
```

  - https://www.npmjs.com/package/generator-angular

- Créer le projet

```
$ yo angular [app-name]
```

Le scaffolding

Generators

Available generators:

- angular (aka angular:app)
- angular:controller
- angular:directive
- angular:filter
- angular:route
- angular:service
- angular:provider
- angular:factory
- angular:value
- angular:constant
- angular:decorator
- angular:view

- Créer un service

```
$ yo angular:service myService
```

- produit app/scripts/services/aService.js

```
angular.module(‘myModule)
  .service('myService', function () {
  // ...
});
```

- yo angular:factory,
  yo angular:provider,
  yo angular:value,
  yo angular:constant

valtech.

Les options de configuration

- Si vous ne savez pas comment le générateur s'appelle :

```
$ yo
[?] What would you like to do?
…
```

- Certains générateurs vous proposeront des options
  - **Oui/Non** : y/n (la valeur par défaut est en Majuscule)
  - **choix** dans une liste : se déplacer avec les flèches puis Enter
  - **choix multiple** : (dé)sélection dans une liste par Space puis Enter pour valider

- Certains générateurs acceptent des **customisations**

```
$ yo  angular —coffee
```

valtech.

Les fichiers générés

```
twinProject/
  app/
    images/
    scripts/
    styles/
    views/
    404.html
    favicon.ico
    index.html
    robots.txt
  bower_components/
  node_modules/
  test/
    spec/
    karma.conf.js
  bower.json
  Gruntfile.js
  package.json
```

- Une arborescence
- Les **fichiers de configuration** de npm, bower et grunt
  - package.json
  - bower.json
  - Gruntfile.js
- Un squelette d'**application Angular**
- Un squelette de **suite de test**

valtech.

Le Gruntfile

- Le Gruntfile généré est très complexe

- Les tâches
  - **défault** : jshint, test et build
  - **test** : lance le test runner karma
  - **build** : fabrique la distribution (dossier dist)
    - clean le dossier dist
    - wiredep : met à jour les dépendances
    - cssmin, htmlmin : tâches de minification
    - concat : fusionne des fichiers pour réduire le nombre requêtes
    - uglify : diverses tâches de compression et mise en forme
  - **serve** : démarre un serveur web de dev
    - live reload : recharge les ressources modifiées et notify le navigateur
  - **serve:dist** : démarre un serveur sur la distribution

valtech.

Wiredep

- Dans index.html
  - Ce block est géré automatiquement par wiredep

```
  <!-- build:js(.) scripts/vendor.js -->
   <!-- bower:js -->
   <script src="bower_components/jquery/dist/jquery.js"></script>
   <script src="bower_components/angular/angular.js"></script>
   <script src="bower_components/bootstrap/dist/js/bootstrap.js"></script>
   <script src="bower_components/angular-cookies/angular-cookies.js"></script>
   <script src="bower_components/angular-resource/angular-resource.js"></script>
   <script src="bower_components/angular-route/angular-route.js"></script>
   <script src="bower_components/angular-sanitize/angular-sanitize.js"></script>
   <!-- endbower -->
   <!-- endbuild -->
```

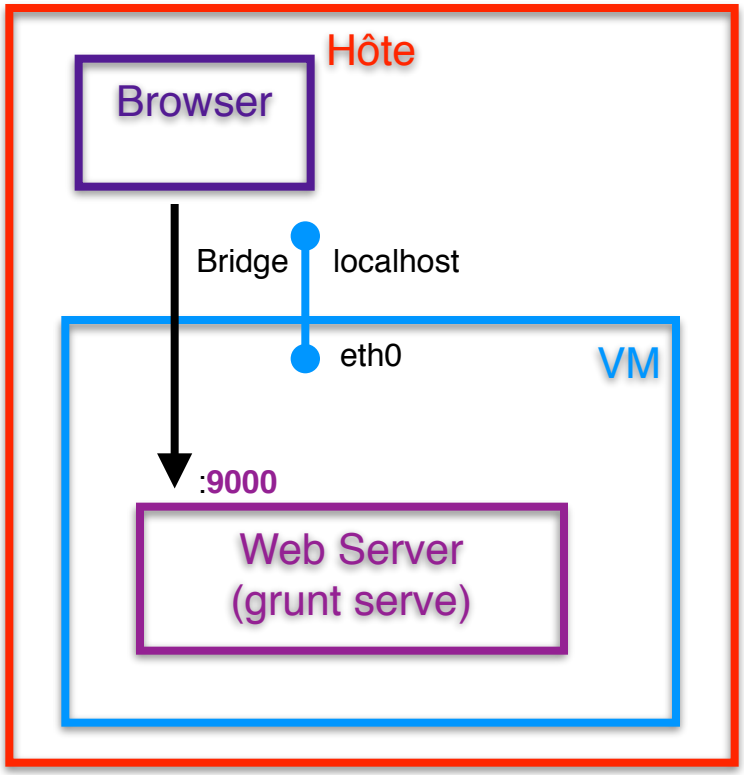
  - Ce block indique à concat de concaténer ces fichiers dans scripts.js lors du build

```
        <!-- build:js({.tmp,app}) scripts/scripts.js -->
        <script src="scripts/app.js"></script>
        <script src="scripts/controllers/main.js"></script>
        <script src="scripts/controllers/about.js"></script>
        <!-- endbuild -->
```

valtech.

Accès depuis le navigateur de l'hôte

- Le serveur Web de dev se lance avec grunt serve
- Il met à jour index.html

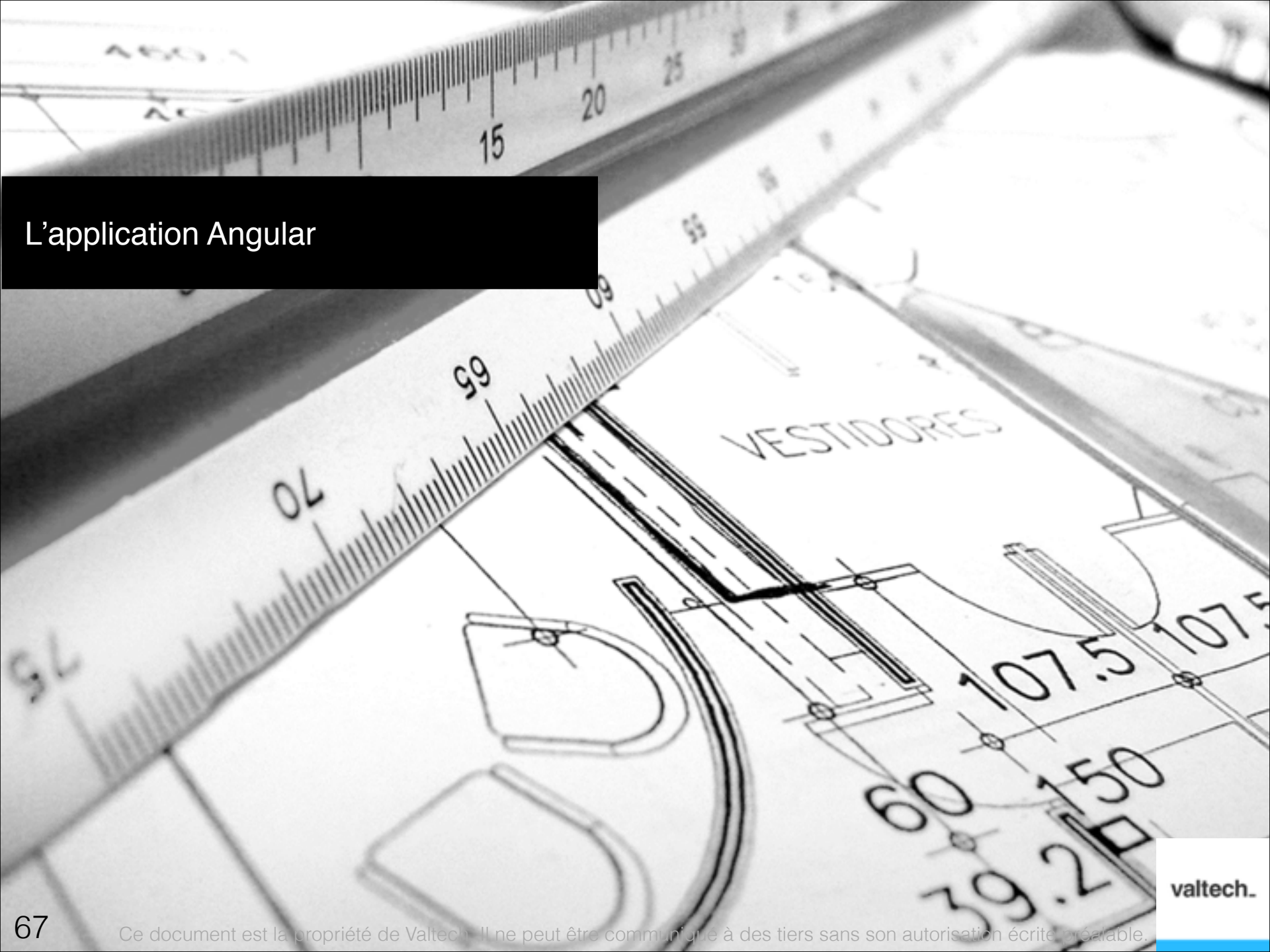


Lab Yeoman

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab6

- Lab Yeoman
  - Installer le générateur Angular
  - Initialiser le projet Twin
  - Lancer le serveur et accéder à la page par défaut

L’application Angular



Ajouter une vue 1/3

- **angular:route** ajoute la vue, le contrôleur et la route

```
$ yo angular:route player
   invoke   angular:controller:/usr/local/lib/
node_modules/generator-angular/route/index.js
   create     app/scripts/controllers/player.js
   create     test/spec/controllers/player.js
   invoke   angular:view:/usr/local/lib/node_modules/
generator-angular/route/index.js
   create     app/views/player.html
```

- La route dans app/scripts/app.js

```
    .when('/player', {
      templateUrl: 'views/player.html',
      controller: 'PlayerCtrl'
    })
```

valtech.

Ajouter une vue 1/3

- la vue app/views/player.js

```
<p>This is the player view.</p>
```

- le contrôleur app/scripts/player.js

```
angular.module('twinApp')
  .controller('PlayerCtrl', function ($scope) {
    $scope.awesomeThings = [
      'HTML5 Boilerplate',
      'AngularJS',
      'Karma'
    ];
  });
```

- Ajoute ces fichiers dans index.html

Ajouter une vue 1/3

- le test test/spec/controllers/player.js

```
'use strict';

describe('Controller: PlayerCtrl', function () {

  // load the controller's module
  beforeEach(module('twinApp'));

  var PlayerCtrl,
    scope;

  // Initialize the controller and a mock scope
  beforeEach(inject(function ($controller, $rootScope) {
    scope = $rootScope.$new();
    PlayerCtrl = $controller('PlayerCtrl', {
      $scope: scope
    });
  }));

  it('should attach a list of awesomeThings to the scope', function () {
    expect(scope.awesomeThings.length).toBe(3);
  });
});
```

valtech.

Les outils de test

Tests unitaires et spec

- Javascript Engine

- Test framework

- Test Runner

plugins

karma-jasmine

karma-firefox-launcher

- Environnement d'exécution

valtech.

Karma

- **Test Runner** pour Javascript
- **Site de référence** : http://karma-runner.github.io

- Supporte différents frameworks de test

- **Fonctions** :
  - Lance la suite de test
  - Permet un mode autotest
  - Interface Web de suivi des tests
  - Génère un fichier de résultat au format XUnit pour le reporting

Jasmine

- **Framework de test BDD** - RSpec
- **Site de référence** : http://jasmine.github.io/

- http://jasmine.github.io/2.1/introduction.html

- **Exemple** de test

formula-spec.js

```
var formula = require("../index.js");

describe("Formula", function () {
  it("should compute the area of the square", function () {
    var area = formula.squareArea(5);
    expect(area).toBe(25);
  });
});
```

BeforeEach et Inject

- **beforeEach** se charge du setup, **inject** interface avec Angular

```
'use strict';

describe('Controller: PlayerCtrl', function () {

  // load the controller's module
  beforeEach(module('twinApp'));

  var PlayerCtrl,
    scope;

  // Initialize the controller and a mock scope
  beforeEach(inject(function ($controller, $rootScope) {
    scope = $rootScope.$new();
    PlayerCtrl = $controller('PlayerCtrl', {
      $scope: scope
    });
  }));

  it('should attach a list of awesomeThings to the scope', function () {
    expect(scope.awesomeThings.length).toBe(3);
  });
});
```

Karma + Jasmine + PhantomJS

- **npm install** —save-dev
  - karma
  - karma-jasmine
  - karma-phantomjs-launcher
- Le fichier de **configuration**
  - karma.conf.js
- **grunt test**
  - lance le test **sans** l'autotest
- **npm test**
  - à configurer pour pouvoir lancer les tests en autotest

karma.conf.js

```
module.exports = function(config) {
  config.set({

    autoWatch: true,

    basePath: '../',

    frameworks: ['jasmine'],

    files: [
      'bower_components/angular/angular.js',
      'bower_components/angular-mocks/angular-mocks.js',
      'bower_components/angular-cookies/angular-cookies.js',
      'bower_components/angular-resource/angular-resource.js',
      'bower_components/angular-route/angular-route.js',
      'bower_components/angular-sanitize/angular-sanitize.js',
      'app/scripts/**/*.js',
      'test/mock/**/*.js',
      'test/spec/**/*.js'
    ],

    port: 8080,

    browsers: [
      'PhantomJS'
    ],

    // Which plugins to enable
    plugins: [
      'karma-phantomjs-launcher',
      'karma-jasmine'
    ],

    singleRun: false

  });
};
```

Lab Angular Karma

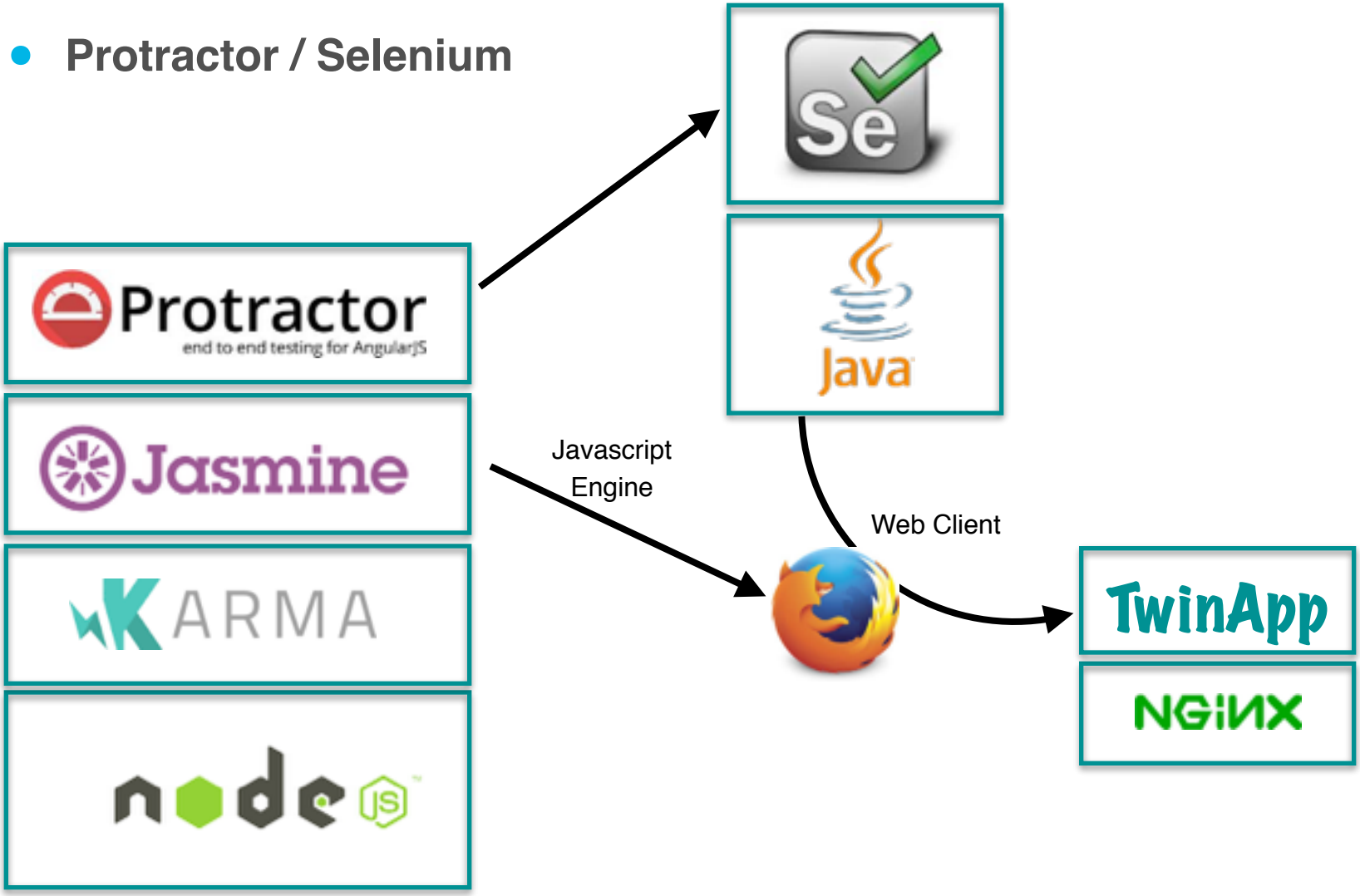
- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab7

- Coder la vue Player
  - Utiliser yeoman pour créer une route
  - Utiliser Karma en autotest
  - Coder la vue, le contrôleur et le test du contrôleur

Les architectures Java SE

End-to-end / GUI tests

- **Protractor / Selenium**



Exemple de test Protractor

Protractor
end to end testing for AngularJS

- Structure d'un test Jasmine

player-view-test.js

```
'use strict';
describe('Player view End-2-End', function() {
    var playerTable
    var newPlayer

    beforeEach(function() {
      browser.get("#/player.html");

      playerTable = element.all(by.repeater('player in players'))
      newPlayer = element(by.model('newPlayer'))
    })

    describe('When Main view appears', function() {

      it('has the title Players', function() {

         expect(element(by.tagName('h2')).getText()).toBe('Players')

      })
```

valtech.

Protractor

Protractor
end to end testing for AngularJS

- Le résultat est similaire à celui de Jasmine

```
#################################################################
############### Running Test suite
Starting selenium standalone server...
[launcher] Running 1 instances of WebDriver
Selenium standalone server started at http://172.17.0.4:43732/wd/hub
...

Finished in 9.553 seconds
3 tests, 3 assertions, 0 failures

Shutting down selenium standalone server.
[launcher] 0 instance(s) of WebDriver still running
[launcher] firefox #1 passed

Done, without errors.
```

valtech.

L'API Protractor

Protractor
end to end testing for AngularJS

- **Emulation du navigateur**

- Lire la page et obtenir le DOM : **browser.get()**

- Obtenir un des éléments du DOM : **element(….)**

- Lire le texte d'un élément : **element.getText()**
- Lire un attribut de l'élément : **element.getAttribute('attr')**

- Cliquer sur un element : **element.click()**
- Entrer une valeur dans un champs : **element.sendKeys('OK')**

valtech.

L'API Protractor

- **Recherche d'un élément**

- **element( )**
- element.**all( )**

- **by.**model(bindingName)
- by.css(classname)
- by.tagName(html-element)
- by.id(id)
- by.repeater(repeater)

valtech.

La configuration

Protractor
end to end testing for AngularJS

- **protractor.conf.js**
- Généralement pas intégré dans la suite de test Karma
- protractor est un **package npm**
- Vous pouvez l'installer en global et le lancer avec protractor protractor.conf.js
- Installation complexe -> Isolé dans un container docker

protractor.conf.js

```
exports.config = {

  allScriptsTimeout: 11000,

  specs: [
    '*.js'
  ],

  capabilities: {
    'browserName': 'firefox'
  },

  baseUrl: 'http://localhost:9000/',

  framework: 'jasmine',

  jasmineNodeOpts: {
    defaultTimeoutInterval: 30000
  }
};
```

valtech.

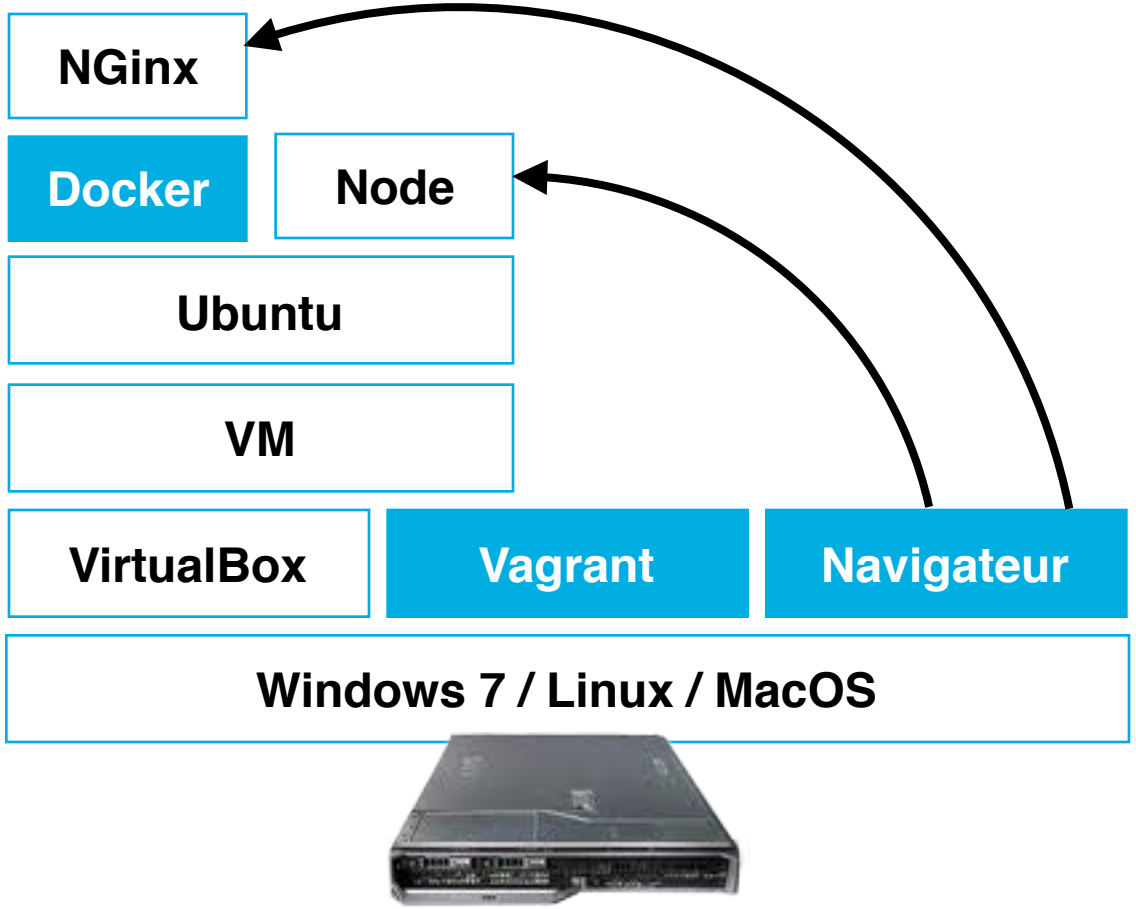
Docker

- Docker est un outil de gestion de container (VM très très light)

- LXC https://linuxcontainers.org/
- Linux

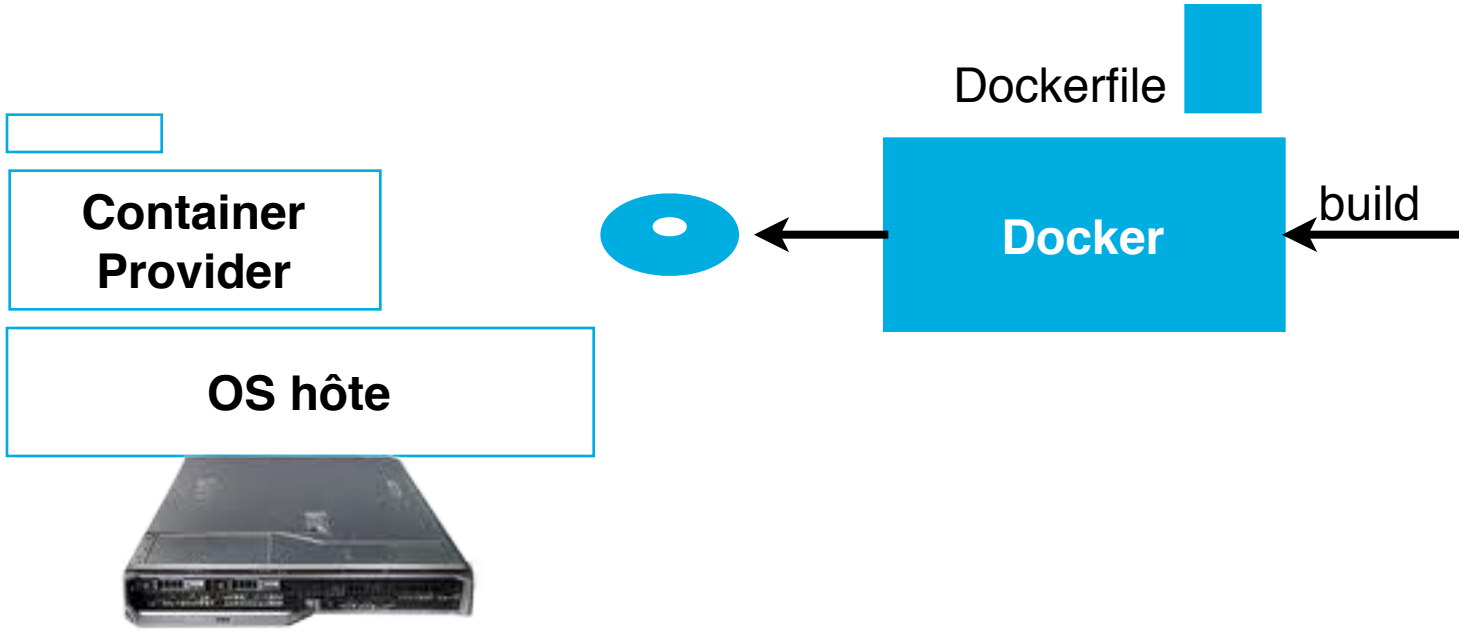
- repose sur chroot et cgroup
  - A chroot on Unix operating systems is an operation that changes the apparent root directory for the current running process and its children. - Wikipedia
  - cgroups (abbreviated from control groups) is a Linux kernelfeature to limit, account, and isolate resource usage (CPU, memory, disk I/O, etc.) of process groups. - Wikipedia

L’architecture pour ce cours



Docker - Créer le container

- Le container est décrit dans le Dockerfile

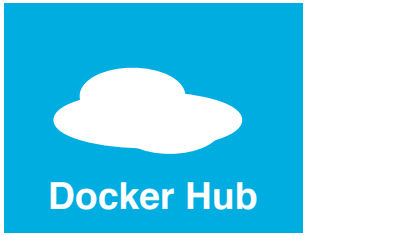


Dockerfile

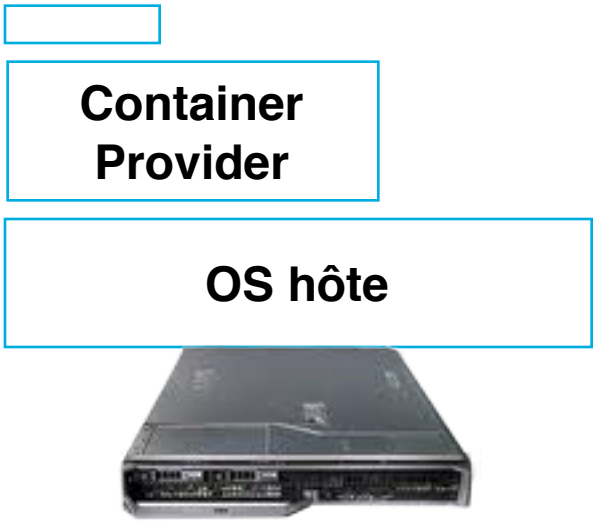

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nginx
RUN echo "daemon off;" >> /etc/nginx/nginx.conf

ADD conf/default /etc/nginx/sites-available/default

EXPOSE 80

CMD ["nginx"]
```

Docker - Obtenir l'image de la VM

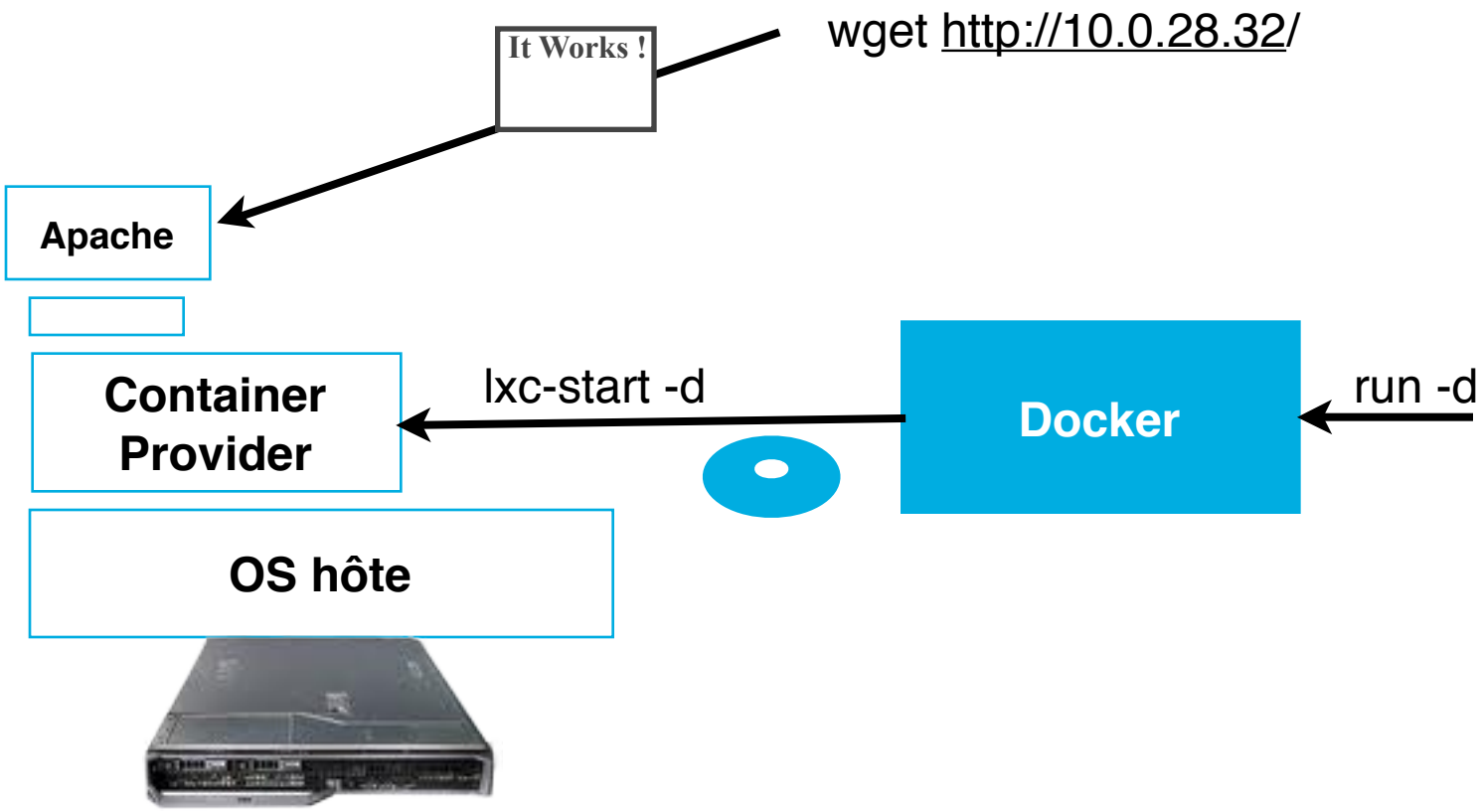
- DockerHub permet de partager des containers déjà buildés



pull



Docker - Utiliser le container



Quelques commandes Docker

- Démarrer un container docker run

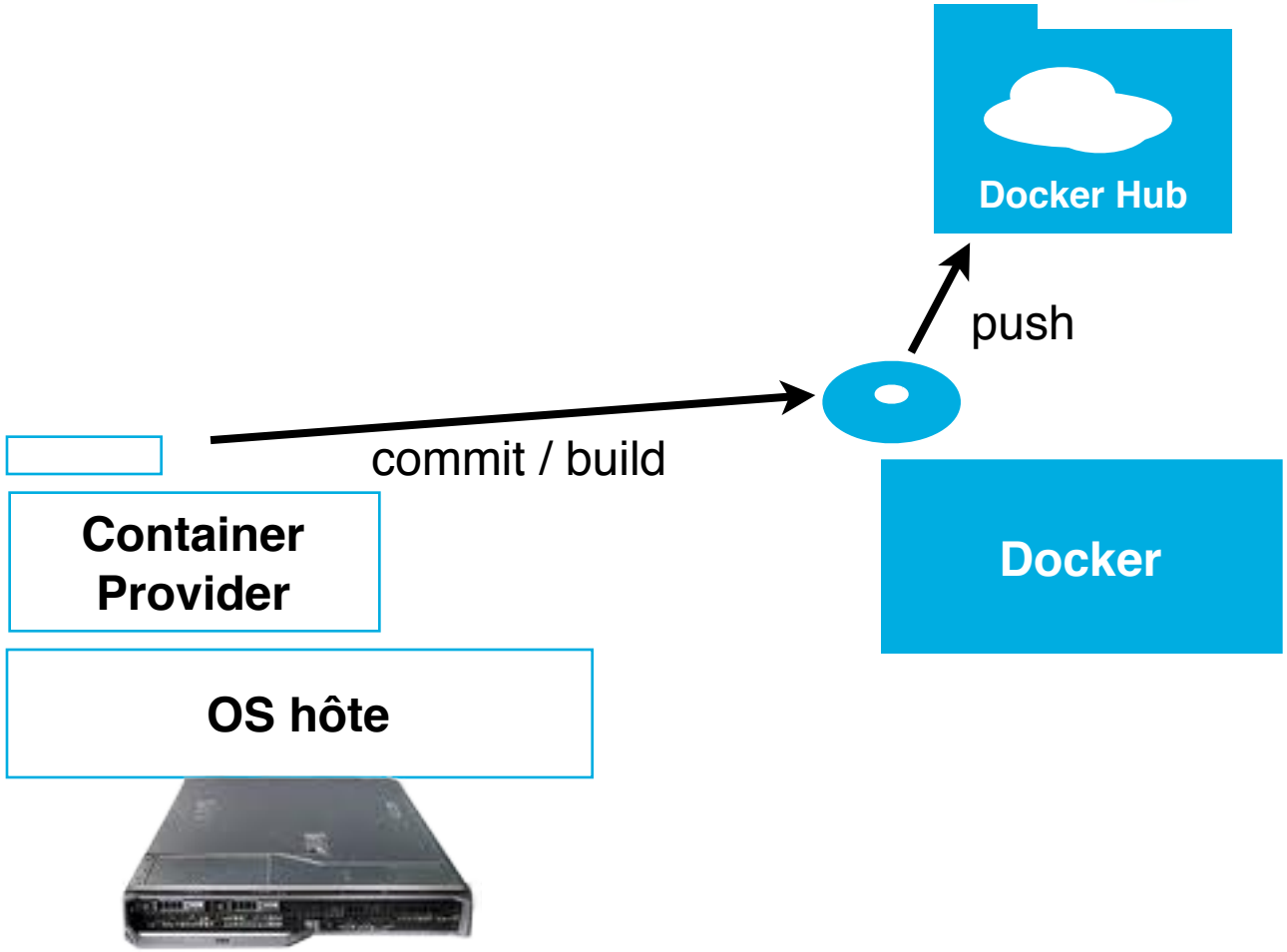
```
$ sudo docker run —name protractor cfalguiere/protractor-test
```

- voir les containers : docker ps
  - affiche l'id

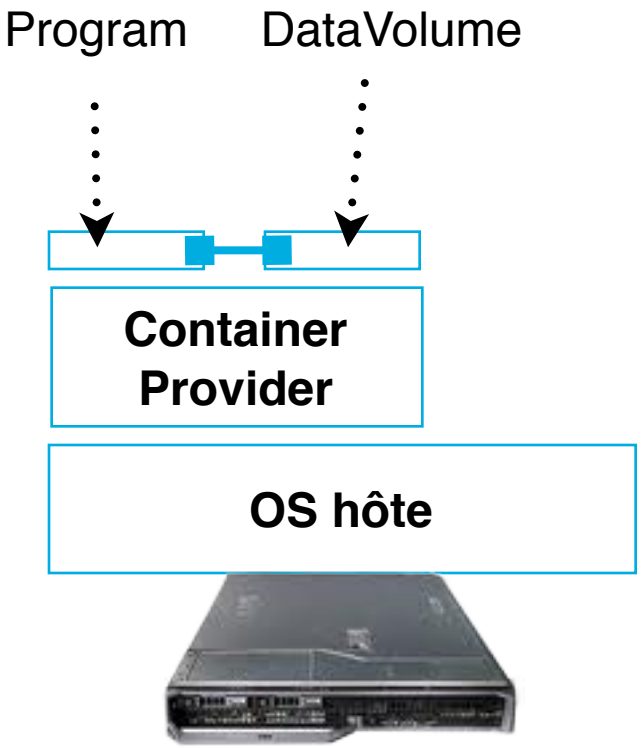
- arrêter un container : docker stop [nom I id]
- supprimer un container : docker rm [nom I id]

valtech.

Docker - Créer une autre image

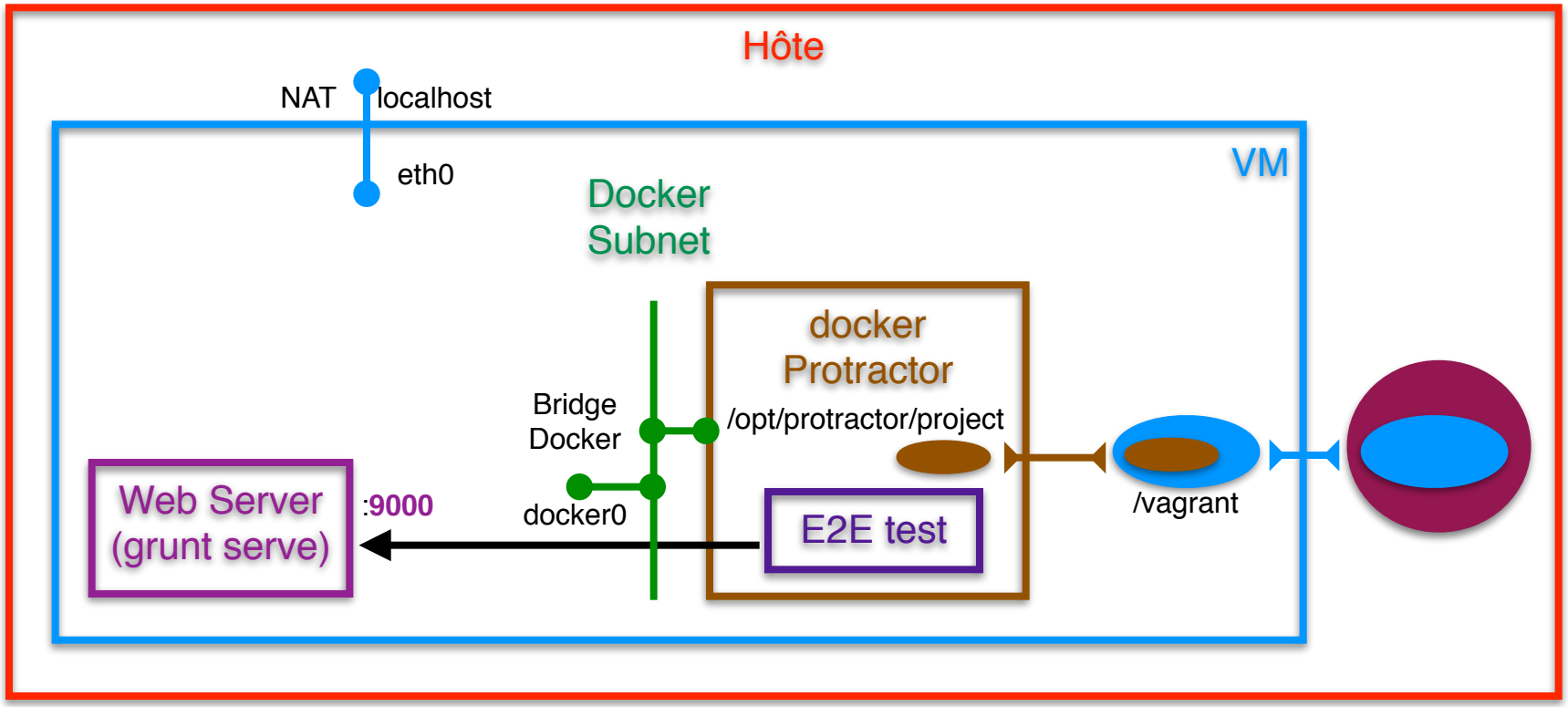


Data Volumes



La configuration pour docker

- Pour accéder au serveur Web, il faut passer par l'interface docker0 de la VM (généralement 172.17.42.1)



Le container Protractor

- démarrer un container docker run avec la variabilisation

```
$ sudo docker run —name protractor  \
                 —volume $PWD:/opt/protractor/project \
                 —env TEST_FILE=test/e2e/protractor.conf.js \
                 cfalguiere/protractor-test
```

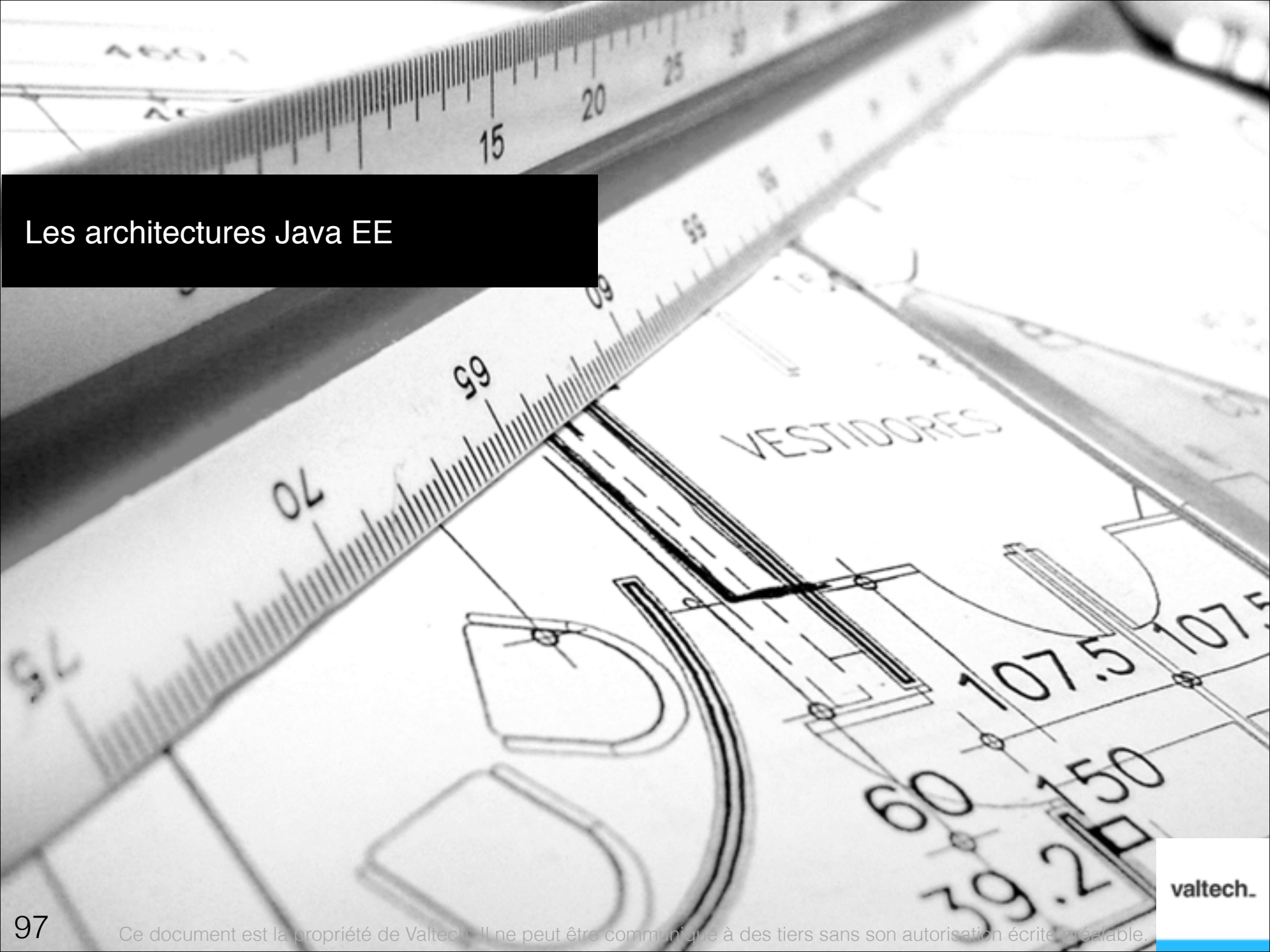
- BaseUrl doit être modifié
  - généralement http://172.17.42.1:9000

Lab Protractor

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab8

- Lab Protractor
  - Ecrire un test End-to-End
  - Utiliser Protractor

Les architectures Java EE



Factory et Service

- Angular défini une notion de provider qui est un singleton qui fournir une valeur ou un service
- Il y a différents raccourcis d'écriture
  - service : la fonction associée au service déclare des opérations
  - factory : la fonction associée à la factory retourne un objet ou une liste
  - value et constant : une valeur est associée au provider

Factory et Service

- factory

```
angular.module('twinApp')
  .factory('CardFactory', function () {
    ...

    var cards = [];

    ...

    return cards;

  });
```

- service

```
angular.module('twinApp')
  .service('Boardservice', function (CardFactory) {

    this.deal = function() {
      ...
    };

    this.sortedCellsByCardId = function() {
      ...
    };

    this.getCellAt = function(row, column) {
      return board[row][column];
    }

  });
```

valtech.

Le mock de la factory

Jasmine

- $provide permet de déclarer directement une valeur

```
// mock CardFactory
beforeEach(function () {
  var cardsDependency =   [
                { id: 1, shape: 'heart', color: 'red'},
                { id: 2, shape: 'heart', color: 'blue'},
                { id: 3, shape: 'star', color: 'red'},
                { id: 4, shape: 'star', color: 'blue'},
                { id: 5, shape: 'music', color: 'red'},
                { id: 6, shape: 'music', color: 'blue'}
            ];

  module(function ($provide) {
    $provide.value('CardFactory', cardsDependency);
  });
});
```

Attention à l'ordre des beforeEach

Jasmine

```
describe('Service: Boardservice', function () {

    // load the service's module
    beforeEach(module('twinApp'));

    // mock CardFactory
    beforeEach(function () {
      var cardsDependency =   [
                    { id: 1, shape: 'heart', color: 'red'},
                    { id: 2, shape: 'heart', color: 'blue'},
                    { id: 3, shape: 'star', color: 'red'},
                    { id: 4, shape: 'star', color: 'blue'},
                    { id: 5, shape: 'music', color: 'red'},
                    { id: 6, shape: 'music', color: 'blue'}
                ];

      module(function ($provide) {
        $provide.value('CardFactory', cardsDependency);
      });
    });

    // instantiate service
    var Boardservice;
    beforeEach(inject(function (_Boardservice_) {
      Boardservice = _Boardservice_;
    }));
```

valtech.

Mock du service

Jasmine

```
// defines a 4x4 board
beforeEach( function () {
  var board = // ...

  var boardDependency =  {
    sortedCellsByCardId : function() {
      return cells;
    },
    getCellAt: function (row, column) {
      return board[row][column];
    },
    deal: function () {
      return board;
    }
  };

  module(function ($provide) {
    $provide.value('Boardservice', boardDependency);
  });
});
```

valtech.

Lab Services et Mocks

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab9

- Lab Service et Mock
  - Ecrire une factory et son test
  - Ecrire un service et son test
  - Mocker les services dans le contrôleur
  - Ecrire un filtre et son test

Le déploiement

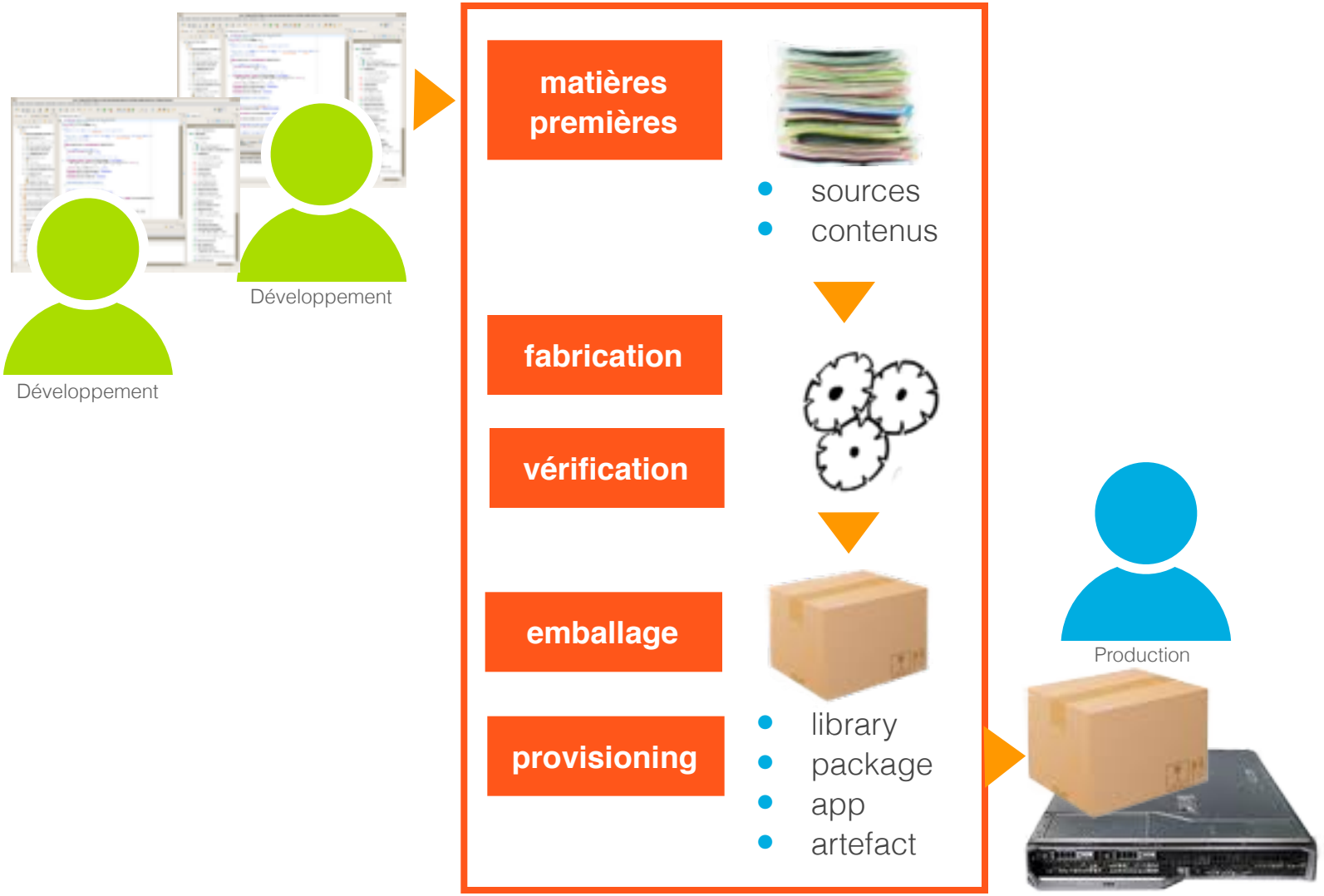
Lab déploiement

- https://github.com/cfalguiere/Hands-On-Angular-Tooling/wiki
- Lab10

- Lab Deploiement
  - Builder l'application avec Grunt
  - Créer un container docker pour NGinx
  - Lancer l'application dans NGinx

L’usine logicielle

Intégration continue - Etapes



Intégration Continue - Exemple d'outillage Java EE

matières premières

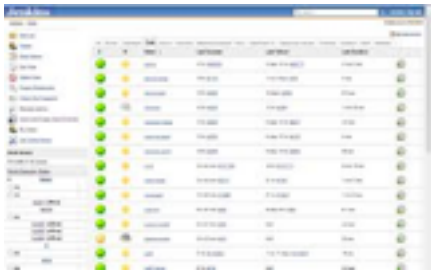
Source management
Git
Subversion

fabrication

vérification

Build management
Jenkins

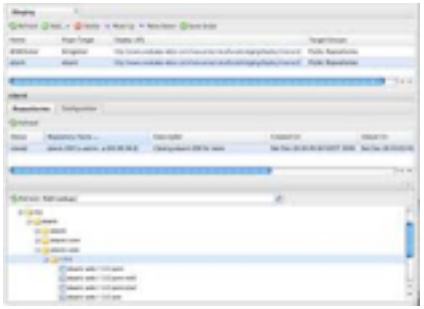
Plugins
Maven
Sonar
...



emballage

provisioning

Provisioning repository
Nexus,



valtech.