

# Serverspec in cloud provision

Serguei Kouzmine  
kouzmine\_serguei@yahoo.com



# Serverspec in cloud provision

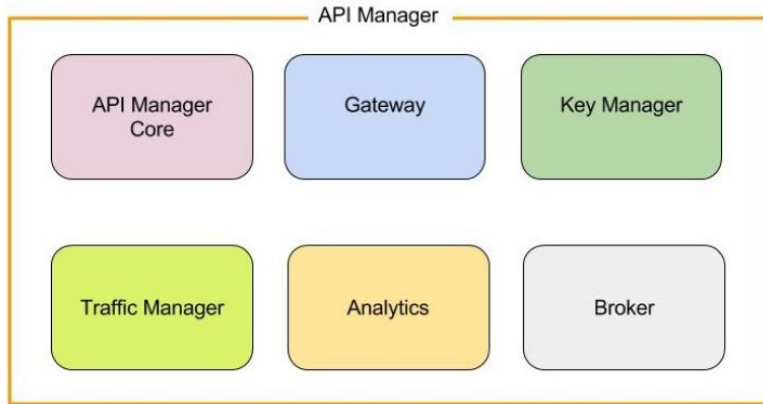
Successful operation automation usually needs, along with core tool technical skills some knowledge about the system under test - not strictly but is recommended. For example, (no)SQL, Angular would be a great help in web testing, HTML / Javascript no longer sufficient .

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

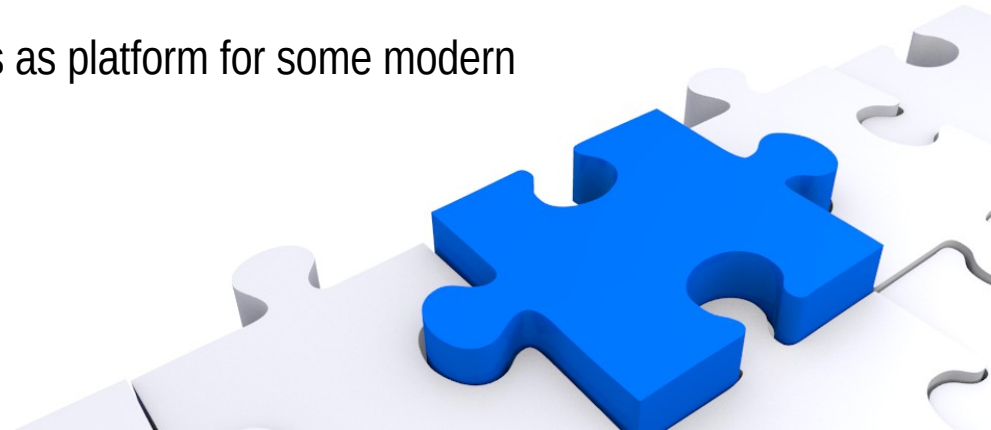
QA engineer using Selenium or Katalon often is or willing to grow as Web developer but unlikely ever interested in the code base of those tools.



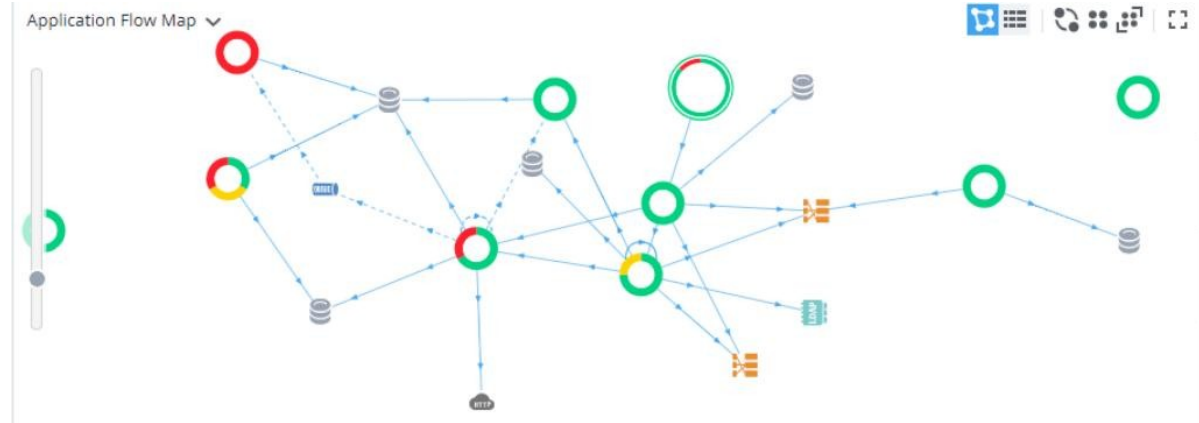
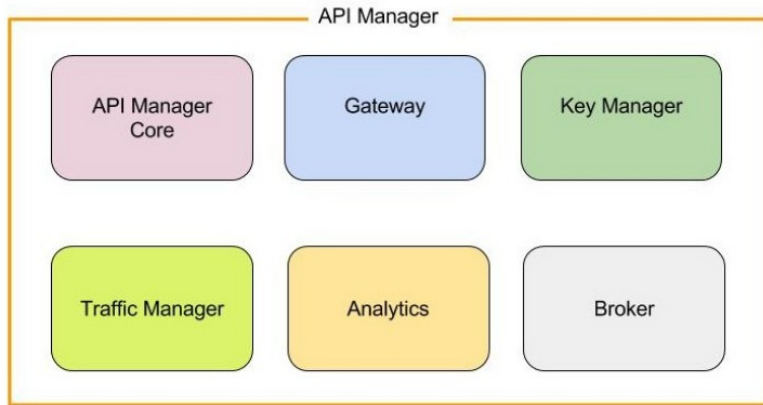
# Serverspec in cloud provision



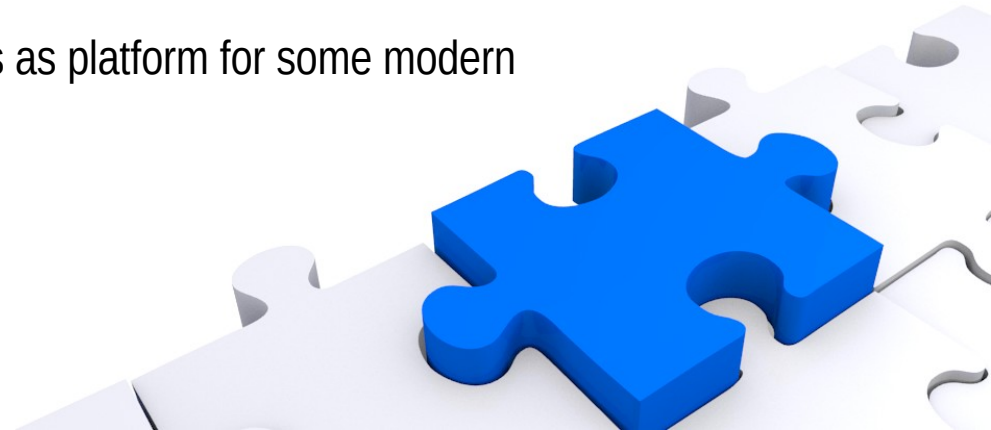
Cloud cluster provisioned by some automation workflow serves as platform for some modern application stack



# Serverspec in cloud provision

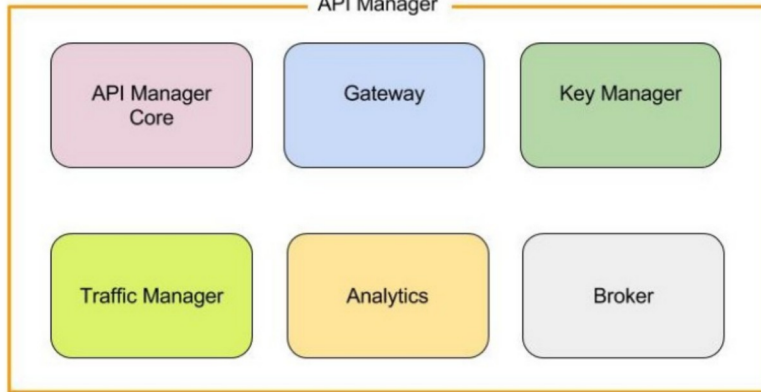


Cloud cluster provisioned by some automation workflow serves as platform for some modern application stack

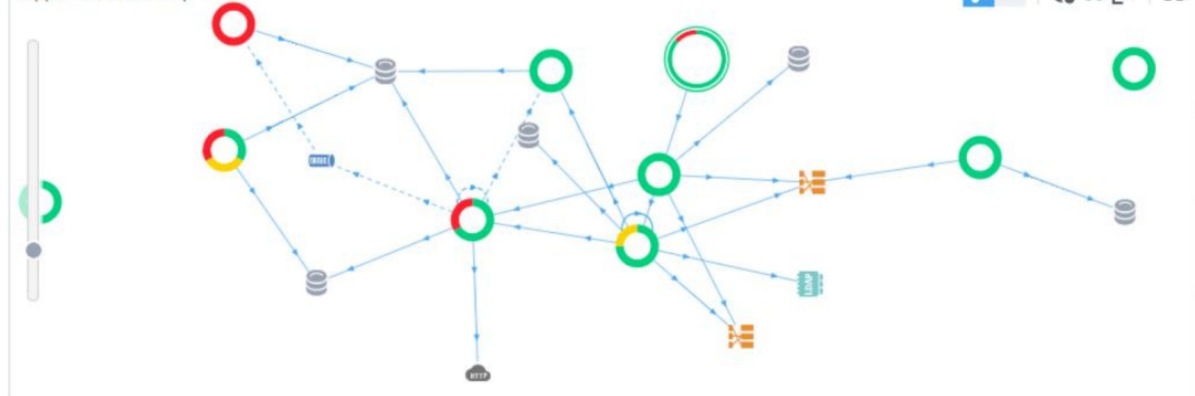


# Serverspec in cloud provision

API Manager



Application Flow Map



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

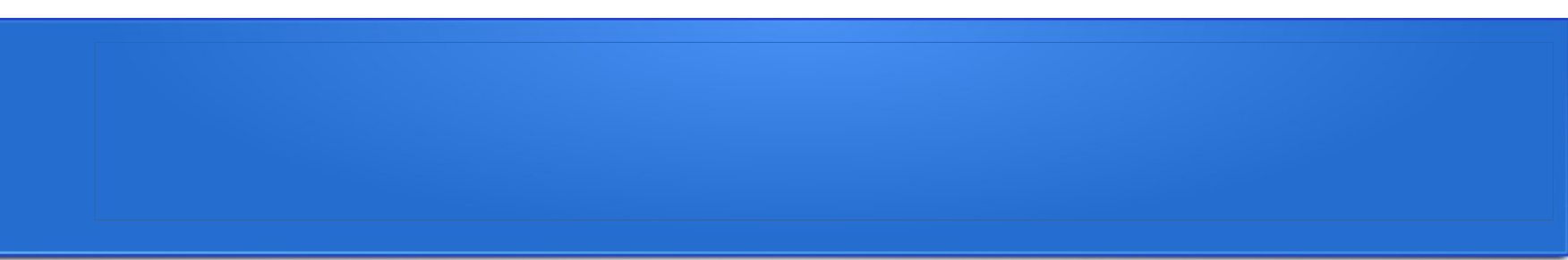
WSO2 Machine Learner

WSO2 Message Broker

WSO2 Private PaaS

WSO2 Storage Server



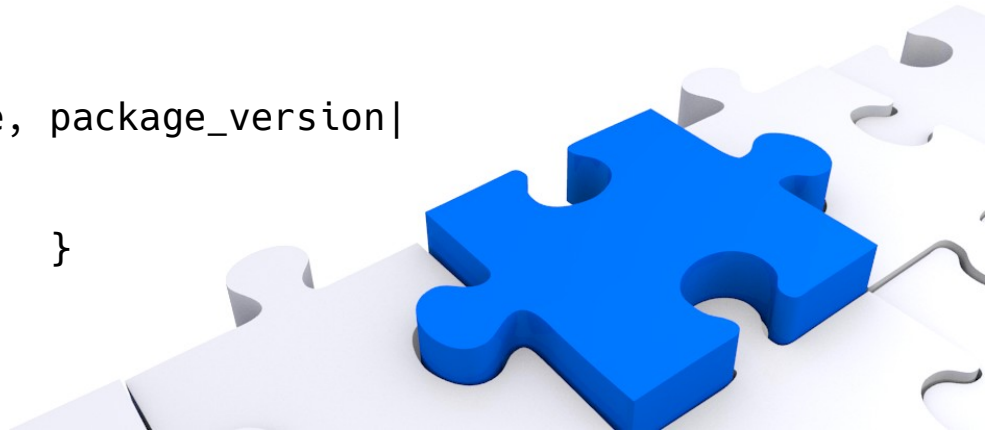


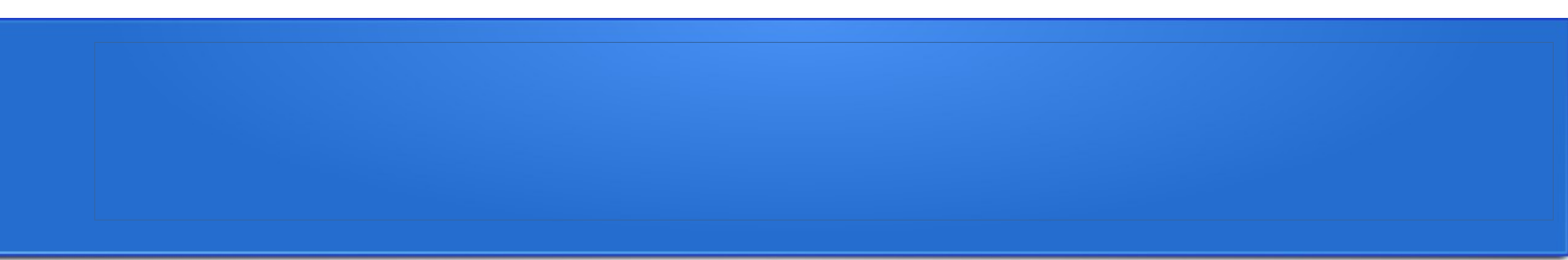
There isn't any 'spy' facilities for server spec or unit test developments, neither are any in pure Ruby, Java or .Net, nor there is any 'recording' environment. To help new developers learn and quickly adopt to server spec follow clear *Rspec/Cucumber* semantics:

```
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
```



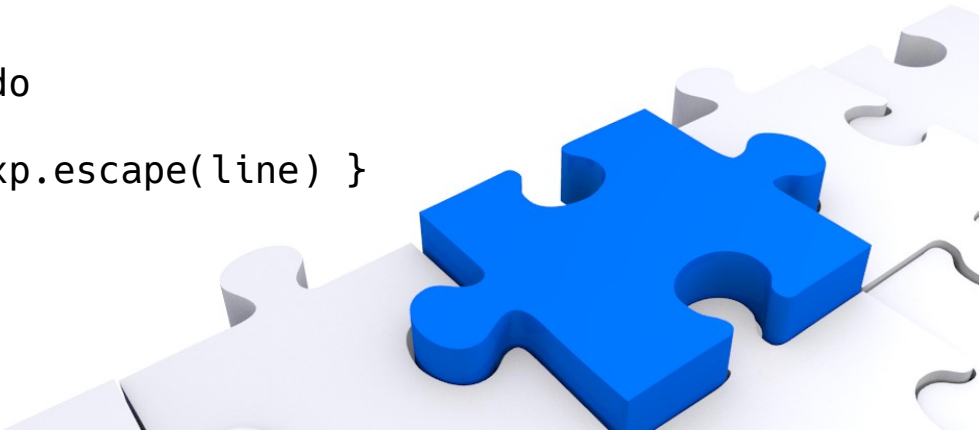


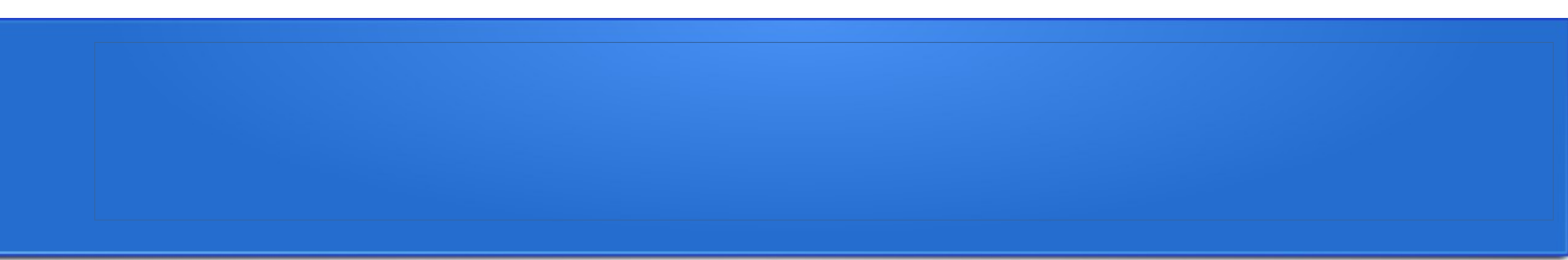
It is shorter than the underlying command

```
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
```



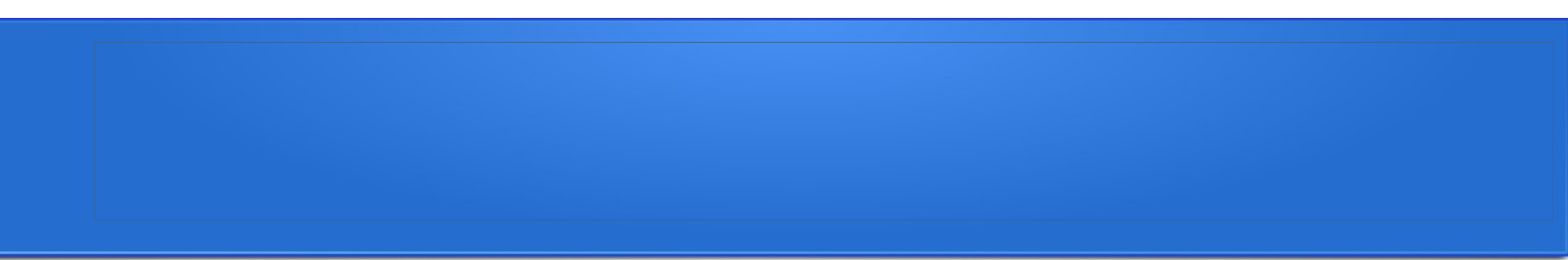


Eventually the *command* is where the tricky part is:

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

context 'Tomcat web.xml 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(:exit_status) { should eq 0 }
    its(:stdout) { should match Regexp.new(class_name) }
  end
end
```





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

```
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
```

# Serverspec in cloud provision

