Serverspec and RSpec Cheat Sheet

Assertions

- should means "assert true".
- should_not means "assert false".

Resources

http://serverspec.org/resource_types.html

file

Files, directories, and devices.

```
describe file('/path/to/file') do

  #INode type
  it { should be_file }       # or be_directory,
or be_symlink
  it { should exist }

  #Permissions
  it { should be_owned_by 'username' }
  it { should be_grouped_into 'groupname' }

  it { should be_mode 440 }

  it { should be_readable }               # or
be_writable or be_executable
  it { should be_readable.by('owner') }   # or
'group' or 'others'
  it { should be_readable.by_user('username') }

  #Links
  if { should be_linked_to '/path/to/target' }

  #Contents
  its(:md5sum)    { should eq '...' }    # or, and
rspec matcher
  its(:sha256sum) { should eq '...' }
  its(:size)      { should < 1024 }
  its(:content)   { should match /some pattern/ }

end
```

user

System users.

```
describe user('username') do
  it { should exist }
  it { should belong_to_group 'group' }
  it { should have_home_directory '/home/username'
}
  it { should have_login_shell '/bin/bash' }
  it { should have_authorized_key 'ssh-rsa ABCD...
user@hostname' }
end
```

group

System user groups.

```
describe group('groupname') do
  it { should exist }
end
```

SeverSpec cheat sheet courtesy of:
https://gist.github.com/mearns/a86286eace104c89c012

packages

Software packages installed on the system.

```
describe package('httpd') do
  it { should be_installed }
end
```

Select based on operating system:

```
describe package('httpd'), :if => os[:family] ==
'redhat' do
  it { should be_installed }
end

describe package('apache2'), :if => os[:family] ==
'ubuntu' do
  it { should be_installed }
end
```

port

Network ports

```
describe port(80) do
  it { should be_listening }
end
```

service

Installed services.

```
describe service('httpd') do
  it { should be_running }
  it { should be_enabled }      # enabled to start
when the OS boots.
end
```

process

Currently running processes.

```
describe process("memchached") do
  it { should be_running }

  # parameters from ps, see `man ps(1)`, under
"STANDARD FORMAT SPECIFIERS"
  its(:user) { should eq 'root' }
  its(:args) { should match /-c 32000\b/ }
  its(:nice) { should > 10 }
end
```

command

Run arbitrary commands and check the results.

```
describe command('ls -al /') do
  its(:stdout) { should match /some pattern/ }
  its(:stderr) { should be_empty }
  its(:exit_status) { should eq 0 }
end
```

host

Hosts on the network

```
describe host('example.org') do
  it { should be_resolvable }

  # address
  its(:ipaddress) { should match /
192\.168\.10\.10/ }     # could be IPv4 or IPv6
  its(:ipv4_address) { ... }
  its(:ipv6_address) { ... }

  # reachability
  it { should be_reachable }          # ping
  it { should be_reachable.with(
        :port => 53,                  # required
parameter
        # Optional params (default values shown)
        :proto => 'tcp',              # or 'udp'
        :timeout => 5                 # in seconds.
  )}
end
```

Matchers (rspec)

https://www.relishapp.com/rspec/rspec-expectations/v/3-4/docs/built-in-matchers
To try these outside of a serverspec test, you'll need to require 'RSpec', and
replace describe with RSpec.describe.

For strings

```
describe 'foobar' do
  it { should eq 'foobar' }         # match using
== operator
  it { should match /ooba/ }        # match using
regex, anywhere in string.
  it { should_not match /^ooba$/ }  # anchor regex
  it { should_not be_empty }        # test for
empty string: ""

  it { should start_with('fo') }
  it { should end_with('bar') }

  it { should be_a(String) }
end
```

For numbers

```
describe 10 do
  it { should eq 10 }
  it { should == 10 }             # same as above.
  it { should < 20 }
  it { should <= 10 }
  it { should > 0 }
  it { should >= 9 }

  it { should be_within(2).of(9) }
  it { should be_within(2).of(11) }

  it { should be_a(Numeric) }     # also consider:
Float, Integer
  it { should be_an_instance_of(Fixnum) } # Direct
class, no higher
end
```

For arrays

```
describe [1, 2, 3] do
  it { should_not be_empty }      # test for empty
list: []

  it { should include(2) }        # membership
test
  it { should_not include(0) }

  it { should all( be_an(Integer) ) }
# apply matcher to all elements
  it { should all( be_an(Integer).and be < 10 ) }
# conjunction

  it { should start_with([1,2]) }
  it { should end_with([2,3]) }

  it { should be_an(Array) }
end
```

For hashes

```
describe ({ :a => 'A', :b => 2 }) do
  it { should have_key(:a) }
  it { should include(:a) }       # same as above
  it { should include(:b => 2) }  # test for
presence of key and the value it maps to.
  it { should_not include(:b => 3) }
  it { should_not include(:c => 2) }

  it { should be_a(Hash) }
end
```

The general purpose satisfy matcher

```
describe(3.14159) do
  # Passes as long as the block returns true.
  it { should satisfy { |uut|
    uut.kind_of?(Numeric) and uut > 3 and uut <
3.2
  }}
end
```

Combining expectations (and, or)

```
describe("thunder") do
  it { should start_with("thun").and
end_with("der") }
  it { should start_with("won").or
start_with("thun") }
  it { should (start_with("won").or
start_with("thun")).and end_with("der") }

  # with line breaks
  it { should (
      start_with("won").or \
      start_with("thun").or \
      start_with("pon")
    ).and (
      end_with("der")
    )
  }
end
```

Defining custom matchers

```
RSpec::Matchers.define :be_multiple_of do |
expected|
  match do |actual|
    actual % expected == 0
  end

  # optional, override description
  description do
    "be multiple of #{expected}"
  end

  # optionally, override failure message:
  failure_message do |actual|
    "expected that #{actual} would be a multiple
of #{expected}"
  end

  # optionally, override failure message when
negated:
  failure_message_when_negated do |actual|
    "expected that #{actual} would not be a
multiple of #{expected}"
  end
end
Example:
describe(9) do
  it { should be_multiple_of(3) }

  #Deliberate failures
  it { should be_multiple_of(2) }
  it { should_not be_multiple_of(3) }
end
```

SeverSpec cheat sheet courtesy of:
https://gist.github.com/mearns/a86286eace104c89c012