Serverspec in cloud provision

Serguei Kouzmine
kouzmine_serguei@yahoo.com

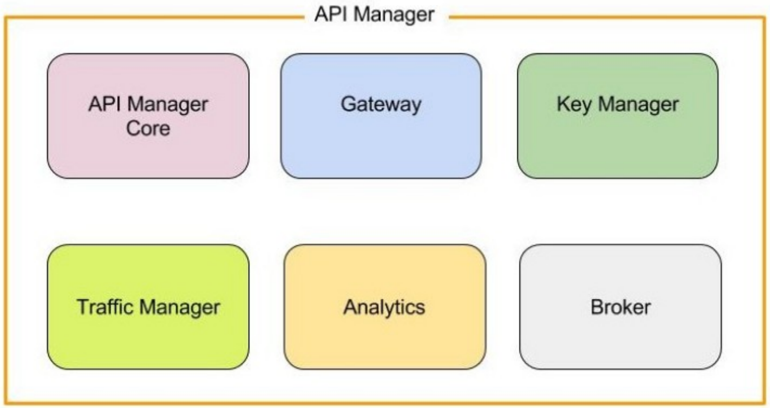
Serverspec in cloud provision

Successful operation automation usually needs, along with core tool technical skills some knowledge about the system under test - not strictly but is recommended. For example, (no)SQL, Angular would be a great help in web testing, HTML / Javascript no longer sufficient .

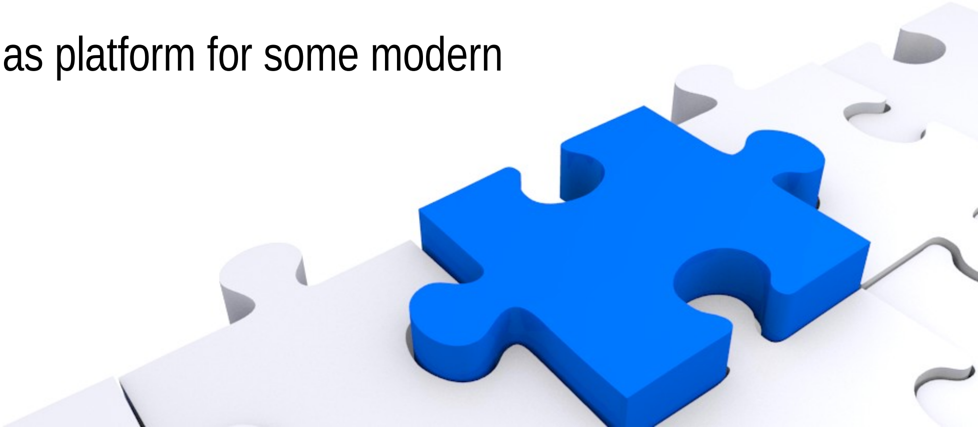
Often more critical than provision engine (Puppet, Chef, Ansible, DSC learning curve.

QA engineer using Selenium or Katalon often is or willing to grow as Web developer but unlikely ever interested in the code base of those tools.

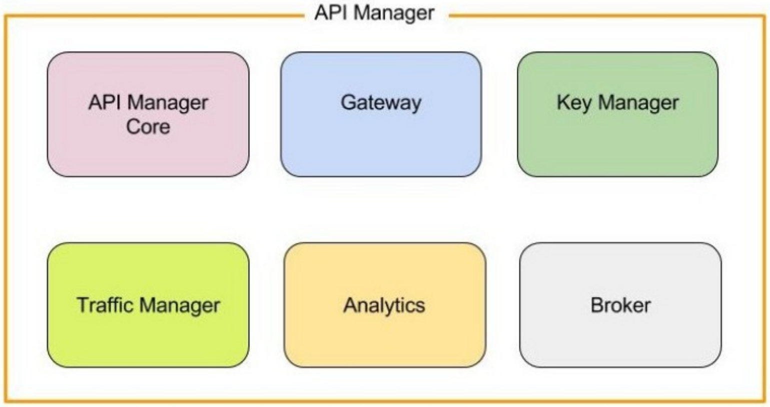
IaaS to PaaS mutation challenge



Cloud cluster provisioned by some automation workflow serves as platform for some modern application stack



IaaS to PaaS mutation challenge



API Manager

API Manager Core

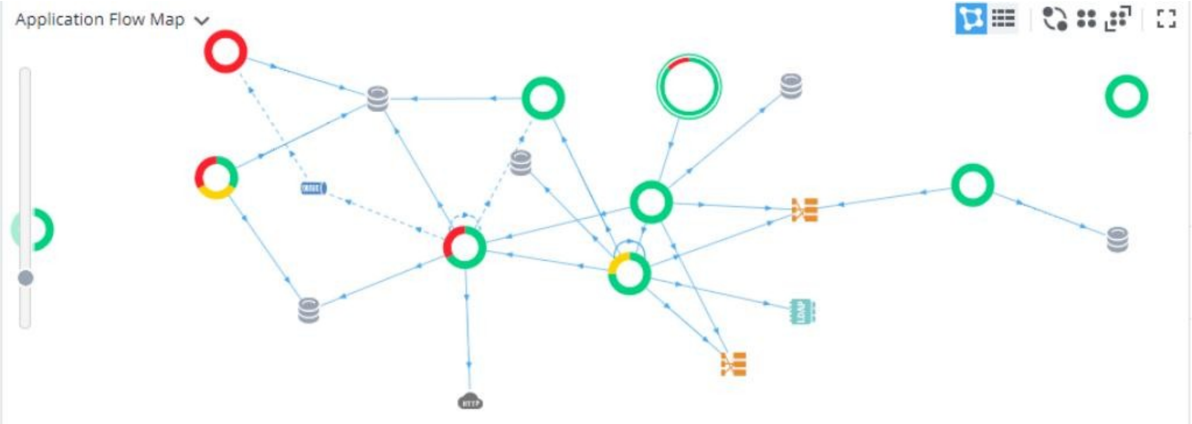
Gateway

Key Manager

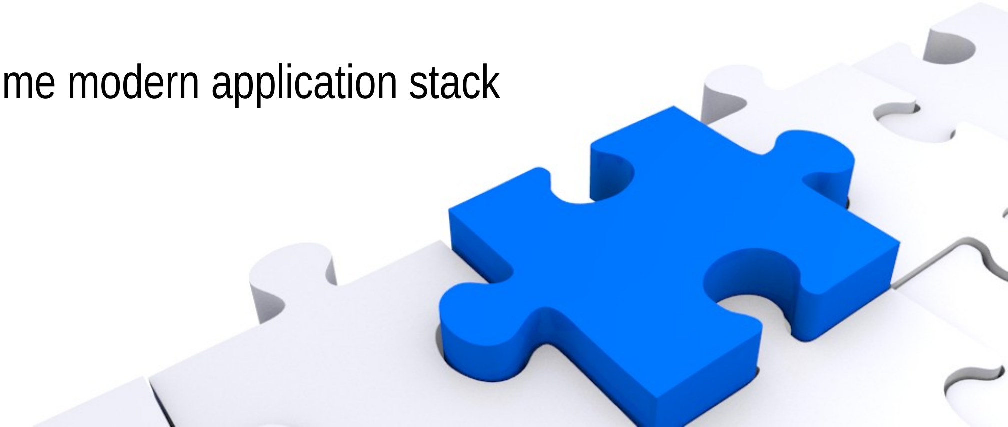
Traffic Manager

Analytics

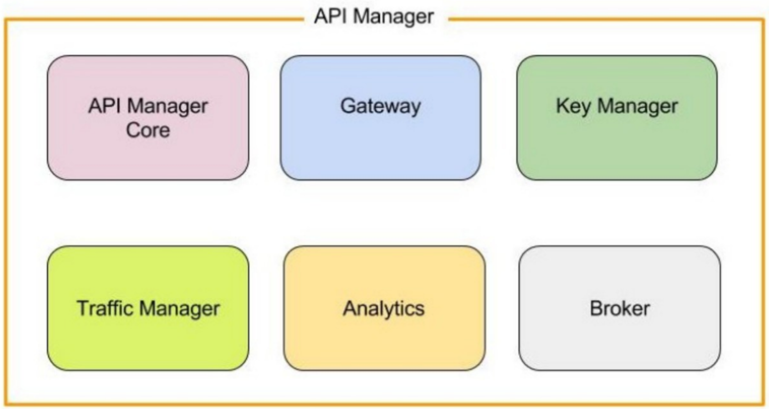
Broker

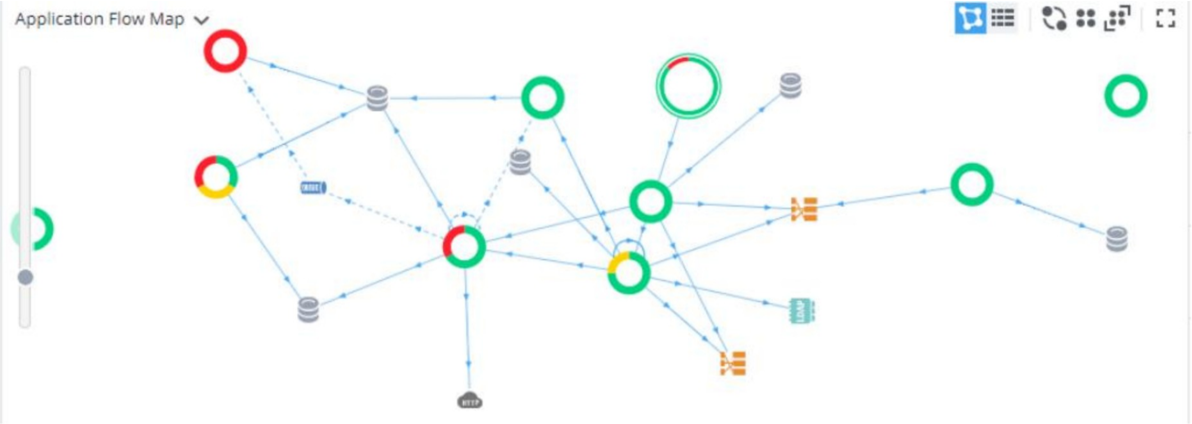


Cloud cluster provisioned by Puppet viewed as platform for some modern application stack



IaaS to PaaS mutation challenge

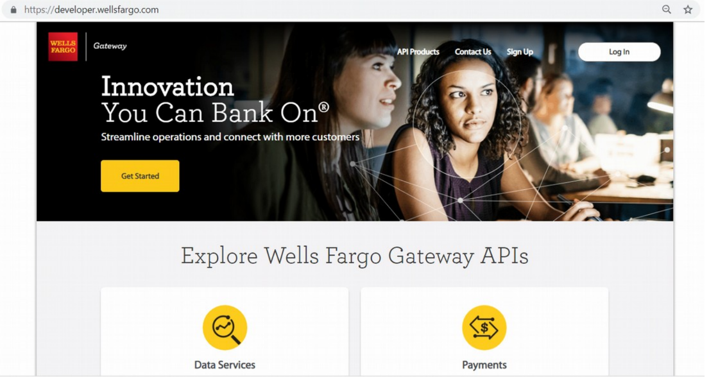




- WSO2 API Manager
- WSO2 API Manager Analytics
- WSO2 App Manager
- WSO2 Application Server
- WSO2 Business Process Server
- WSO2 Business Rules Server
- WSO2 Carbon
- WSO2 Complex Event Processor
- WSO2 Dashboard Server
- WSO2 Data Analytics Server
- WSO2 Data Services Server
- WSO2 Elastic Load Balancer
- WSO2 Enterprise Integrator
- WSO2 Enterprise Mobility Manager
- WSO2 Enterprise Service Bus
- WSO2 Enterprise Service Bus Analytics
- WSO2 Enterprise Store
- WSO2 Governance Registry
- WSO2 Identity Server
- WSO2 Identity Server Analytics
- WSO2 Identity Server Key Manager
- WSO2 IoT Server
- WSO2 Machine Learner
- WSO2 Message Broker
- WSO2 Private PaaS
- WSO2 Storage Server

https://wso2.com/

IaaS to PaaS mutation challenge



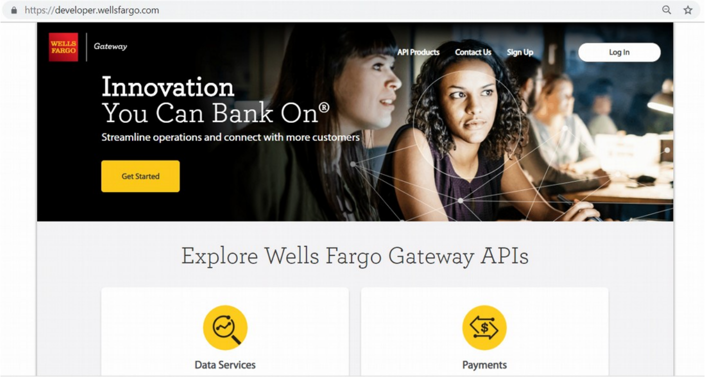
https://developer.wellsfargo.com

Innovation
You Can Bank On®

Streamline operations and connect with more customers

Explore Wells Fargo Gateway APIs

IaaS to PaaS mutation challenge



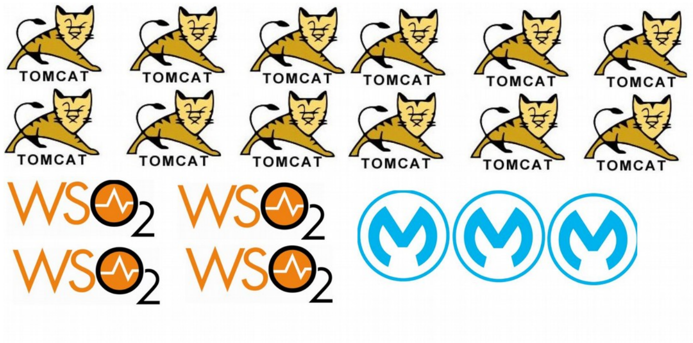
Innovation
You Can Bank On®

Streamline operations and connect with more customers

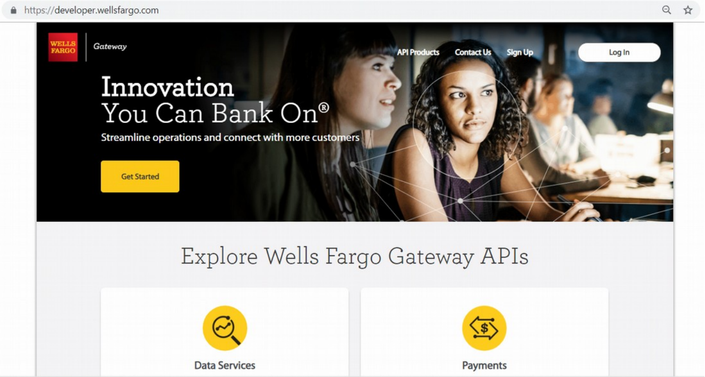
Explore Wells Fargo Gateway APIs

Data Services

Payments



IaaS to PaaS mutation challenge

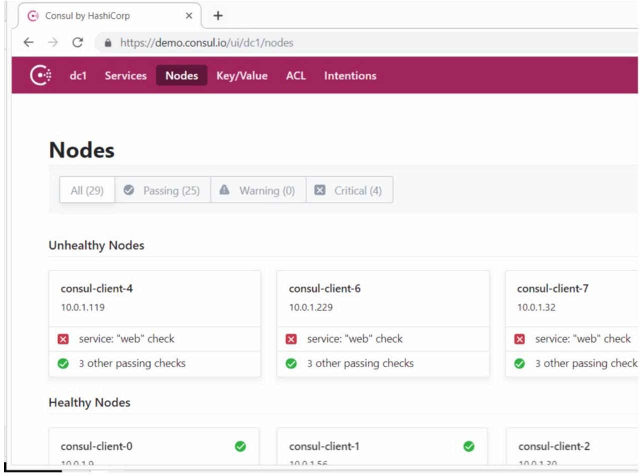


https://developer.wellsfargo.com

Innovation
You Can Bank On®

Streamline operations and connect with more customers

Explore Wells Fargo Gateway APIs



https://demo.consul.io/ui/dc1/nodes

Nodes

Unhealthy Nodes

consul-client-4
10.0.1.119
☒ service: "web" check
✓ 3 other passing checks

consul-client-6
10.0.1.229
☒ service: "web" check
✓ 3 other passing checks

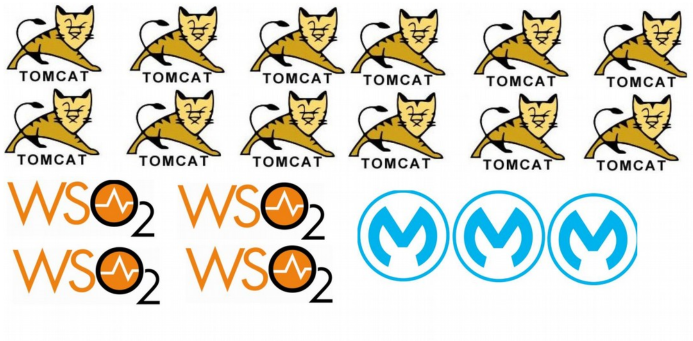
consul-client-7
10.0.1.32
☒ service: "web" check
✓ 3 other passing check

Healthy Nodes

consul-client-0 ✓

consul-client-1 ✓

consul-client-2



Serverspec comes to rescue

The home page https://serverspec.org/ describes core resource types:
bond | bridge | cgroup | command | cron | default_gateway | docker_container | docker_image | file | group | host | iis_app_pool | iis_website | interface | ip6tables | ipfilter | ipnat | iptables | kernel_module | linux_audit_system | linux_kernel_parameter | lxc | mail_alias | mysql_config | package | php_config | port | ppa | process | routing_table | selinux | selinux_module | service | user | x509_certificate | x509_private_key | windows_feature | windows_registry_key | yumrepo | zfs

Code hosted on github in mizzy/serverspec, mizzy/specinfra ,vvchik/vagrant-serverspec, covers 20+ operating systems

A very similar inspec/inspec framework exists for Chef.

A handful of active projects present extended types created for popular app stack spec like npm, ELK etc.

Extreme serverspec

The engine responsible for the serverspec execution resembles that of Puppet or Chef: source is sent from developer to the target node to be eventually converted into target OS specific low level commands to execute - result is sent back to the developer. Unlike provision the serverspec is executed for its direct, not side effects. Both serverspec and Puppet has significant amount of code wrapping the actual command in some custom DSL, however a plain Exec/Command class is still available.

In the extreme case in the body of a Ruby spec Command, one could find a full source code of a java class that would be compiled and run in the target node to load and examine some cryptic JDBC, or ELK configuration changes applied in the course of node provision:

There isn’t any ‘spy’ facilities for server spec or unit test developments, neither are any in pure Ruby, Java or .Net, nor there is any ‘recording’ environment. To help new developers learn and quickly adopt to server spec follow clear *Rspec*/*Cucumber* semantics:

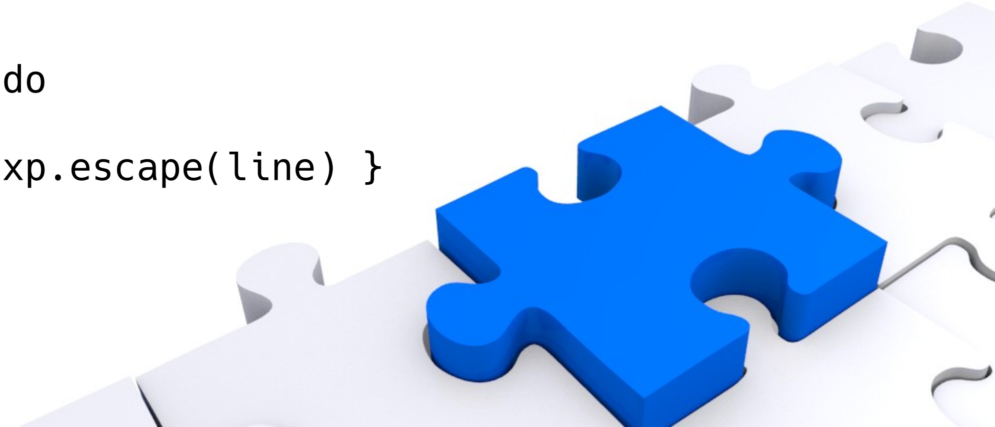
```
describe service(‘tomcat’) do
  it {should be_running }
end

describe port(8443) do
  it { should be_listening.with(‘tcp’) }
end

{'linux-kstat' => '0.1.3' }.each do |package_name, package_version|
   describe package(package_name) do
     it { should be_installed.by('gem')
                   .with_version(package_version)  }
    end
  end
```

It is shorter then the underlying command

```
describe command("/bin/gem list --local #{package_name}") do
  its(:stderr) { should be_empty }
  its(:exit_status) {should eq 0 }
  its(:stdout) { should match Regexp.new(package_version) }
end
```

A slightly less magical expectation

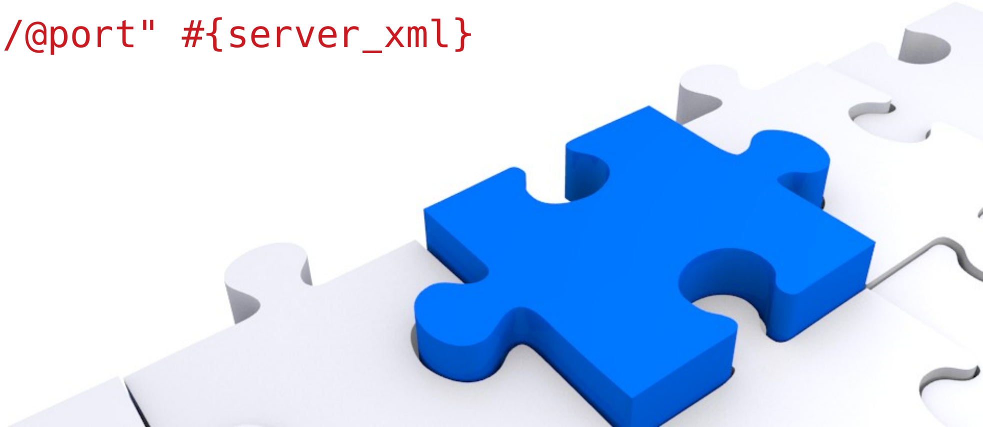
```
context 'Virtual Host settings' do
  describe file('/etc/httpd/conf.d/vhost.conf') do
    [ 'ProxyRequests Off', ... ].each do |line|
      its(:content) { should match "^\s*" + Regexp.escape(line) }
    end
  end
end
```



Eventually the *command* is where the tricky part is:

```
describe command('curl -k -I http://localhost') do
  its(:stdout) { should match /Server: Apache\/\d\.\d+\.\d+ (:?Unix|CentOS)/i }
end
```

```
context 'Tomcat shutdown port' do
  server_xml = "#{catalina_home}/conf/server.xml"
  describe command(<<-EOF
   xmllint --xpath "/Server[@shutdown='SHUTDOWN']/@port" #{server_xml}
  EOF
  ) do
    its(:exit_status) { should eq 0 }
    its(:stdout) { should match 'port="-1"' }
  end
end
```



https://tomcat.apache.org/tomcat-8.5-doc/appdev/web.xml.txt

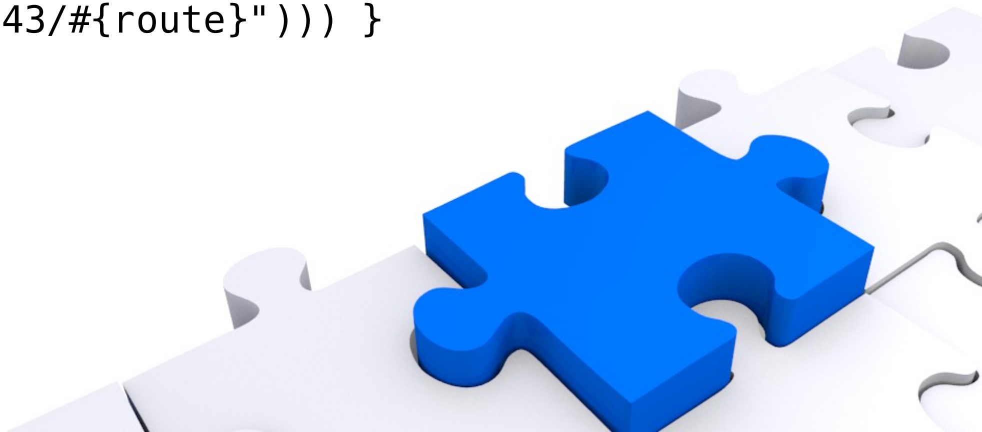
Gradually the *command* itself could become tricky but reusable (Ruby or libxml2 used to focus on specific XML node):

```
describe 'redirect port 8080' do
  doc = Document.new(content)
  result = REXML::XPath.first(doc, "/Server/Service/Connector[@port = \"8080\"]
/@redirectPort").value
  it { result.should match '8443' }
end

context 'Tomcat servlet configuration' do
  class_name = 'com.mycompany.mypackage.ControllerServlet'
  describe command(<<-EOF
   xmllint --xpath "//*[local-name()='servlet']/*[local-name()='servlet-class']/text()"
#{web_xml}
  EOF
  ) do
    its(:stdout) { should match Regexp.new(class_name) }
 end
```

https://tomcat.apache.org/tomcat-8.5-doc/appdev/web.xml.txt

Eventually the *command* is where the tricky part is (jq used to focus into the specific node of the JSON configuration):

```
context 'Consul service health check configuration' do
 {
   'myservice' => 'api/health'
 }.each  do |service, route|
    describe command("jq '.service.checks[].http' < '/etc/consul.d/#{service}.json'") do
        let(:path) { '/bin:/usr/bin:/usr/local/bin'}
        its(:stdout) { should match( Regexp.new(
              Regexp.escape("https://127.0.0.1:8443/#{route}"))) }
    end
  end
end
```



https://tomcat.apache.org/tomcat-8.5-doc/appdev/web.xml.txt

On Windows, Powershell and C# is used to retrieve obscure information about .Net

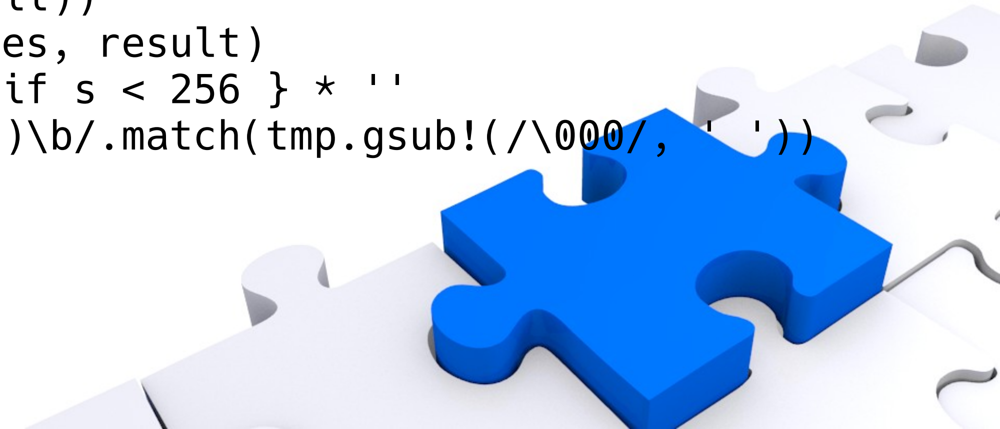
```
context 'specific assembly in GAC' do
  assembly_name = 'WindowsFormsIntegration'
  token = '31bf3856ad364e35'
  describe command(<<-EOF
  $result = @()
    [Object].Assembly.GetType('Microsoft.Win32.Fusion').GetMethod('ReadCache')
    .Invoke($null, @([Collections.ArrayList]$result, '#{assembly_name}', [UInt32]2 ))
  $result
  EOF
  )
  do
    its(:stdout) { should contain
     "#{assembly_name}, Version=3.0.0.0, Culture=neutral, PublicKeyToken=#{token}" }
  end
end
```

Any passing *serverspec expectation* may easily become a Puppet *fact*:

```
if Facter.value(:kernel) == 'windows'
  Facter.add('version') do
    begin
      script = "([xml](get-content -path 'version.xml')).Info.Product.version"
      command = "powershell.exe -executionpolicy remotesigned -command \" &
{ #{script} }\""
      result = Facter::Util::Resolution.exec(command)
    rescue => ex
      $stderr.puts ex.to_s
    end
  end
end
```

Serverspec derived fact could replace native Ruby Puppet *fact*:

```
Facter.add('version') do
  extend FFI::Library
  ffi_lib 'version.dll'
  attach_function :resource_size,:GetFileVersionInfoSizeA [:ptr, :ptr ], :int
  attach_function :version, :GetFileVersionInfoA, [ :ptr,:int,:int,:buf_out], :int
  version_information = '\VarFileInfo\Translation'.encode('UTF-16LE')
  result = ' ' * (resource_size(filepath, nil))
  status = version(filename, 0, size_in_bytes, result)
  tmp = result.unpack('v*').map{ |s| s.chr if s < 256 } * ''
  version_match = /FileVersion\s+\b([0-9.]+)\b/.match(tmp.gsub!(/\000/, ' '))
  version_match[1].to_s
end
```



Serverspec expectation could be used for consul *“script-kind” service check* similar to `/etc/consul.d/mongodb.json`:/etc/consul.d/mongodb.json

```
{
  "service": {
    "name": "mongo-db",
    "tags": ["mongo"],
    "address": "192.168.31.02",
    "port": 27017,
    "checks": [
      {
        "name": "Checking MongoDB"
        "script": "/usr/bin/check_mongo.py --host 192.168.31.02 --port 27017",
        "interval": "5s"
      }

    ]
}
```

https://www.consul.io/docs/agent/checks.html

The *http-kind service check* heartbeat-response is provided by Spring after annotation is provided and dependency added to `pom.xml`:

```
@SpringBootApplication
@EnableDiscoveryClient
public class DiscoverableApplication {
  public static void main(String[] args) {
    SpringApplication.run(DiscoverableApplication.class, args);
  }
}

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Vagrant Serverspec provisoner

serverspec vagrant provisioner is part of the Vagrant flow, a little bit of disadvantage so its `rake spec` is from a deep stack of Ruby calls

elementary tasks like `$DEBUG = ENV.fetch('DEBUG', false)` become a bit problematic

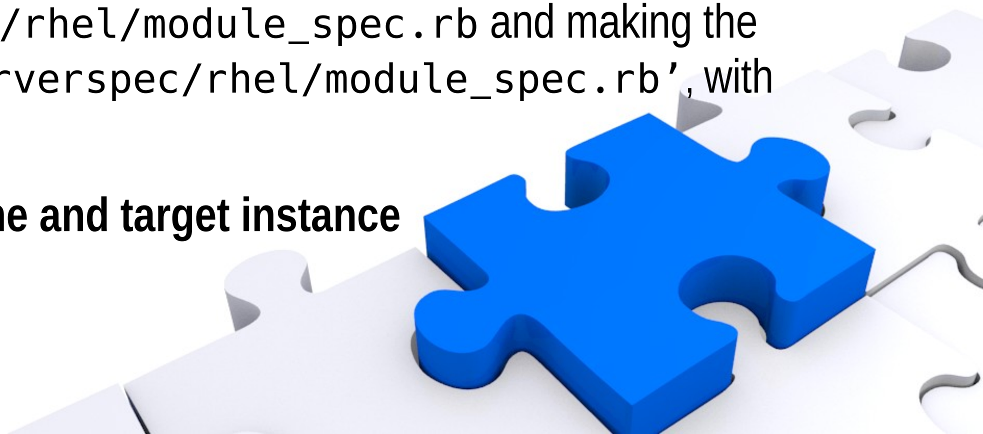
serverspec is scheduled afterprovision and rerun is time consuming - not really when module is idempotent

with default settings error stack is super extra verbose

spec file (`node_spec.rb`) is not visible to therefore can not be produced by Puppet module - solvable through relative reference placing under `files/serverspec/rhel/module_spec.rb` and making the legacy one simply `require_relative ‘../../files/serverspec/rhel/module_spec.rb’`, with the actual path determined by workspace directory layout

assumes the availability of ssh between developer machine and target instance

which may change during secure environment provision



https://tomcat.apache.org/tomcat-8.5-doc/appdev/web.xml.txt

Serverspec through own Puppet module

serverspec being just a handful of text files plus a Ruby runtime – calls to be provisioned (rvm-hosted) through Puppet from archive and templates and an exec for `rake spec` on the instance then updates Puppet and Vagrant logs with the result. This remediates limitations

`rake spec` is directly in console and can be run explicitly after provision and the spec file edited in the instance. Debugging is easy.

Spec file is generated by Puppet from template, hieradata etc. for version-sensitive portion
(one can also keep serverspec require relative for Vagrant runs)

Runs on DMZ machine after lockdown, the results pushed to the developer, CICD etc.

A little cumbersome to modify file locally and push to the vm to validate

https://tomcat.apache.org/tomcat-8.5-doc/appdev/web.xml.txt

Puppet-RSpec

- Stubs target OS, environment facts, module parameters and hiera data
- Compiles and examines the 'Catalog'
- Asserts the specified actions are taken
- All that without requiring one to spawn the real instance
- Real provision will behave according to those specs

http://rspec-puppet.com/

Puppet Rspec is useful with
module and profile development and refactoring
intelligent upgrade / downgrade logic is critical (present|latest|absent)
with complex module logic or generation of complex configurations

https://github.com/voxpupuli/puppet-logrotate

Questions?