



BDD in the SUSE Manager team

Behavior-Driven Development process

Óscar Barrios
QA Engineer

What are these slides about?

- **BDD** (Behaviour-Driven Development)
- **Gherkin**
- **Cucumber**
- **SUSE Manager Test Framework**
- **Capbara**
- **Jenkins CI Test Suite**

Behavior-Driven Development

Requirement = User Story + Acceptance Criteria

- **User Story**

- Natural language **description of a feature**
- Written from the perspective of an end user
- A common description can follow that format:
 - As a *[role]* I want *[feature]*, so that *[benefit]*

- **Acceptance criteria**

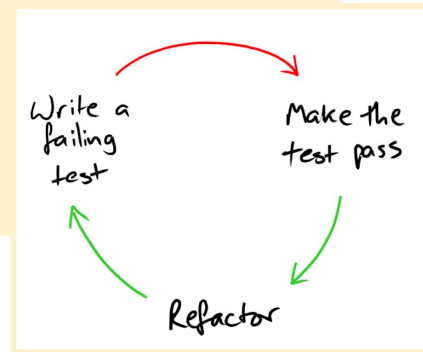
- **Validates** that we accomplished that **user story**
- A common acceptance criteria can follow that format:
 - Given *[initial context]*, when *[event occurs]*, then *[ensure an outcome]*

Behavior-Driven Development

- BDD is a software development process
 - Was made to approach **business interests** and **technical insight**
 - Using **natural language constructs** (English-like sentences), allow us to express the **behavior** and the **expected outcomes**
 - Through a simple DSL (Domain-specific language) it converts natural language **statements into executable tests**

Result: Map an **acceptance criteria in a DSL** and validate the functionality at the same time that we develop it

- *Mocking* the implementation at start and iterate through it
- We will validate functionality for each change of the code



Driving design and implementation from scenarios

Feature: Login Profile

As an employee of the company
I want to login using my credentials
So that I can collaborate with my colleagues

Background: User navigates to Company home page

Given I am on the "Company home" page on URL "www.mycompany.com"

Then I should see "Log In as Employee" message

Scenario: Successful login

When I fill in "Username" with "Test"

And I fill in "Password" with "123"

And I click on the "Log In" button

Then I am on the "My profile" page on URL "www.mycompany.com/myprofile"

And I should see "Welcome to your profile" message

And I should see the "Log out" button

Define User Story and
its acceptance criteria

Write executable
acceptance criteria

New business requirement

Acceptance tests

Design feature

Develop feature

Break feature into
scenarios

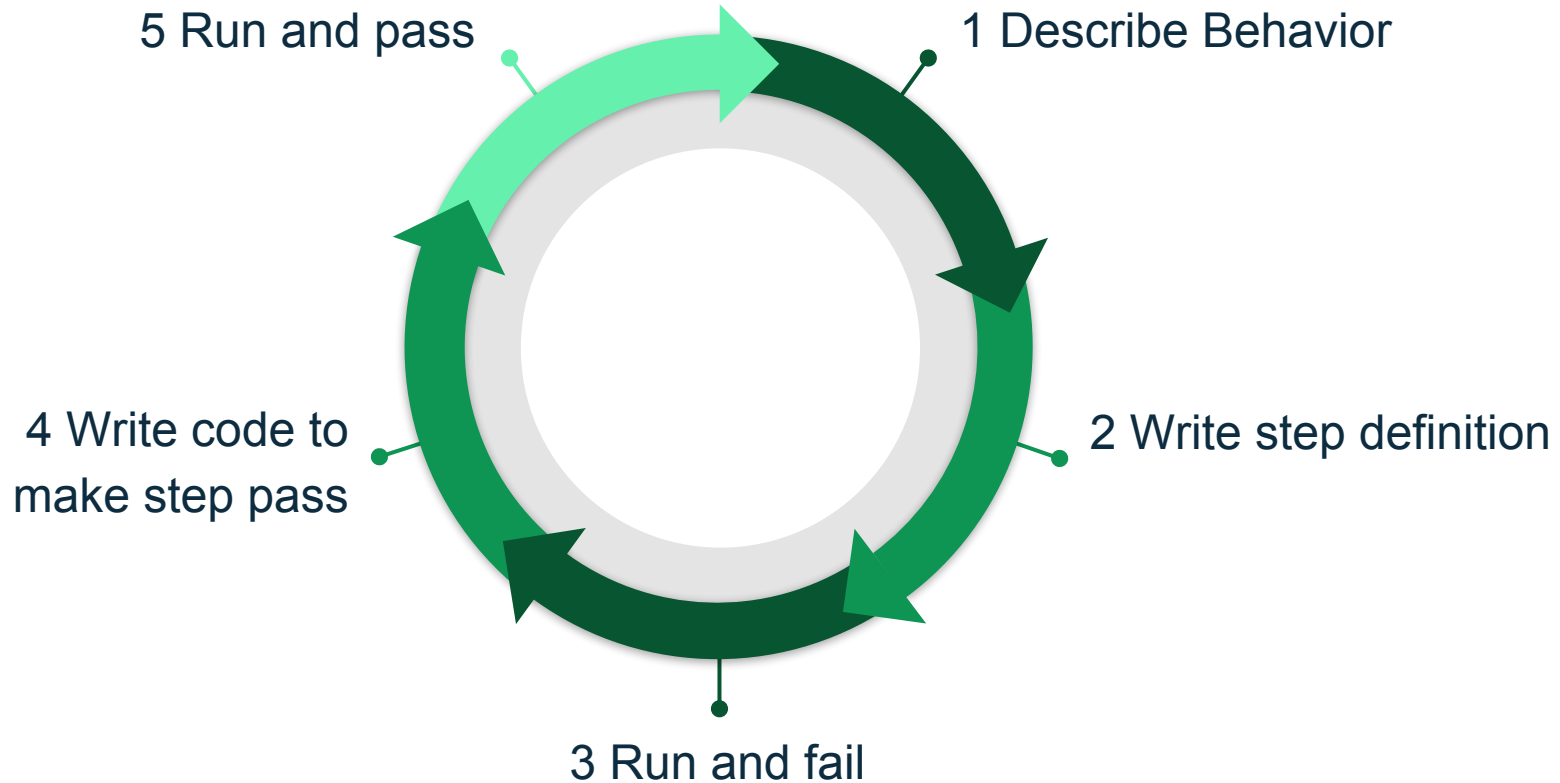
Iterate through the acceptance
tests to develop code

Feature: Login Profile

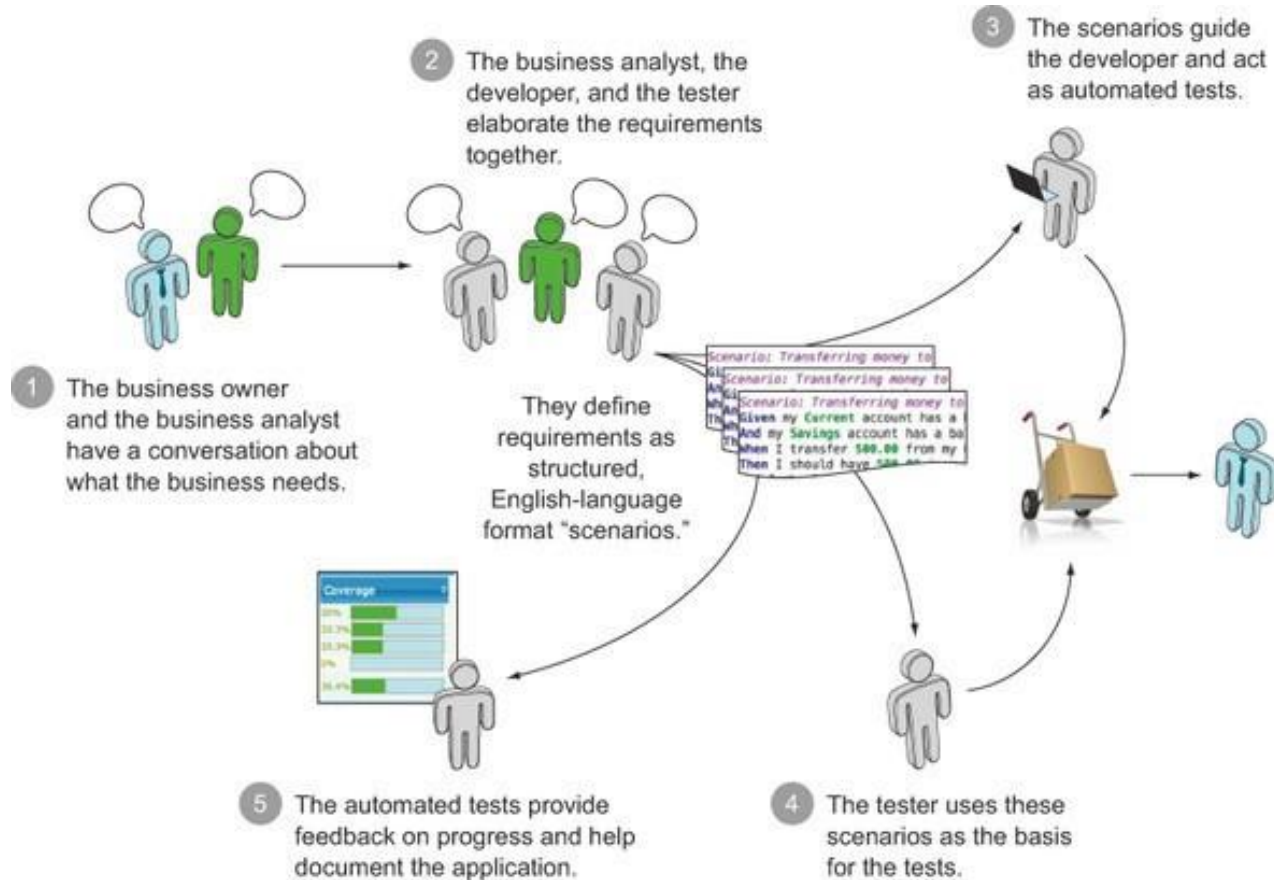
Scenario: Successful login

Scenario: Failed login using wrong credentials

Develop a feature using BDD



How BDD works in a team?



Write the tests after finish implementation

Test after is not BDD

Many people write tests after the code is written. This is not BDD or TDD, because the tests do not *drive* the implementation when they are written afterwards.

TDD/BDD is not about testing

A common misunderstanding of TDD and BDD is that they are testing techniques. They're not. As the name suggests, TDD and BDD are about software *development*.

It is the process of approaching your design and forcing you to think about the desired outcome and API before you code.

Automated tests are a byproduct of TDD and BDD.

From <https://docs.cucumber.io/bdd/overview/>

Gherkin

The Gherkin language

- Is a **Domain-Specific Language** business readable
- Is designed to be **non-technical** and **human readable**
- **Describes** how the system behaves
- The Gherkin language was designed for the purpose of:
 - Improving/Simplifying/Automating documentation
 - Automated tests as executable acceptance criteria

The Gherkin language

- Uses indentation to define structure
- Has special keywords and support multiple languages:
 - *Feature*
 - *Background*
 - *Scenario*
 - *Scenario Outline*
 - *Given*
 - *When*
 - *Then*
 - *And*
 - *But*
 - ...

Gherkin syntax - Structure

Feature: Description of the user story (aka feature)

As a *[role]*

I want *[feature]*

So that *[benefit]*

Scenario: Description of an acceptance criteria for that user story

Given *[initial context]*

And some other precondition

When *[event occurs]*

Then *[ensure an outcome]*

But some other postcondition will not happen

Scenario: Description of another acceptance criteria

....

Gherkin syntax - Features


- Each feature is described in a .feature file
- We will start the file with the keyword **Feature:** and its description
- The feature can contain **multiple scenarios**, where we define the acceptance criteria for that feature
- In addition to basic scenarios, a feature may contain **scenario outlines** (scenarios iterating with different inputs) and **backgrounds** (preconditions for all the scenarios)

Gherkin syntax - Scenarios

- Each scenario starts with the keyword **Scenario:**
- Describes an acceptance criteria
- A scenario **contains one or more steps**

Example:

```
Scenario: Create admin user and first organization
  Given I access the host the first time
  When I go to the home page
  And I enter "SUSE Test" as "orgName"
  And I enter "admin" as "login"
  And I enter "admin" as "password"
  And I click on "Create Organization"
  Then I am logged in
```



Gherkin syntax - Steps

- The purpose of **Given** steps is to **put the system in a known state** before the user starts interacting with the system
 - *Given the database is clean*
- The purpose of **When** steps is to **describe the key action** the user performs
 - *When I press "login" button*
- The purpose of **Then** steps is to **observe outcomes**. The observations should inspect the output of the system from a user perspective (UI, Report, command output, ...)
 - *Then the output should be "Login Successful"*
- If you have several Given, When or Then steps, you can use the step **And** or **But**, so the scenario can be read more fluently

Gherkin syntax - Background

- Allows you to **add a context to all the scenarios** of the current feature
- Is run before each of your scenarios

Example:

Background:

Given I am authorized as "admin" with password "admin"

When I follow "Software > Channels > All" in the left menu

Gherkin syntax - Scenario Outline

- **Rid of copy-pasted scenarios** to use different arguments
- Provides a **template using placeholders** and an “Examples” section
- The scenario it will be executed for each line of the example section

Example:

```
Scenario Outline: Eating
  Given there are <start> cucumbers
  When I eat <eat> cucumbers
  Then I should have <left> cucumbers
```

Examples:

	start		eat		left	
	12		5		7	
	20		5		15	

Gherkin syntax - Scenario Outline

Scenario: Create an activation key with a channel and a package list for x86_64

```
Given I am on the Systems page
When I follow "Activation Keys" in the left menu
And I follow "Create Key"
And I enter "SUSE Test PKG Key x86_64" as "description"
And I enter "SUSE-PKG-x86_64" as "key"
And I enter "20" as "usageLimit"
And I select "Test-Channel-x86_64" from "selectedBaseChannel"
And I click on "Create Activation Key"
And I follow "Packages"
And I enter "man" as "packages"
And I click on "Update Activation Key"
Then I should see a "Activation key SUSE Test PKG Key x86_64" text
And I should see a "Details" link
And I should see a "Packages" link
And I should see a "Configuration" link in the content area
And I should see a "Groups" link
And I should see a "Activated Systems" link
```

Scenario: Create an activation key with a channel and a package list for i586

```
Given I am on the Systems page
When I follow "Activation Keys" in the left menu
And I follow "Create Key"
And I enter "SUSE Test PKG Key i586" as "description"
And I enter "SUSE-PKG-i586" as "key"
And I enter "20" as "usageLimit"
And I select "Test-Channel-i586" from "selectedBaseChannel"
And I click on "Create Activation Key"
And I follow "Packages"
And I enter "man" as "packages"
And I click on "Update Activation Key"
Then I should see a "Activation key SUSE Test PKG Key i586 has been modified." text
And I should see a "Details" link
And I should see a "Packages" link
And I should see a "Configuration" link in the content area
And I should see a "Groups" link
And I should see a "Activated Systems" link
```

Gherkin syntax - Scenario Outline

```
Scenario Outline: Create an activation key with a channel and a package list for <architecture>
  Given I am on the Systems page
  When I follow "Activation Keys" in the left menu
  And I follow "Create Key"
  And I enter "SUSE Test PKG Key <arch>" as "description"
  And I enter "SUSE-PKG-<architecture>" as "key"
  And I enter "20" as "usageLimit"
  And I select "Test-Channel-<architecture>" from "selectedBaseChannel"
  And I click on "Create Activation Key"
  And I follow "Packages"
  And I enter "man" as "packages"
  And I click on "Update Activation Key"
  Then I should see a "Activation key SUSE Test PKG Key <architecture> has been modified." text
  And I should see a "Details" link
  And I should see a "Packages" link
  And I should see a "Configuration" link in the content area
  And I should see a "Groups" link
  And I should see a "Activated Systems" link
  Examples:
    | architecture |
    | x86_64       |
    | i586         |
```


The Gherkin syntax

- Allows multiple arguments
 - *When I write "Jhon" in "Name" text field*
- To better organize, it allows *tags* and you can filter by them

```
@billing
```

```
Feature: Verify billing
```

```
  Scenario: Missing product description
```

The Gherkin syntax

- Accepts **tables** to specify larger data set

Scenario:

Given the following people exist:

name	email	phone
Aslak	aslak@email.com	123
Joe	joe@email.com	234
Bryan	bryan@email.org	456

- Accepts **multiline strings** (pystrings)

Scenario:

Given a blog post named "Random" with:

"""

Some Title

=====

Here is the first paragraph of my blog post.

"""

Automate tests with a BDD Framework

cucumber

Ruby

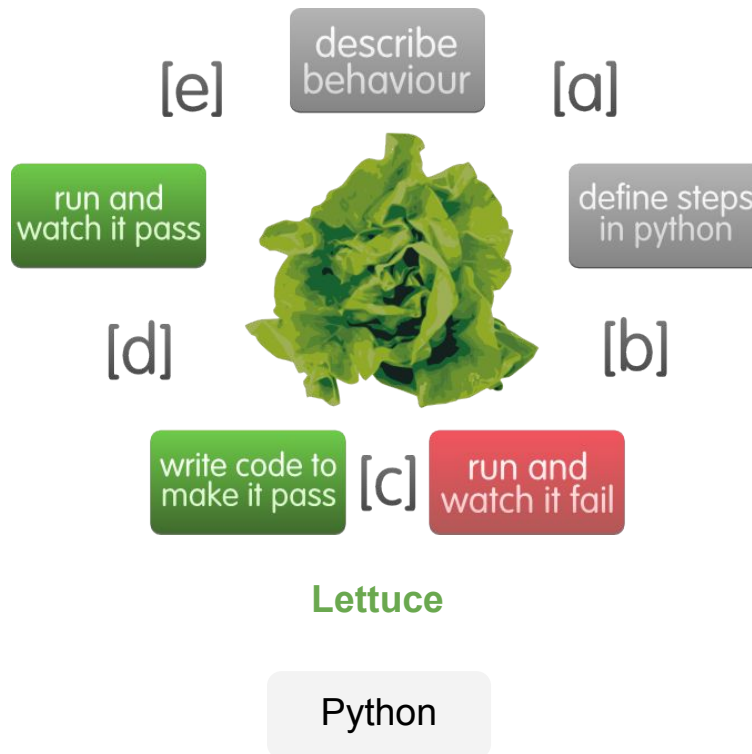
jbehave

Java

Jasmine

Javascript

And a lot more frameworks!

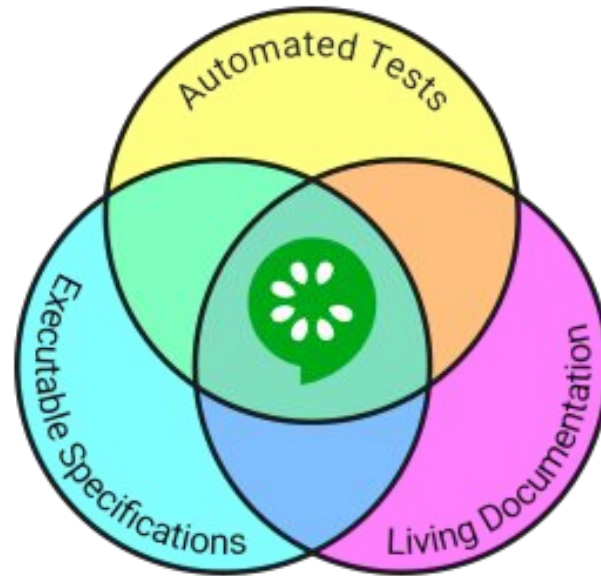


Cucumber

Cucumber

- Is a tool that supports Behaviour-Driven Development(BDD)
- It parse Gherkin syntax to map step definitions into methods in Ruby
- Verifies that the software conforms with the specification and generates a report indicating ✓ success or ✗ failure for each scenario.

Cucumber



Console commands

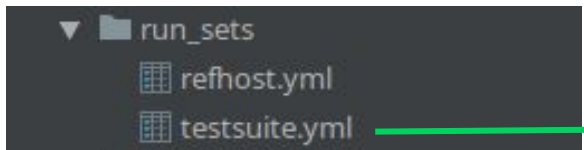
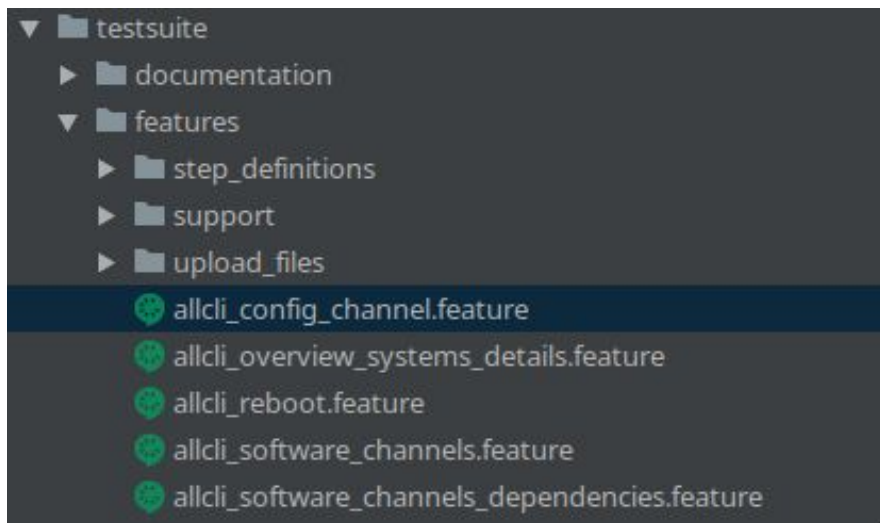
- Tell Cucumber to **only run scenarios** with a particular tag
 - `cucumber --tags "@smoke and @fast"`
- Tell Cucumber to **ignore scenarios** with a particular tag
 - `cucumber --tags "not @smoke"`
- Run a **concrete scenario** at a line of your .feature
 - `cucumber features/authenticate_user.feature:42`
- Run the scenario(s) filtered **by its name**
 - `cucumber features --name "Failed login"`

Report

- Cucumber supports a bunch of plugins to report the results
 - pretty : ASCII text with colors
 - html : Review the results in a web page
 - json : very useful if we need to parse and process the results
 - junit : useful to be combined with other tools like Jenkins

SUSE Manager Test Framework

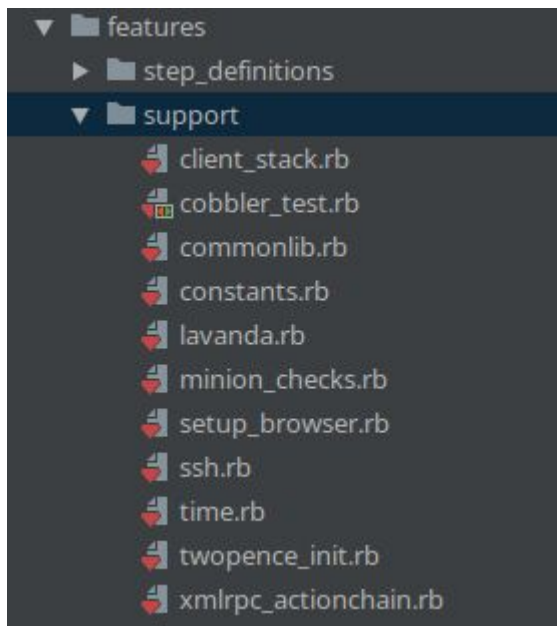
How is the test project organised?



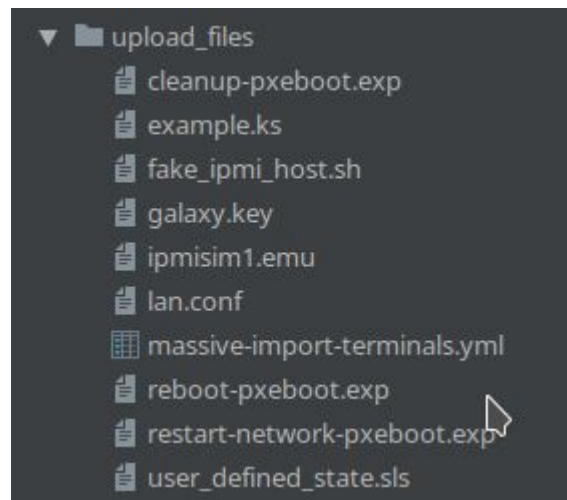
```
## Core features BEGIN ###  
  
# IMMUTABLE ORDER  
  
- features/core_first_settings.feature  
# initialize SUSE Manager server  
- features/core_srv_channels_add.feature  
- features/core_srv_push_package.feature  
- features/core_srv_create_repository.feature  
- features/core_srv_systemspage.feature  
...  
  
## Core features END ###  
  
## Secondary features BEGIN ##  
  
# IDEMPOTENT  
  
- features/allcli_reboot.feature  
- features/trad_config_channel.feature  
...  
- features/srv_notifications.feature  
- features/min_empty_system_profiles.feature  
## Secondary features END ##
```

How is the test project organised?

Framework setup and ruby support code



Files to be used during test execution



How is the test project organised?

```
Feature: Add a repository to a channel
  In order to distribute software to the clients
  As an authorized user
  I want to add a repository
  I want to add this repository to the base channel

Scenario: Add a test repository for x86_64
  Given I am authorized as "testing" with password "testing"
  When I follow "Home" in the left menu
  And I follow "Channels"
  And I follow "Manage Software Channels" in the left menu
  And I follow "Manage Repositories" in the left menu
  And I follow "Create Repository"
  And I enter "Test-Repository-x86_64" as "label"
  And I enter "http://localhost/pub/TestRepo/" as "url"
  And I click on "Create Repository"
  Then I should see a "Repository created successfully" text
  And I should see "metadataSigned" as checked

Scenario: Disable metadata check for the x86_64 test repository
  Given I am authorized as "testing" with password "testing"
  When I follow "Home" in the left menu
  And I follow "Channels"
  And I follow "Manage Software Channels" in the left menu
  And I follow "Manage Repositories" in the left menu
  And I follow "Test-Repository-x86_64"
  And I uncheck "metadataSigned"
  And I click on "Update Repository"
  Then I should see a "Repository updated successfully" text
  And I should see "metadataSigned" as unchecked

Scenario: Add the repository to the x86_64 channel
  Given I am authorized as "testing" with password "testing"
  When I follow "Home" in the left menu
  And I follow "Channels"
  And I follow "Manage Software Channels" in the left menu
  And I follow "Overview" in the left menu
  And I follow "Test-Channel-x86_64"
  And I follow "Repositories" in the content area
  And I select the "Test-Repository-x86_64" repo
  And I click on "Update Repositories"
  Then I should see a "Test-Channel-x86_64 repository information was successfully updated" text
```

Let's jump to the step definitions!

```
▼ testsuite
  ► documentation
  ▼ features
    ▼ step_definitions
      centos_tradclient.rb
      command_steps.rb
      common_steps.rb
      datepicker_steps.rb
      docker_steps.rb
      lock_packages_on_client.rb
      navigation_steps.rb
      salt_steps.rb
      security_steps.rb
      smdba_steps.rb
      xmlrpc_common.rb
```


How is the test project organised?

```
#
# Click on a button and confirm in alert box
When(/^I click on "([^"]*)" and confirm$/) do |arg1|
  accept_alert do
    step %(I click on "#{arg1}")
    sleep 1
  end
end

#
# Click on a link
#
When(/^I follow "([^"]*)"$/) do |text|
  click_link(text, wait: CLICK_TIMEOUT)
end

#
# Click on the first link
#
When(/^I follow first "([^"]*)"$/) do |text|
  click_link(text, wait: CLICK_TIMEOUT, match: :first)
end
```

- The **step definitions** are **methods** made using Ruby
- Each method maps a **natural language step** (DSL) using Cucumber Framework
- To **interact with the web** browser we use Capybara
- Is structured using different .rb files depending the scope of each method

Test Environment

- A simple test environment is composed by these machines:
 - **Server**
 - Includes a **SUSE Manager** instance (support SLES versions)
 - **Clients**
 - Server distribution of each **supported Operating System**
 - **Controller**
 - OpenSUSE instance where we are going to execute the test suite

Test Environment

- In order to automatically deploy a test environment we use **Sumaform**, that project use **Terraform** with some **Salt** configuration, allowing us to deploy and install all needed in a set of virtual machines



How we run the test suite?

- In order to run the testsuite we have included a command available from the **controller**, you only need to execute “**run-testsuite**” and the magic will happen!
- If you want to run only one feature, go ahead with a cucumber command

Capybara

Implementing the steps using Capybara



- It can mimic actions of real users **interacting with web-based applications**. It can receive pages, parse the HTML and submit forms.
 - It supports different web drivers, currently we are testing using **Selenium Web Driver** (with a **Chrome** Browser Driver)
- It allows us to navigate, find, click links and buttons, use checkboxes, text inputs, interact with modals, use XPath, CSS and selectors, ...

Capybara cheat sheet

Navigating

```
visit articles_path
```

Interacting with forms

```
attach_file 'Image', '/path/to/image.jpg'  
fill_in 'First Name', with: 'John'
```

```
check 'A checkbox'  
unchecked 'A checkbox'
```

```
choose 'A radio button'
```

```
select 'Option', from: 'Select box'  
unselect
```

Clicking links and buttons

```
click 'Link Text'  
click_button  
click_link
```

Limiting

```
within '.classname' do  
  click '...'  
end
```

```
within_fieldset :id do  
  ...  
end
```

From <https://devhints.io/capybara>

Capybara cheat sheet

Predicates

```
page.has_css?('.button')
expect(page).to have_css('.button')
page.should have_css('.button')
```

Positive

has_content?

has_css? (selector)

has_xpath? (path)

has_link? (selector)

has_button? (selector)

has_field? (selector)

has_checked_field? (selector)

has_table? (selector)

has_select? (selector)

Negative

has_no_content?

has_no_css?

has_no_xpath?

has_no_link?

has_no_button?

has_no_field?

has_unchecked_field?

has_no_table?

has_no_select?

In RSpec, these also map to matchers like `page.should have_content`.

Selectors

```
expect(page).to have_button('Save')
```

```
expect(page).to have_button('#submit')
```

```
expect(page).to have_button('//*[@id="submit"]')
```

The selector arguments can be text, CSS selector, or XPath expression.

RSpec assertions

```
page.has_button?('Save')
```

```
expect(page).to have_no_button('Save')
```

In RSpec, you can use `page.should` assertions.

About negatives

```
expect(page).to have_no_button('Save') # OK
```

```
expect(page).not_to have_button('Save') # Bad
```

Use `should have_no_*` versions with RSpec matchers because `should_not have_*` doesn't wait for a timeout from the driver.

Capybara cheat sheet

Matchers

```
expect(page).to \
```

```
  have_selector '.blank-state'  
  have_selector 'h1#hola', text: 'Welcome'  
  have_button 'Save'  
  have_checked_field '#field'  
  have_unchecked_field  
  have_css '.class'  
  have_field '#field'  
  have_table '#table'  
  have_xpath '//div'
```

```
  have_link 'Logout', href: logout_path
```

```
  have_select 'Language',  
    selected: 'German'  
    options: ['English', 'German']  
    with_options: ['English', 'German'] # partial match
```

```
  have_text 'Hello',  
    type: :visible # or :all  
    # alias: have_content
```

Common options

All matchers have these options:

```
text: 'welcome'  
text: /Hello/  
visible: true  
count: 4  
between: 2..5  
minimum: 2  
maximum: 5  
wait: 10
```

Capybara cheat sheet

Finding

```
find(selector)
find_button(selector)
find_by_id(id)
find_field(selector)
find_link(selector)
locate
```

Scripting

```
execute_script('$("#input").trigger("change")')
evaluate_script('window.ga')
```

Executes JavaScript.

Page

```
page
  .all('h3')
  .body
  .html
  .source
  .current_host
  .current_path
  .current_url
```

Scoping

```
within '#delivery' do
  fill_in 'Street', with: 'Hello'
end
```

```
within :xpath, '//article'
within_fieldset
within_table
within_frame
scope_to
```

```
find('#x').fill_in('Street', with: 'Hello')
# same as within
```

Debugging

```
save_and_open_page
```

Opens the webpage in your browser.

























AJAX

```
using_wait_time 10 do
  ...
end
```

Jenkins CI Test Suite

Jenkins Jobs

<https://ci.suse.de/user/manager/mv-views/view/TestSuites/>

<div>AllDocumentationSUMA NEXTSUMA30SUMA31SaltTestSuitescucumber-manager</div>			
S	W	Name ↓	Last Success
		manager-2.1-cucumber	1 yr 9 mo - #3714
		manager-3.0-cucumber	4 mo 15 days - #4386
		manager-3.0-qam-cucumber	N/A
		manager-3.1-cucumber	3 days 7 hr - #4342
		manager-3.1-qam-cucumber	1 yr 0 mo - #2
		manager-3.2-cucumber	13 days - #1144
		manager-3.2-qam-cucumber	N/A
		manager-Head-cucumber	1 mo 16 days - #1750
		manager-TEST-cucumber	1 mo 10 days - #3402
		manager-TEST-Hexagon-cucumber	N/A
		manager-TEST-Orion-cucumber	2 days 19 hr - #127
		Uyuni-Master-cucumber	1 mo 29 days - #584

How can I follow the status of my tests?

Project manager-3.2-cucumber

Full test suite of SUSE Manager for 3.2 at PROVO

Controller: suma-32-ctl.prv.suse.net

Server: suma-32-srv.prv.suse.net



[Workspace](#)



[Recent Changes](#)



[Latest Test Result](#) (4 failures / -9)

- You can see **real-time information** about the execution of the Cucumber test scenarios from **console output**
- Query test results in JUnit Tests format clicking on clipboard icon
- Review in a more **detailed report** (including **screenshots on failures**) going to Workspace and looking for last **output.html** file

Console Output

Console Output

Skipping 4,535 KB.. [Full Log](#)

```
m                                     # features/step_definitions/navigation_steps.rb:219
13:35:36     And I follow "Configuration" in the left menu                    # features/step_definitions/navigation_steps.rb:219
13:35:37     And I follow "Configuration Channels" in the left menu             # features/step_definitions/navigation_steps.rb:219
13:35:38     And I follow first "statechannel3"                                    # features/step_definitions/navigation_steps.rb:202
13:35:38     And I follow "Delete Channel"                                         # features/step_definitions/navigation_steps.rb:191
13:35:39     Then I should see a "Are you sure you want to delete this config channel?" text # features/step_definitions/navigation_steps.rb:446
13:35:39     When I click on "Delete Config Channel"                               # features/step_definitions/navigation_steps.rb:172
13:35:40     Then I should see a "Channel 'statechannel3' has been deleted." text   # features/step_definitions/navigation_steps.rb:446
13:35:40     And I remove "/root/statechannel3" from "sle-minion"                 # features/step_definitions/common_steps.rb:498
13:35:40
13:35:40 # Copyright (c) 2016-2018 SUSE LLC
13:35:40 # Licensed under the terms of the MIT license.
13:35:40 Feature: System package list is updated if packages are manually installed or removed
13:35:40
13:35:41     Scenario: Pre-requisite: install milkyway-dummy-1.0 packages            # features/min_salt_pkgset_beacon.feature:6
13:35:41         This scenario ran at: 2019-02-12 13:35:39 +0100
13:35:41         Given I am authorized as "admin" with password "admin"             # features/step_definitions/navigation_steps.rb:360
13:35:42         And I run "zypper -n mr -e Devel_Galaxy_BuildRepo" on "sle-minion"   # features/step_definitions/command_steps.rb:382
13:35:43         And I run "zypper -n ref" on "sle-minion"                           # features/step_definitions/command_steps.rb:382
13:35:45         And I run "zypper -n in --oldpackage milkyway-dummy-1.0" on "sle-minion" without error control # features/step_definitions/command_steps.rb:387
13:35:45
13:35:48     Scenario: Pre-requisite: ensure the errata cache is computed             # features/min_salt_pkgset_beacon.feature:12
13:35:48         This scenario ran at: 2019-02-12 13:35:44 +0100
13:35:48         Given I am on the Systems overview page of this "sle-minion"         # features/step_definitions/navigation_steps.rb:300
13:35:49         When I follow "Software" in the content area                        # features/step_definitions/navigation_steps.rb:219
13:35:50         And I follow "List / Remove" in the content area                    # features/step_definitions/navigation_steps.rb:219
13:35:50         And I enter "milkyway-dummy" in the css "input[placeholder='Filter by Package Name: ']" # features/step_definitions/salt_steps.rb:275
13:36:04         And I click on the css "button.spacewalk-button-filter" until page does contain "milkyway-dummy-1.0" text # features/step_definitions/salt_steps.rb:252
13:36:04         Then I follow "Admin"                                                # features/step_definitions/navigation_steps.rb:191
13:36:05         And I follow "Task Schedules"                                       # features/step_definitions/navigation_steps.rb:191
13:36:06         And I follow "errata-cache-default"                                 # features/step_definitions/navigation_steps.rb:191
13:36:08         And I follow "errata-cache-bunch"                                   # features/step_definitions/navigation_steps.rb:191
13:36:09         Then I click on "Single Run Schedule"                               # features/step_definitions/navigation_steps.rb:172
13:36:09         And I should see a "bunch was scheduled" text                       # features/step_definitions/navigation_steps.rb:446
13:36:11         Then I wait until the table contains "FINISHED" or "SKIPPED" followed by "FINISHED" in its first rows # features/step_definitions/navigation_steps.rb:565
13:36:11
```


Cucumber Test Report

Cucumber Features

115 scenarios (10 failed, 105 passed)
1171 steps (10 failed, 48 skipped, 1113 passed)
Finished in 26m32.950s seconds
[Collapse All](#) [Expand All](#)

```
# Copyright (c) 2017-2018 SUSE LLC
# Licensed under the terms of the MIT license.
```

Feature: Very first settings

```
In order to use the product
As the admin user
I want to create the organisation, the first users and set the HTTP proxy
```

```
Scenario: Create admin user and first organization
Scenario: Create testing username
Scenario: Grant testing user administrative privileges
Scenario: Wait for refresh of list of products to finish
Scenario: Check services which should run
Scenario: Setup HTTP proxy
Scenario: Detect latest Salt changes on the server
```

```
# Copyright (c) 2019 SUSE LLC
# Licensed under the terms of the MIT license.
```

Feature: Adding channels

```
In Order distribute software to the clients
As an authorized user
I want to add channels
```

```
Background
Scenario: Add a base channel
Scenario: Add a child channel
Scenario: Add a base test channel for i586
Scenario: Add a child channel to the i586 test channel
Scenario: Add a test base channel for x86_64
Scenario: Add a child channel to the x86_64 test channel
Scenario: Add Fedora x86_64 base channel
Scenario: Add Ubuntu AMD64 base channel
Scenario: Fail when trying to add a duplicate channel
Scenario: Fail when trying to use invalid characters in the channel label
Scenario: Fail when trying to use invalid characters in the channel name
Scenario: Fail when trying to use reserved names for channels
Scenario: Fail when trying to use reserved labels for channels
Scenario: Create a channel that will be changed
Scenario: Fail when trying to change the channel name to a reserved name
```

Cucumber Test Report

```
Scenario: Refresh the channel's repository data
Scenario: Reposync handles wrong encoding on RPM attributes
@ubuntu_minion
Scenario: Reposync handles wrong encoding on DEB attributes

  This scenario ran at: 2019-01-16 12:14:31 +0100

  Given I am authorized as "admin" with password "admin"

  When I follow "Home" in the left menu

  And I follow "Channel List"

  And I follow "Test-Channel-Deb-AMD64"

  And I follow "Packages" in the content area

  Then I should see a "blackhole-dummy" text

  ./features/step_definitions/navigation_steps.rb:484:in `^I should see a "([^\"]*)" text$/'
  features/core_srv_create_repository.feature:176:in `Then I should see a "blackhole-dummy" text'
  ./features/step_definitions/navigation_steps.rb:484:in `^I should see a "([^\"]*)" text$/'
  features/core_srv_create_repository.feature:176:in `Then I should see a "blackhole-dummy" text'

482   unless page.has_content?(text)
483     sleep 2
484     raise unless page.has_content?(text)
485   end
486 end
487 # gem install syntax to get syntax highlighting
```

Screenshot

