InSpec for Integration Testing

LAB

Simple Web Server Cookbook

- ❑ The Apache web server should be running
- ❑ The Apache web server should be listening on the default port
- ❑ The Apache web server should be configured to start on boot

CHEF

Create a directory for cookbooks

```
$ mkdir -p ~/cookbooks
```

Move into the directory for cookbooks

```
$ cd ~/cookbooks
```

Generate an apache cookbook

```
$ chef generate cookbook apache
```

```
Generating cookbook apache
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. Type `cd apache` to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.
Type `delivery local --help` to see a full list.

Why not start by writing a test? Tests for the default recipe are stored at:

test/smoke/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:

recipes/default.rb
```

Move into the apache cookbook's directory

```
$ cd ~/cookbooks/apache
```

```
Generating cookbook apache
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. Type `cd apache` to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.
Type `delivery local --help` to see a full list.

Why not start by writing a test? Tests for the default recipe are stored at:

test/smoke/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:

recipes/default.rb
```

Remember…

Infrastructure policies need testing

- ↳ Linting
- ↳ Static Analysis
- ↳ Unit Testing
- ↳ Integration Testing
- ↳ Compliance Testing

“Infrastructure as Code” should be tested like ANY other codebase.

Remember…

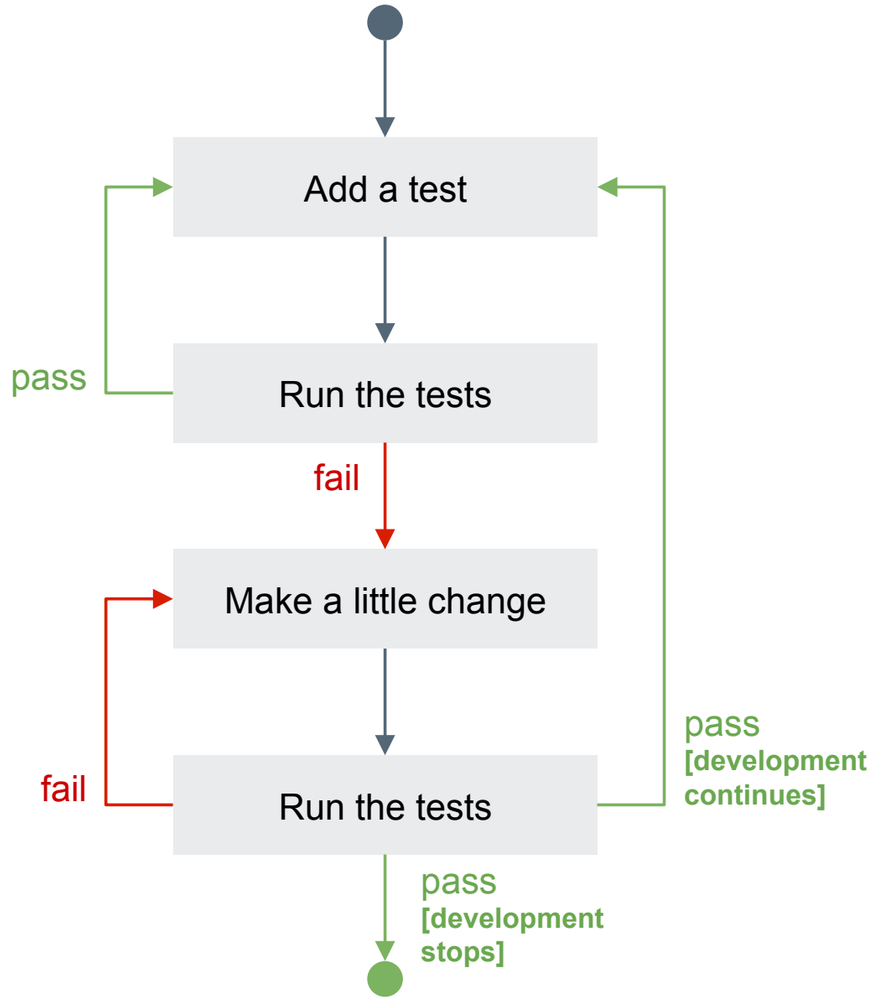
Infrastructure policies need testing

- ↳ Linting
- ↳ Static Analysis
- ↳ Unit Testing
- ↳ Integration Testing
- ↳ Compliance Testing

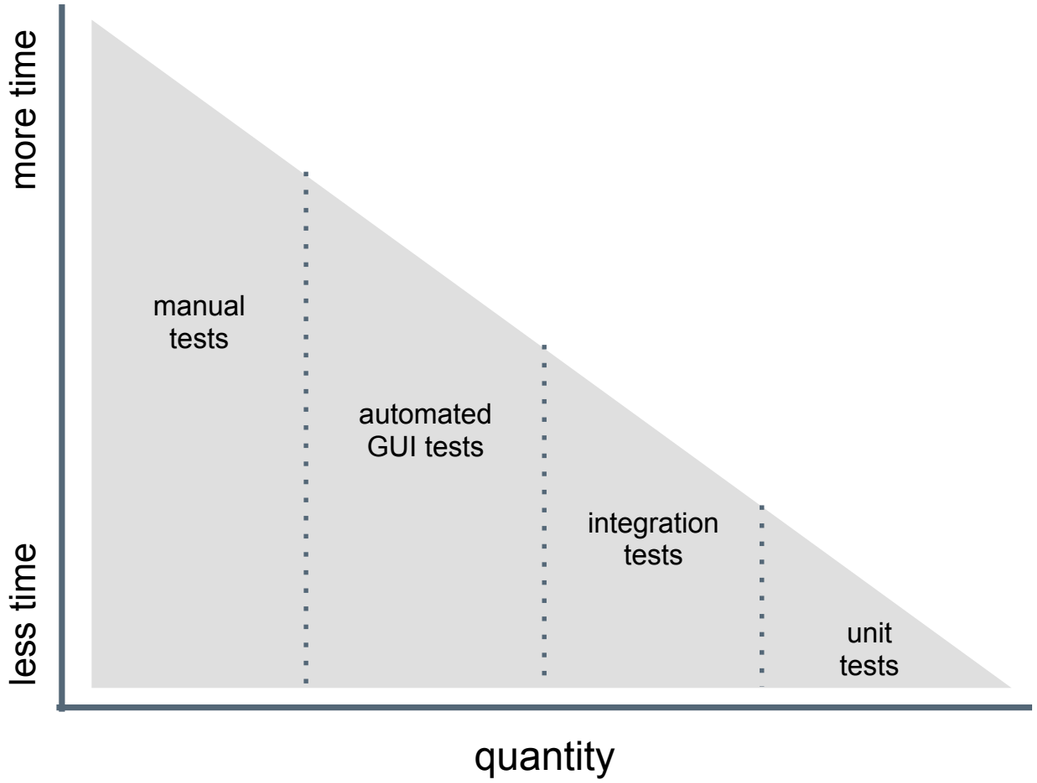
“Infrastructure as Code” should be tested like ANY other codebase.

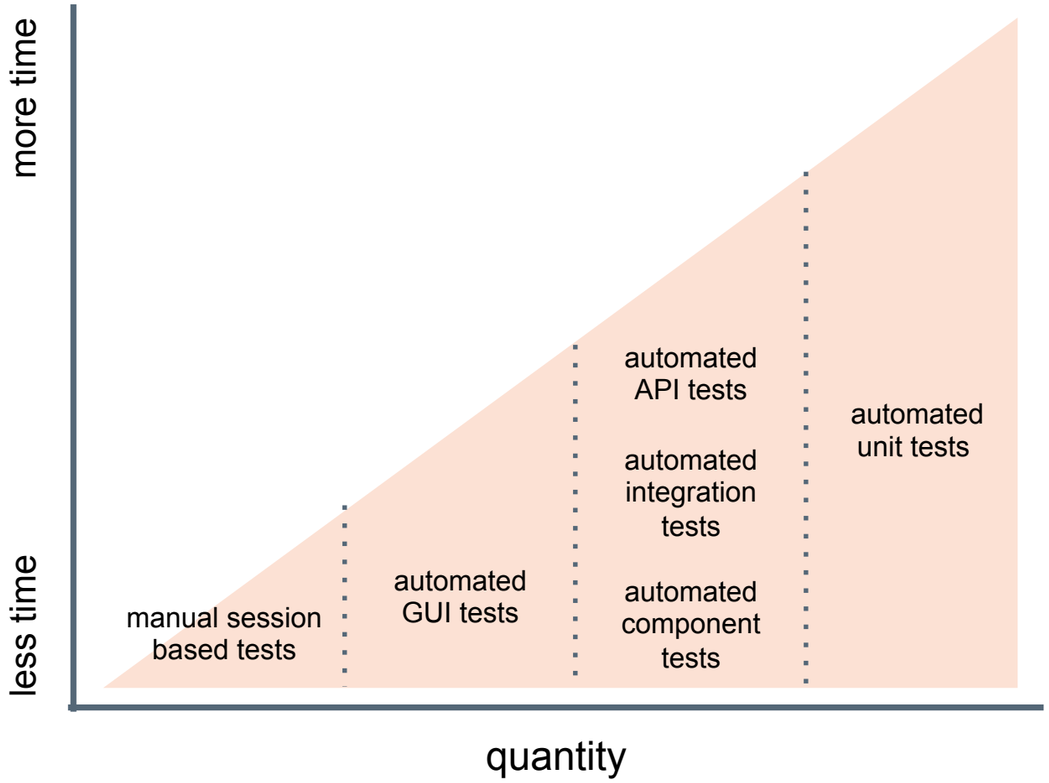
Test-driven Development

- Write a test, watch it fail
- Write some code
- Write and run more tests
- Code review
- Delivery pipeline to production
- Lowered chance of production failure



Software Testing and Why it Matters





Testing **builds safety** through feedback loops

Inexpensive experiments to provide validation

Reduces risk

Optimize Testing: Do more of the inexpensive testing first!

What can delivery local do?

```
$ delivery local --help
```

```
delivery-local
Run Delivery phases on your local workstation.

USAGE:
    delivery local [FLAGS] <phase>

FLAGS:
    -h, --help               Prints help information
        --no-spinner         Disable the spinner
        --non-interactive    Disable cli interactions
    -V, --version            Prints version information

ARGS:
    <phase>    Delivery phase to execute [values: unit, lint, syntax, provision, deploy, smoke, cleanup]
```

Test Kitchen

kitchen list
kitchen create
kitchen converge
kitchen verify
kitchen destroy
kitchen test
kitchen login

Delivery

[No matching delivery command]
delivery local provision
delivery local deploy
delivery local smoke
delivery local cleanup
[No matching delivery command]
[No matching delivery command]

~/cookbooks/apache/.delivery/project.toml

Update the Kitchen Configuration

~/cookbooks/apache/.kitchen.yml

```
---
driver:
  name: vagrant
  name: docker
  use_sudo: false
```

Update the Kitchen Configuration

~/cookbooks/apache/.kitchen.yml

```
platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
  - name: centos-6.8
```

Final .kitchen.yml

~/cookbooks/apache/.kitchen.yml

```
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero
  # You may wish to disable always updating cookbooks in CI or other testing environments.
  # For example:
  #   always_update_cookbooks: <%= !ENV['CI'] %>
  always_update_cookbooks: true

verifier:
  name: inspec

platforms:
  - name: centos-6.8

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    verifier:
      inspec_tests:
        - test/smoke/default
    attributes:
```

List Kitchen Instances

```
$ kitchen list
```

```
Instance             Driver   Provisioner  Verifier  Transport  Last Action    Last Error
default-centos-68    Docker   ChefZero     Inspec    Ssh        <Not Created>  <None>
```

Run the integration tests

```
$ delivery local smoke
```

```
Chef Delivery
Running Smoke Phase
-----> Starting Kitchen (v1.14.2)
-----> Creating <default-centos-68>...
       Sending build context to Docker daemon 193.5 kB
       Step 1 : FROM centos:centos7
 ...
-----> Verifying <default-centos-68>...
       Loaded

Target:  ssh://kitchen@localhost:32768


  User root
     ✔  should exist
     ↻  This is an example test, replace with your own test.
  Port 80
     ✔  should not be listening
     ↻  This is an example test, replace with your own test.

Test Summary: 2 successful, 0 failures, 2 skipped
       Finished verifying <default-centos-68> (0m0.45s).
-----> Kitchen is finished. (1m4.44s)
```

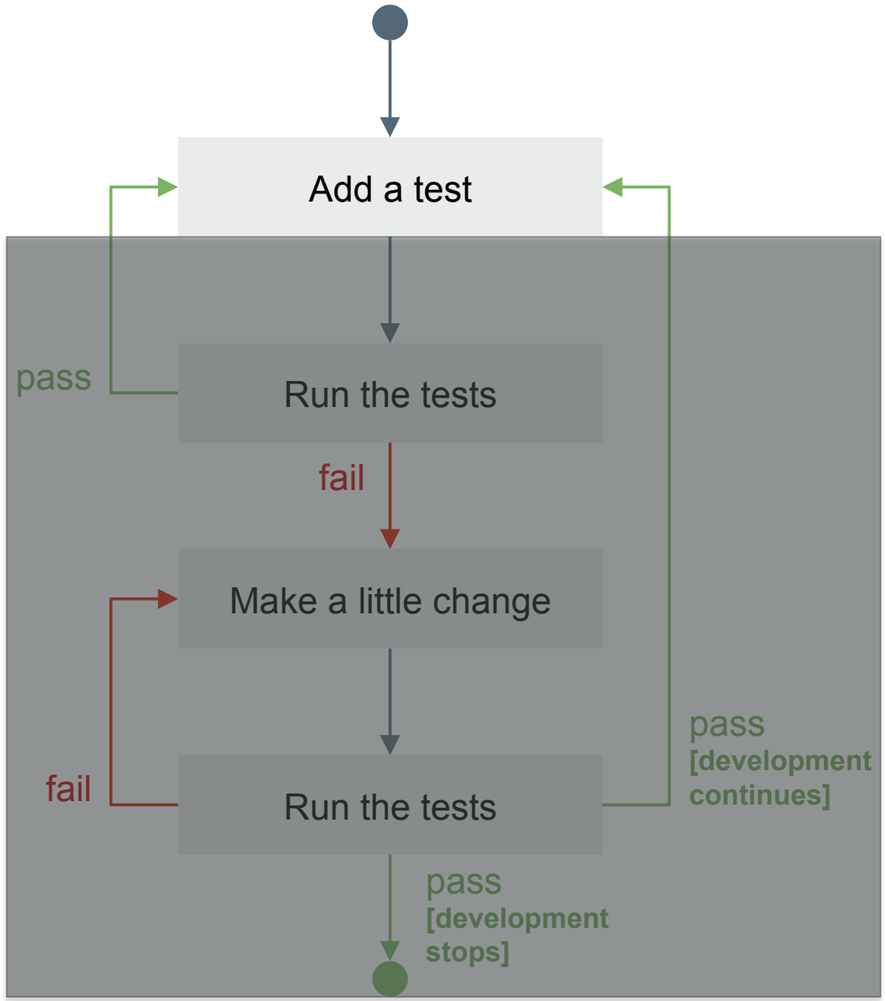

CHEF

Integration Testing – Add tests

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

describe service 'httpd' do
  it { should be_running }
  it { should be_enabled }
end

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



Resource: package

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

where

```
('name') must specify the name of a package, such as 'nginx'
be_installed is a valid matcher for this resource
```

Use the package resource to test if the named package and/or package version is installed on the system.

Resource: service

```
describe service('service_name') do
  it { should be_installed }
  it { should be_enabled }
  it { should be_running }
end
```

Use the service resource to test if the named service is installed, running and/or enabled.

Resource: port

```
describe port(514) do
  it { should be_listening }
end
```

Use the port InSpec audit resource to test basic port properties, such as port, process, if it's listening.

Run the Integration Test

```
$ delivery local smoke
```

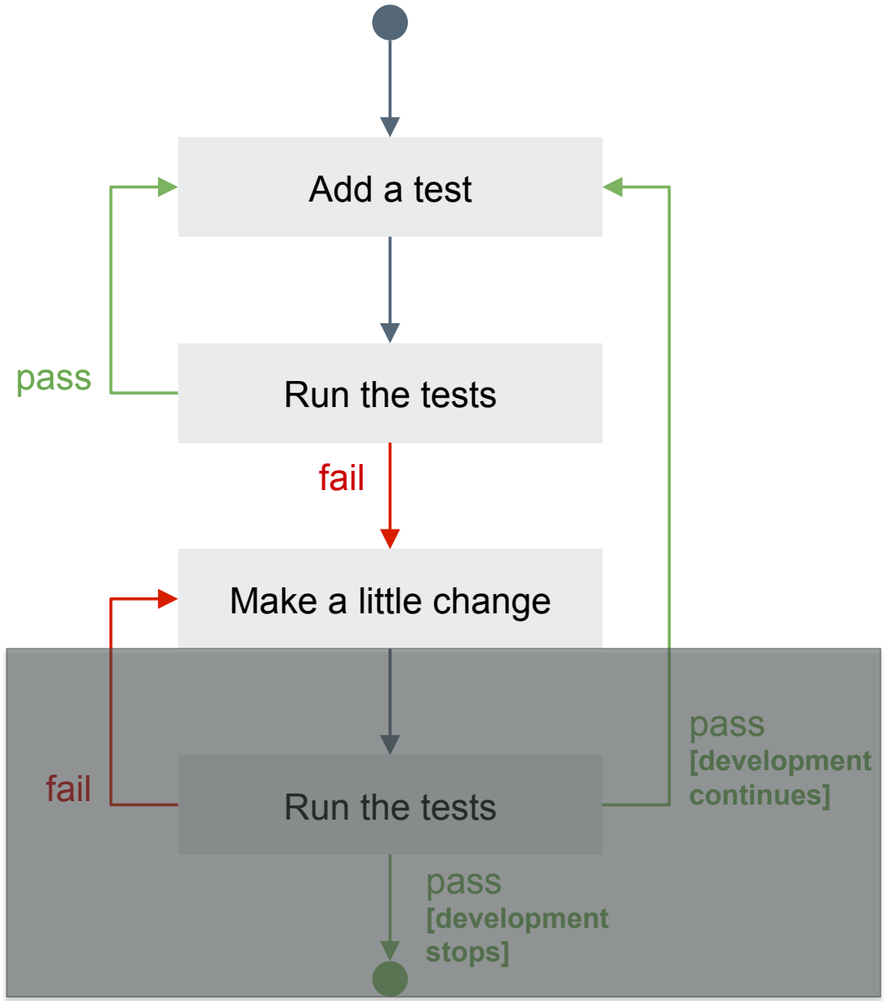
```
System Package
   Ø  httpd should be installed
   expected that `System Package httpd` is installed
Service httpd
   Ø  should be running
   expected that `Service httpd` is running
   Ø  should be enabled
   expected that `Service httpd` is enabled
Port 80
   Ø  should be listening
   expected `Port 80.listening?` to return true, got false

Test Summary: 0 successful, 4 failures, 0 skipped
```

Integration Testing – Make a change

```
package 'httpd' do
  action :install
end

service 'httpd' do
  action [ :start, :enable ]
end
```



Apply the change

```
$ delivery local deploy
```

```
Chef Delivery
Running Deploy Phase
-----> Starting Kitchen (v1.14.2)
-----> Converging <default-centos-68>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 5.2.0...
...
Recipe: apache::default
 * yum_package[httpd] action install
   - install version 2.2.15-56.el6.centos.3 of package httpd
 * service[httpd] action start
   - start service service[httpd]
 * service[httpd] action enable
   - enable service service[httpd]

Running handlers:
Running handlers complete
Chef Client finished, 3/3 resources updated in 06 seconds
Finished converging <default-centos-68> (0m11.54s).
-----> Kitchen is finished. (0m12.42s)
```

Run the Integration Test

```
$ delivery local smoke
```

```
System Package
   ✔  httpd should be installed
Service httpd
   ✔  should be running
   ✔  should be enabled
Port 80
   ✔  should be listening

Test Summary: 4 successful, 0 failures, 0 skipped
     Finished verifying <default-centos-68> (0m0.58s).
-----> Kitchen is finished. (0m1.46s)
```

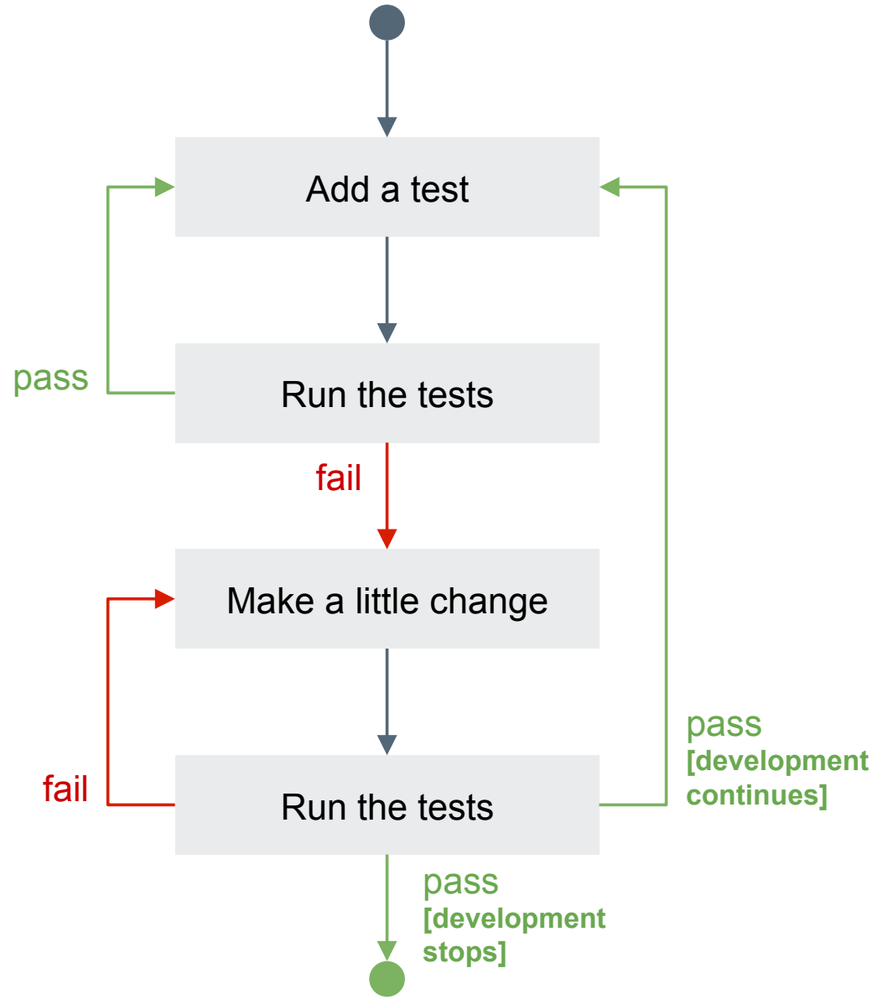
Integration Testing – Run the tests

```
-----> Verifying <default-centos-72>...
       Loaded

Target:  ssh://vagrant@127.0.0.1:2222


  System Package
     ✔  httpd should be installed
  Service httpd
     ✔  should be running
     ✔  should be enabled
  Port 80
     ✔  should be listening

Test Summary: 4 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-72> (0m0.91s).
```



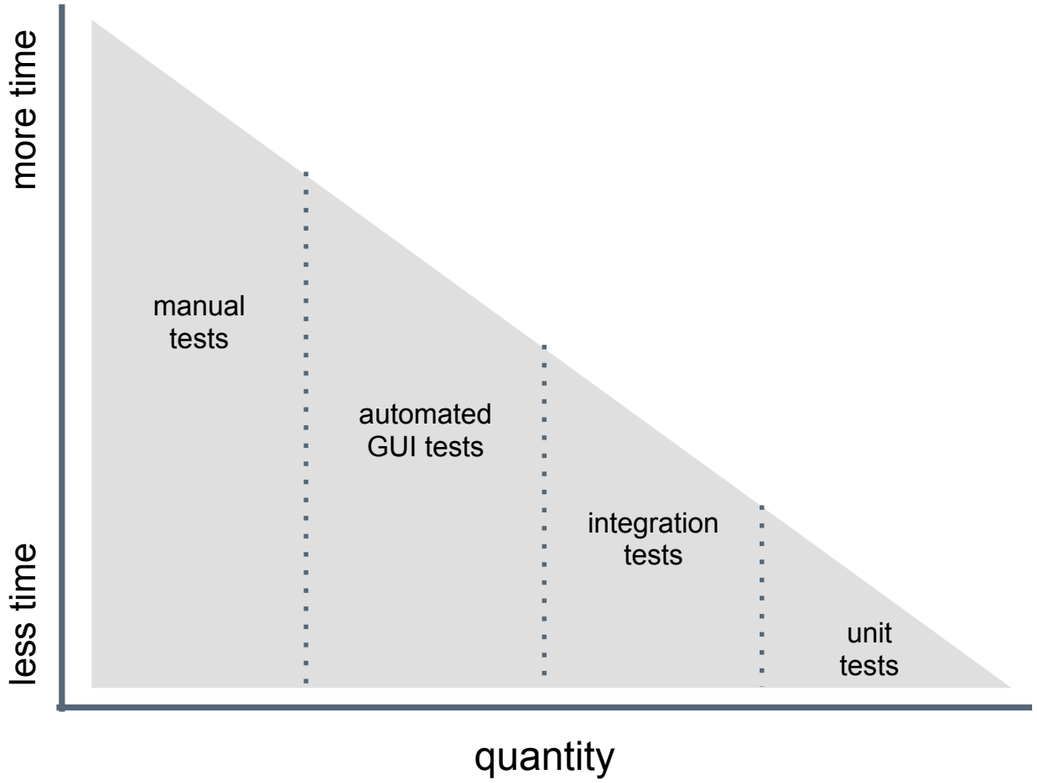
LAB

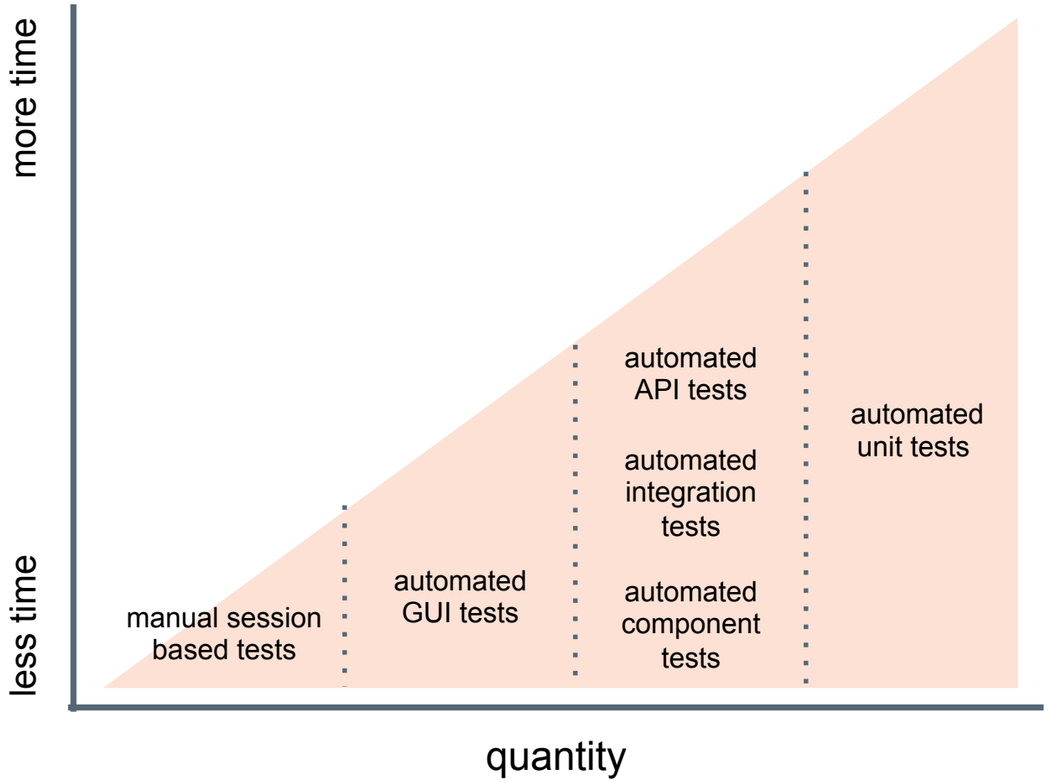
Simple Web Server Cookbook

- ✓ The Apache web server should be running
- ✓ The Apache web server should be listening on the default port
- ✓ The Apache web server should be configured to start on boot

CHEF

Software Testing and Why it Matters





Testing **builds safety** through feedback loops

Inexpensive experiments to provide validation

Reduces risk

Optimize Testing: Do more of the inexpensive testing first!

DISCUSSION

Additional Testing

What command would you use to complete lint testing?
What command would you use to complete syntax testing?
What command would you run to complete unit tests?
What additional kitchen-related commands should we run?
How would you InSpec exec your tests over the docker protocol?

Objectives

- ✓ Execute an InSpec test on a local machine
- ✓ Execute an InSpec test on a remote machine
- ✓ Generate an InSpec profile
- ✓ Add InSpec-based integration test to a Chef cookbook
- ✓ Run InSpec-based integrations tests during Chef cookbook development

List additional resources and places to look for support with InSpec