Creating a dynamic software deployment solution using free/libre software

LinuxTag 2014

08.05.2014

Mattias Giese
Solutions Architect for Systems Management and Monitoring
B1 Systems GmbH
giese@b1-systems.de

Agenda

Agenda

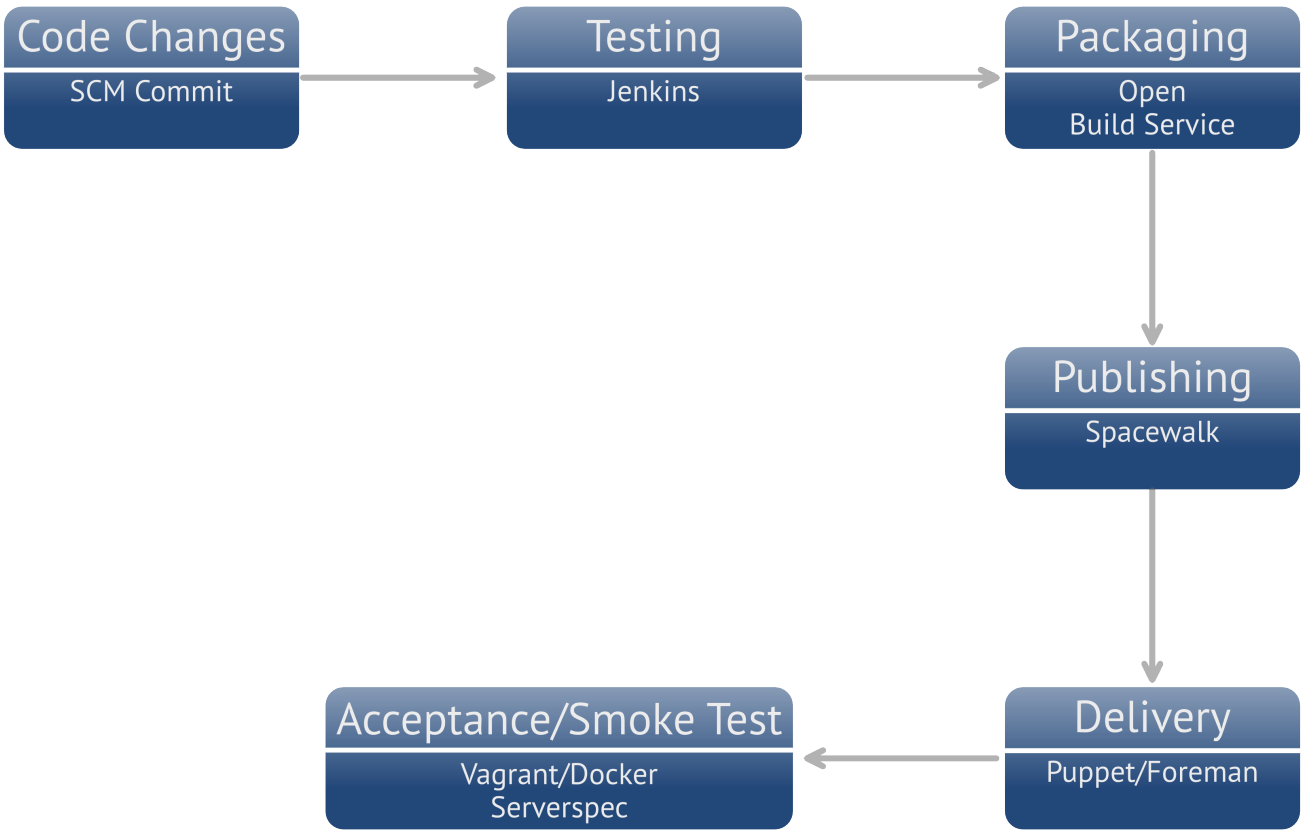
- Workflow/Process Overview
- SCM Hooks
- Jenkins
- Open Build Service
- Spacewalk
- Puppet/Foreman

Process Overview

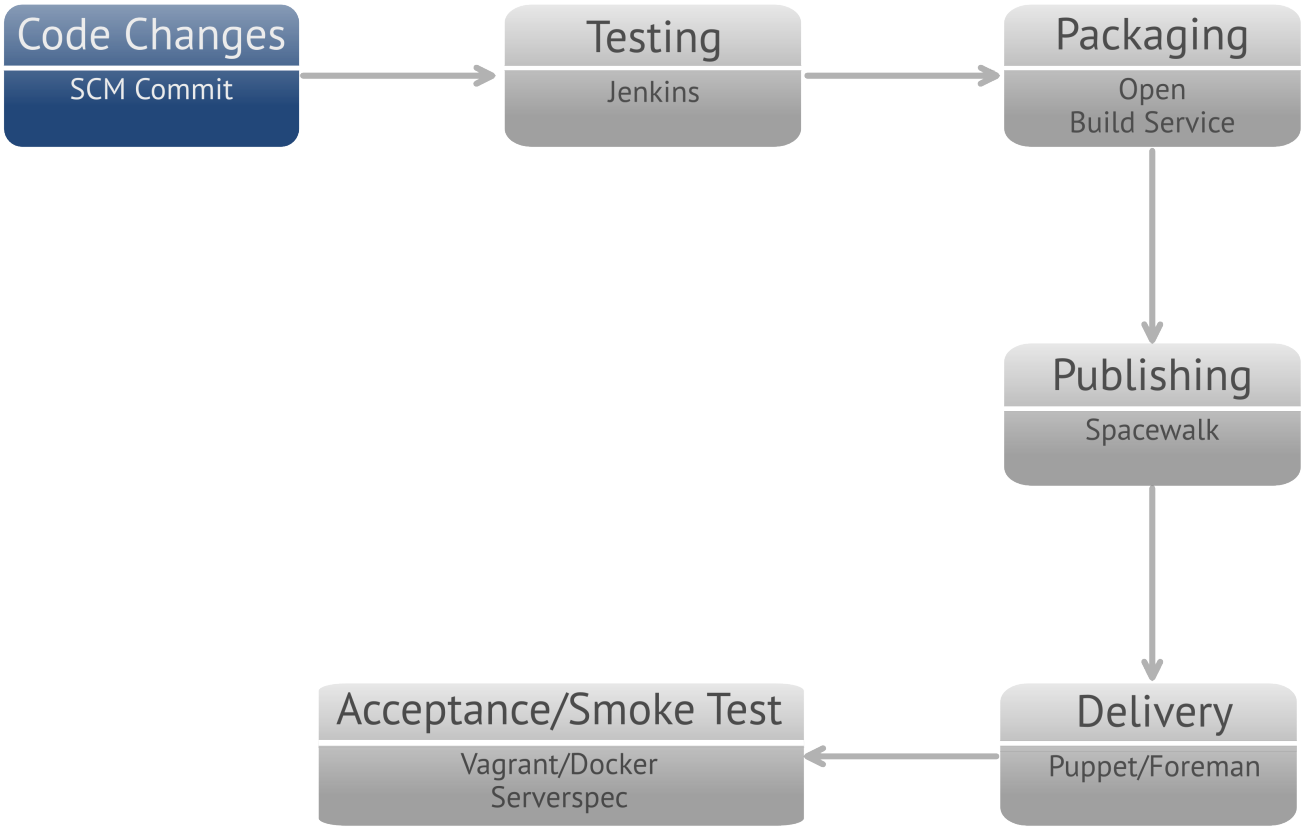
What are we trying to accomplish?

- Continuous Integration
  - New features should be merged as quickly as possible
  - Confidence that changes don't break production
  - Base for test-driven development (TDD)
- Continuous Delivery
  - We don't want to deploy every revision of our software
  - But would be cool if we could

Process Overview



Code Changes and Commit


Code Changes
SCM Commit
Testing
Jenkins
Packaging
Open
Build Service
Publishing
Spacewalk
Delivery
Puppet/Foreman
Acceptance/Smoke Test
Vagrant/Docker
Serverspec

Using SCM-Hooks

- Hooks → do stuff
- supported in many SCMs
- git-hooks may be run at different phases
- Use Cases:
  - Syntax check
  - Linter
  - Enforcement of other policy
  - Triggering of other processes
- We want to trigger Jenkins!

Using SCM-Hooks

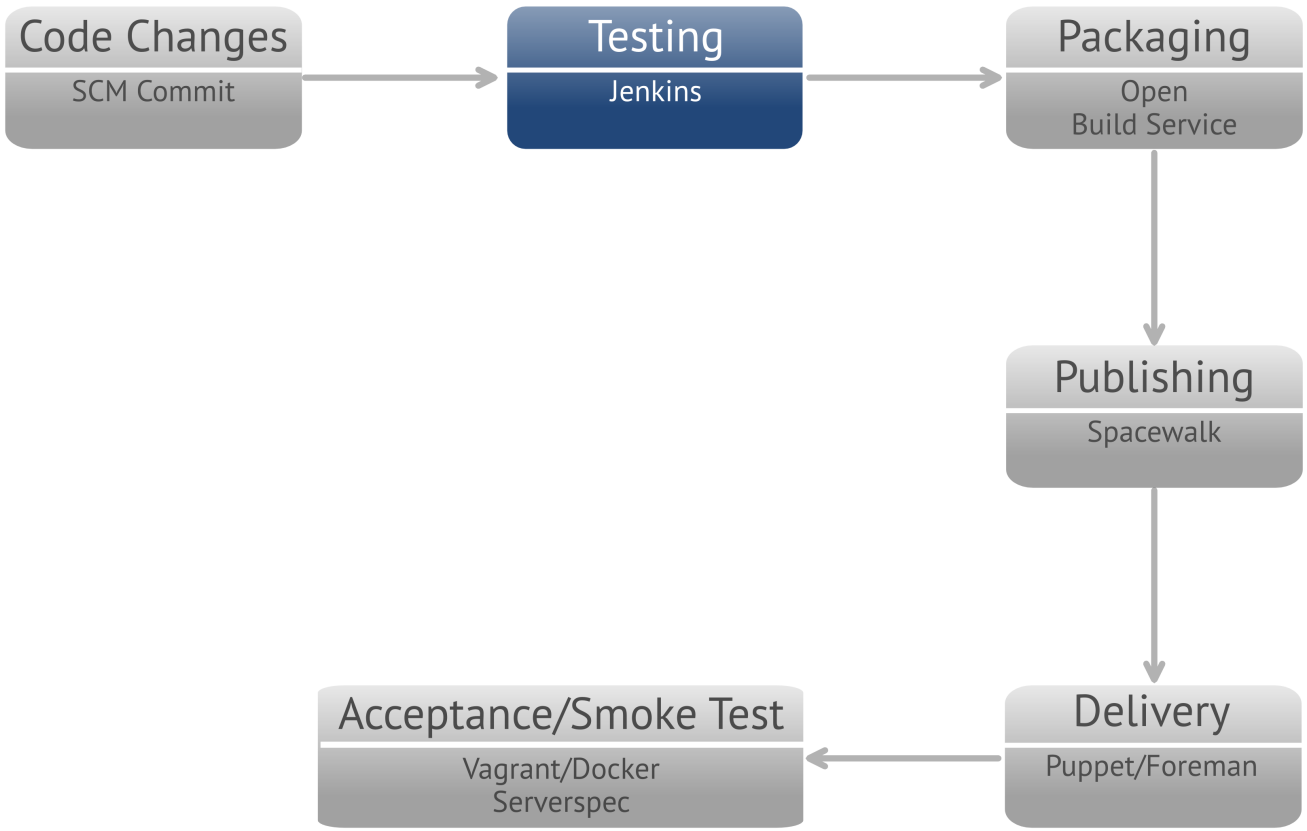
Simple post-receive hook

```
#!/bin/bash
AUTH='myuser:pass'
JOB=myjob
TOKEN=mybuildtoken

echo Poking jenkins
/usr/bin/curl --user $AUTH -s \
"http://jenkins:8080/job/$JOB/build?token=$TOKEN"
```

Testing with Jenkins

Testing with Jenkins



Jenkins Overview

- The artist formerly known as Hudson
- The first address for CI
- can be used as Cron, general Job-Scheduler or Monitor
- 600+ plugins (nifty)
- Alternatives: Travis-CI, CruiseControl, Bamboo

Jenkins Overview

Must-Have Plugins

- Warnings
- Post-Build
- Build Pipeline
- Promoted Builds
- Git
- Chuck Norris

Usage in this scenario

Jenkins

- is triggered through git post-receive hook
- checks out git-repo (specific branch)
- runs configured build steps
- after successful build: trigger packaging process

Jenkins Post-Build

Post-Build Step, complete

```
#!/bin/bash
auth='obsuser:passwd'
project='foobar'
main_project='internal-tools'
git_host='git.example.com'
obs_api='https://api.obs.example.com'
branch='development'
service="<services>
 <service name="tar_scm">
  <param name="scm">git</param>
  <param name="url">git://${git_url}/${project}.git</param>
  <param name="revision">$GIT_COMMIT</param>
 </service>
[...]
</services>"
echo $service | curl -u $auth -s -X PUT -T - \
  "$obs_api/source/$main_project/$project/_service"
```

Post-Build Step, Service File

```
<services>
 <service name="tar_scm">
  <param name="scm">git</param>
  <param name="url">git://${git_url}/${project}.git</param>
  <param name="revision">$GIT_COMMIT</param>
 </service>
 <service name="recompress">
  <param name="file">*$project*.tar</param>
  <param name="compression">bz2</param>
 </service>
 <service name="download_url">
  <param name="path">/$project/$branch/${project}.spec</param>
  <param name="host">$git_host</param>
  <param name="protocol">https</param>
 </service>
 <service name="set_version">
 </service>
</services>
```

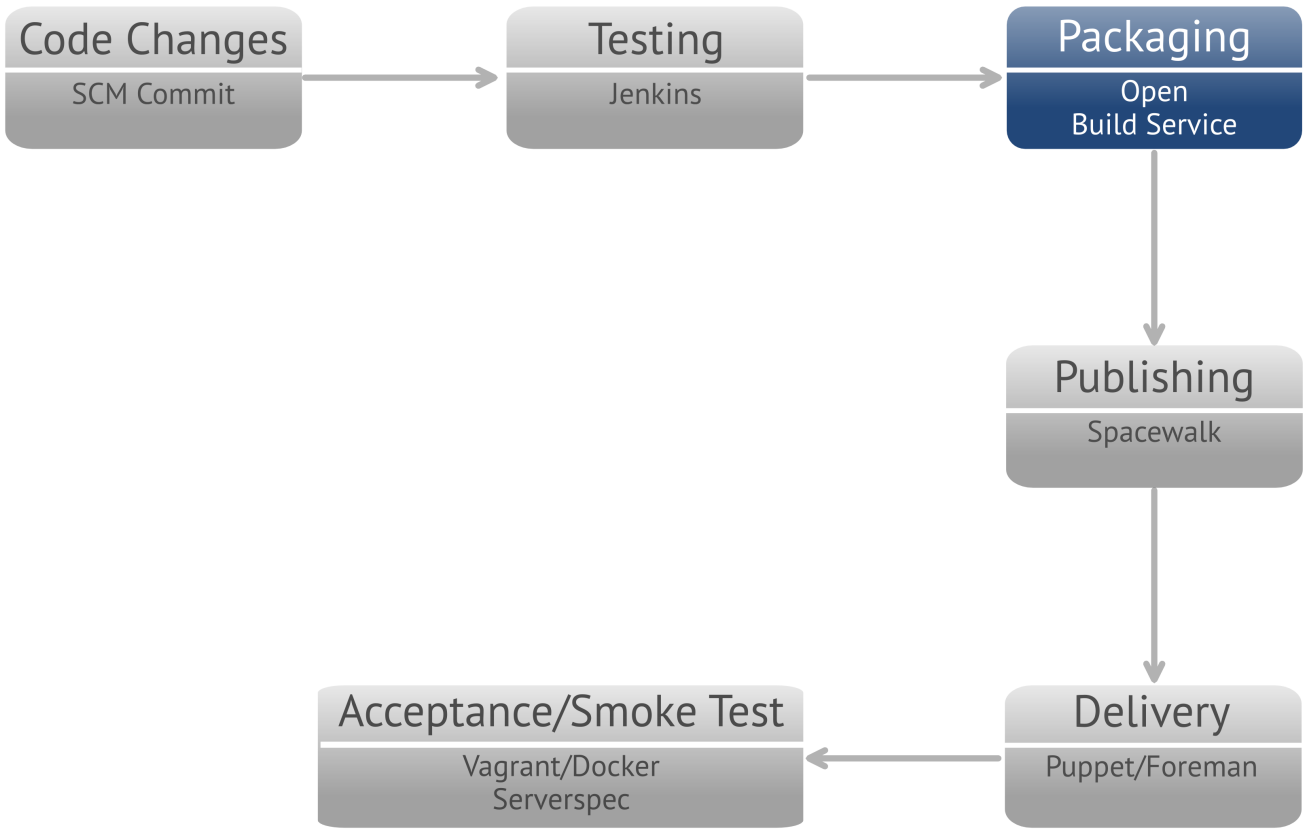
Jenkins Post-Build

Post-Build Step, the trigger itself

```
echo $service | curl -u $auth -s -X PUT -T - \
  "$obs_api/source/$main_project/$project/_service"
```

Packaging

Packaging with OBS



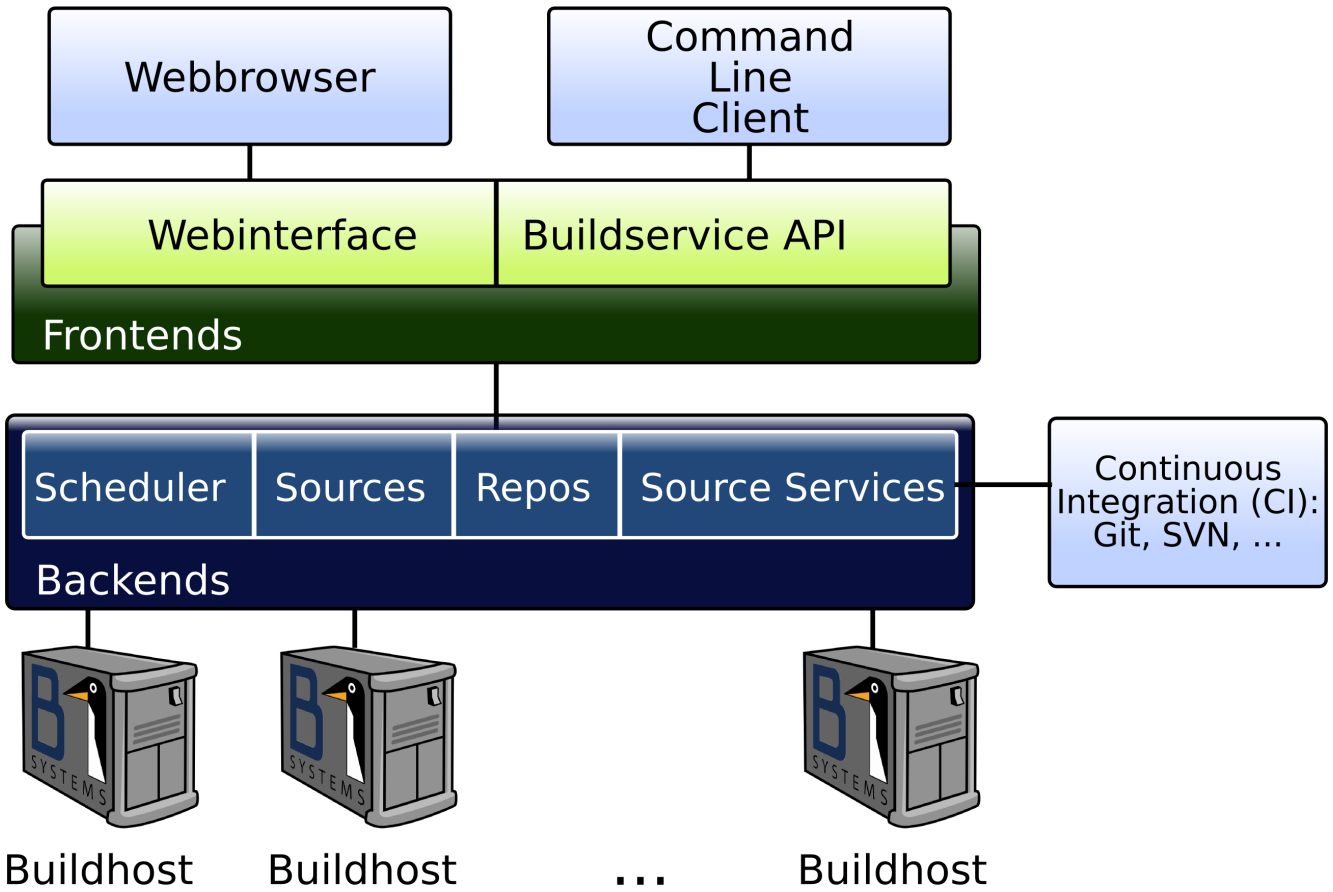
Open Build Service Overview

- Build Platform for packages or live media (kiwi)
- builds RPM (SUSE/RHEL), DPKG (Debian/Ubuntu) and Pacman (Archlinux)
- public reference install at `build.opensuse.org`
- Central build platform for openSUSE

Open Build Service Overview

- Extensive API (CLI is an API client)
- can run hooks after successful publishing of a package/project
- Alternatives: None, koji if all you want is RHEL

Open Build Service Overview

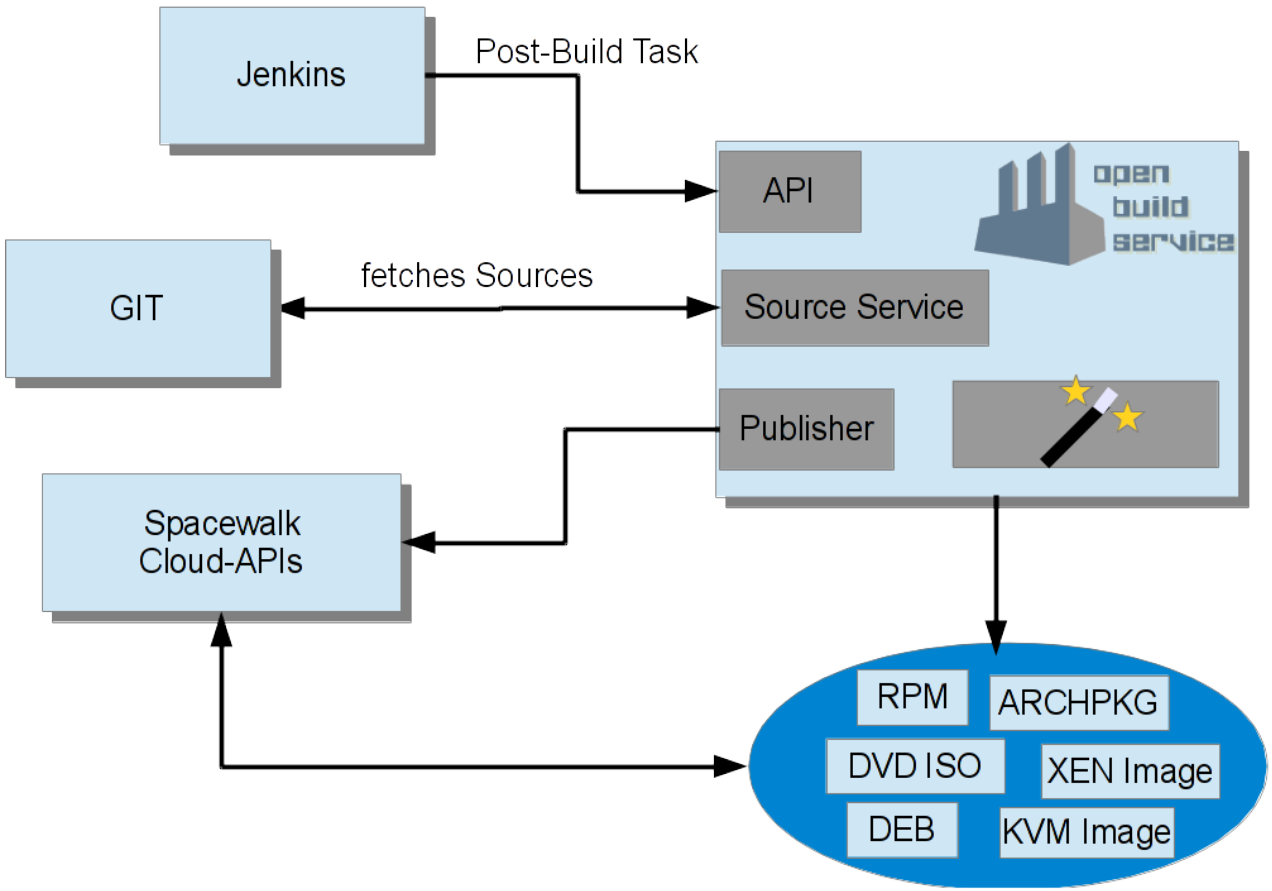


Usage in this scenario

Open Build Service is used

- to build packages for different distributions/architectures
- to trigger Spacewalk to sync packages

Open Build Service Overview



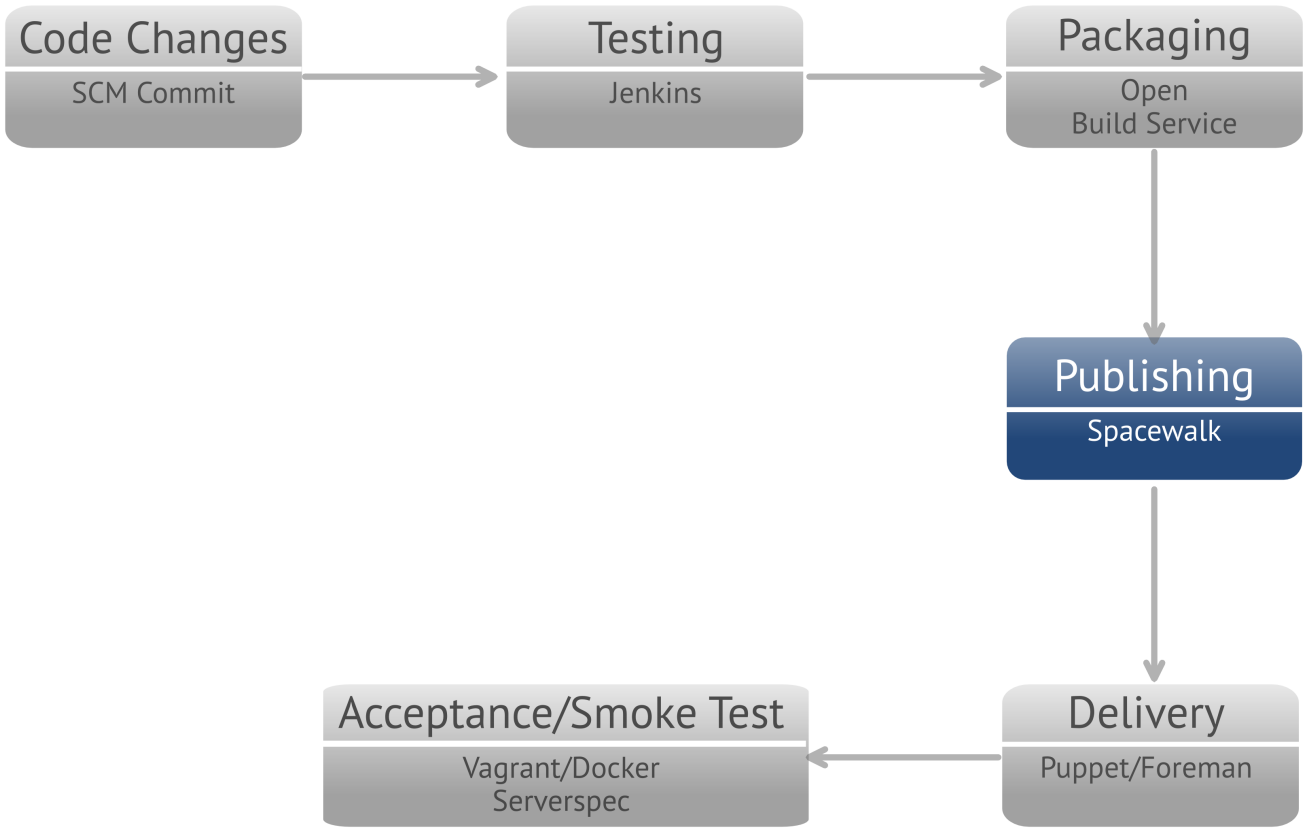
OBS should trigger Spacewalk!

OBS published-hook

```
#!/usr/bin/env python2
import xmlrpclib
import sys
SPACEWALK_URL = "http://spacewalk.example.com/rpc/api"
SPACEWALK_LOGIN = "apiuser"
SPACEWALK_PASSWORD = "testing"
try:
  channel = sys.argv[1]
  client = xmlrpclib.Server(SPACEWALK_URL, verbose=0)
  key = client.auth.login(SPACEWALK_LOGIN, SPACEWALK_PASSWORD)
  client.channel.software.syncRepo(key,channel)
  client.auth.logout(key)
except Exception, e: # Gotta catch em all
  print 'Failed to sync channel: ', e
  sys.exit(1)
```

Publishing

Package Publishing with Spacewalk



Spacewalk Overview

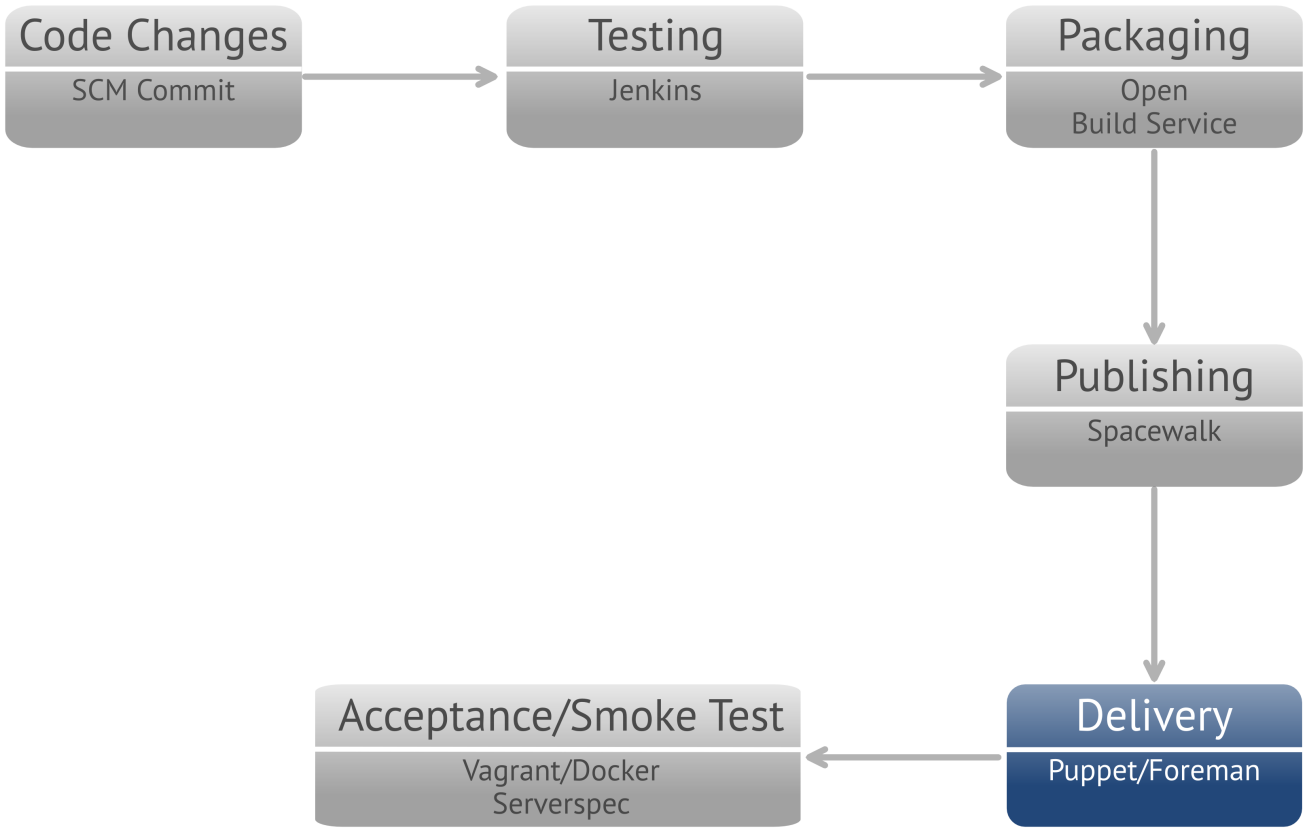
- System Lifecycle Management
- Extensive API
- runs on RHEL/Fedora and Derivatives
- supports SUSE/RHEL and Derivatives, Debian somewhat.
- Base for SUSE Manager and Satellite (changing with 6.x)
- Alternatives: Pulp, if all you want is RHEL

Usage in this scenario

- Provisioning of packages
- Staging of packages
- Automatic synchronization of packages in “Development” channel
- On-Demand/automatic promotion of channel contents to integration/production stages

Delivery

Package Delivery with Puppet/Foreman



Puppet

- Configuration Management Tool
- based on Ruby, easy to learn DSL
- runs on many platforms (Linux, *nix, Windows)
- abstracts differences between those platforms
- is idempotent
- features Inventory and Reporting on Infrastructure changes
- Alternatives: CFEngine, Chef, maybe Ansible or SaltStack

Simple Puppet manifest

```
package{'myapp':
  ensure => latest,
}
file{'/etc/myapp/config':
  ensure => file,
  owner => 'serviceuser',
  group => 'servicegroup',
  mode => '0640',
  content => template('myapp-config.erb'),
  require => Package['myapp'],
  notify => Service['myapp'],
}
service{'myapp':
  ensure => running,
  enable => true,
  require => Package['myapp'],
}
```

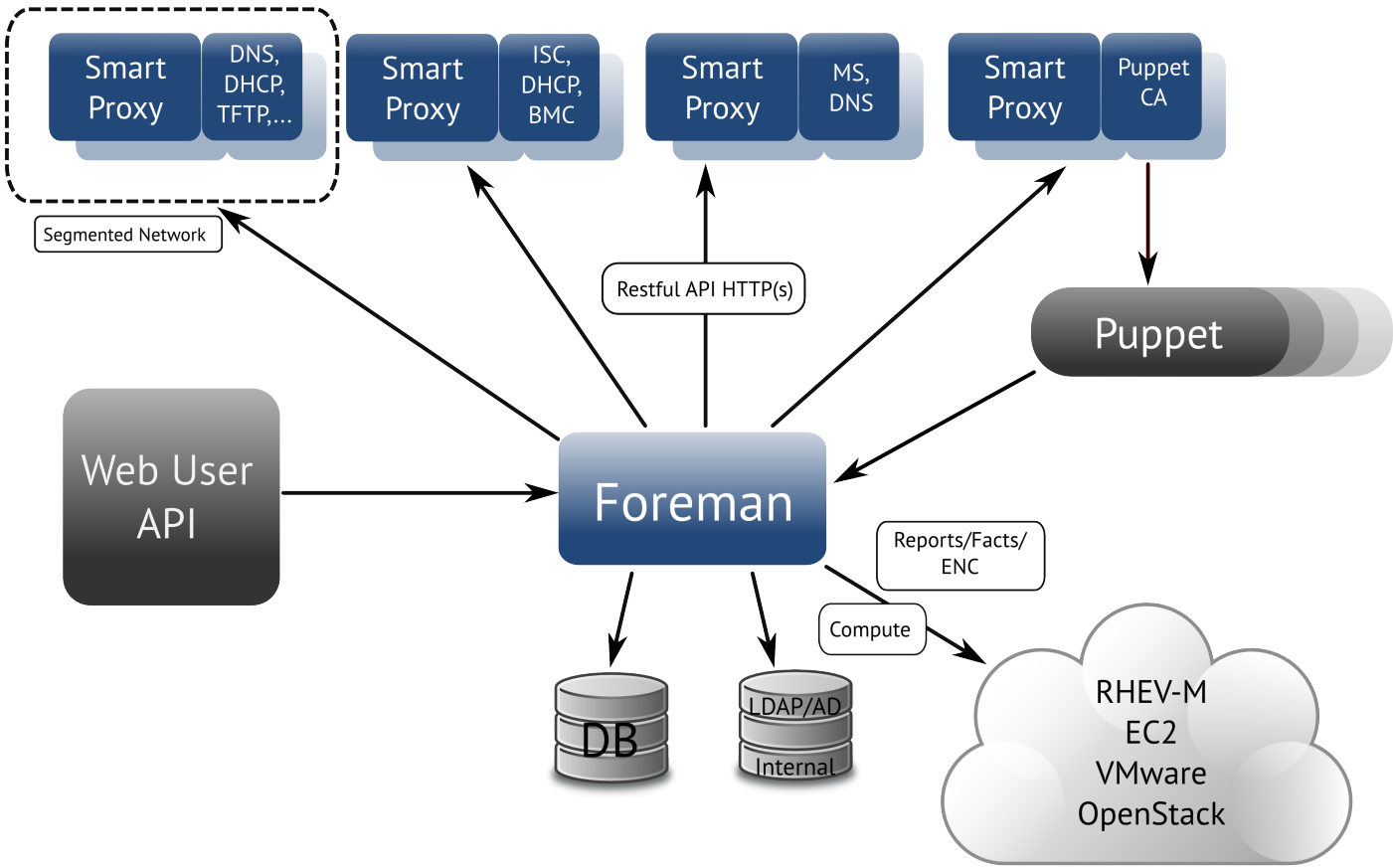
The Foreman

- WebApp and API to automate infrastructure
- integrates well with Puppet
- can manage DHCP, DNS, TFTP, Puppet, “Clouds"
- features nice reporting and auditing
- Alternatives:

  Provisioning: Cobbler, Razor

  Management: Puppet Dashboard/Puppet Enterprise Console, Rudder (CFEngine)

Architecture Overview

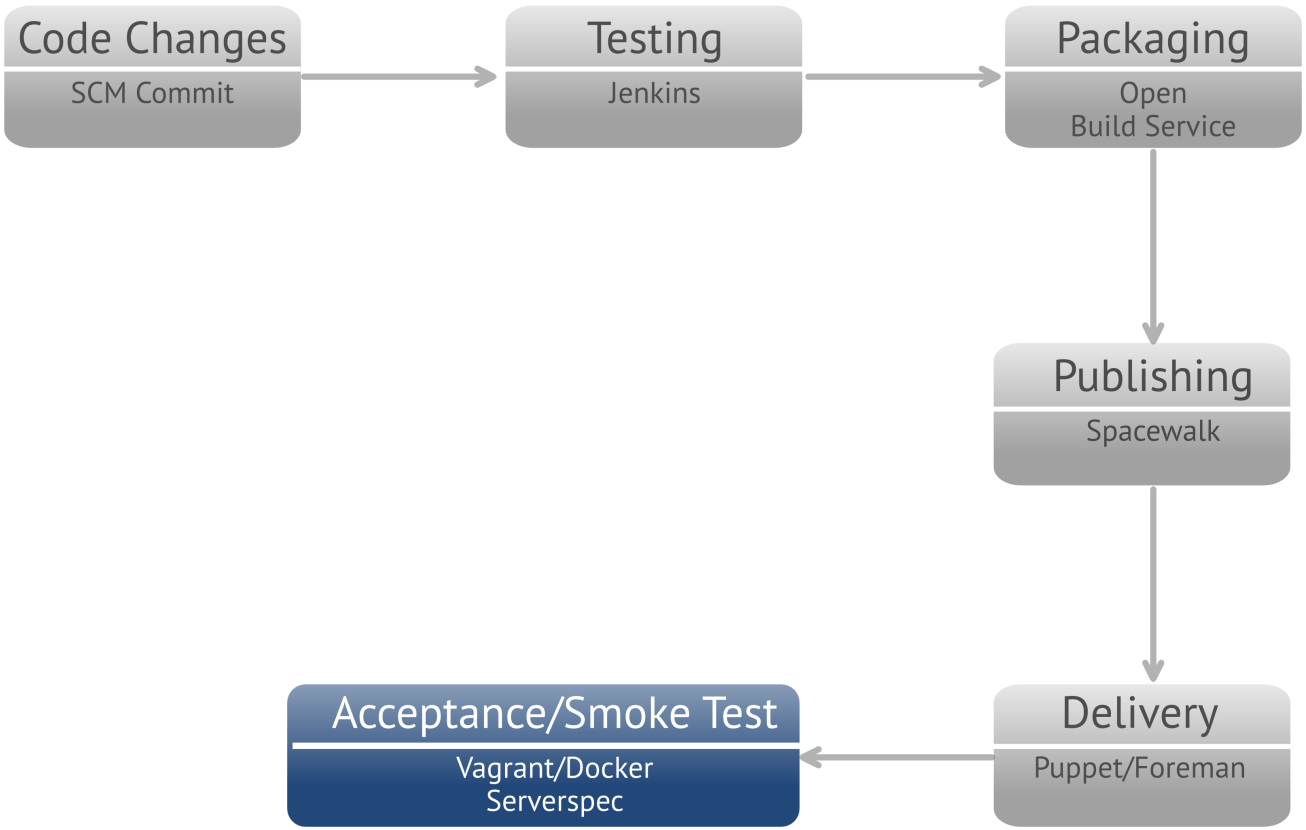


Usage of Foreman in this setup

- Provisioning of new bare-metal or virtual machines
- Single source of truth for Puppet
- Reporting and Auditing of Infrastructure
- Idea: Use Foreman-API to trigger VM provisioning for acceptance tests

System/Acceptance Tests

System/Acceptance Tests



Overview

- Vagrant
- Docker
- Smoketests
- Serverspec

Vagrant

- Wrapper around different Hypervisors and Provisioners
- supports VirtualBox (First Class), VMware (commercial), libvirtd (somewhat) and Docker
- runs on Linux, Mac OS X, Windows
- Provisioners may be Shell, Puppet, Chef and so on
- Easy to use:
  1. `mkdir myvm; cd myvm`
  2. `vagrant init precise64 http://files.vagrantup.com/precise64.box`
  3. `vagrant up`
  4. `vagrant ssh`
  5. `vagrant halt/destroy`

Docker

- chroot on steroids
- designed to execute and contain one process
- uses LXC, cgroups, UnionFS(AUFS)
- does not need a custom kernel (anymore)
- runs on many current Linux distributions
- yep, it also runs on current RHEL6(.5) and SLES12 once released
- not useful in every scenario (no mounts, modules, init and so on)

VM with no VM overhead

Example: Initialization, Execution, Destruction

```
$ time docker run -t -i smoketest-suse hostname
d20cdb21deb3

real  0m0.230s
user  0m0.007s
sys 0m0.003s
$ time docker run -t -i --rm \
    smoketest-suse hostname
e1e1b445465d

real  0m4.250s
user  0m0.000s
sys 0m0.010s
```

Smoketest with Docker

Application of Puppet Manifest: Does it blow up?

```
$ docker run -t -i smoketest-suse \
    bash -c "r10k deploy -p; puppet apply \
    --detailed-exitcodes --modulepath=/test/modules \
    -e 'include memcached'"
```

Serverspec

- based on RSpec
- executes tests locally or remotely (`ssh`)
- integrates with Vagrant somewhat
- Supported resources: `cgroup`, `command`, `cron`, `default_gateway`, `file`, `group`, `host`, `iis_app_pool`, `iis_website`, `interface`, `ipfilter`, `iptables`, `kernel_module` and more

Serverspec - Example Spec

Simple System/Acceptance Test for memcached (excerpt)

```
describe package('memcached') do
  it { should be_installed }
end
describe service('memcached') do
  it { should be_enabled   }
  it { should be_running   }
end
describe port(11211) do
  it { should be_listening.with('tcp') }
  it { should be_listening.with('udp') }
end
```

Serverspec – Example Spec

Simple System/Acceptance Test for memcached (excerpt)

```
describe file('/etc/sysconfig/memcached') do
  it { should be_readable }
  its(:content) { should match /PORT="11211"/ }
end
describe command('echo -e \
            "add Test 0 60 11\r\nHello World\r" | nc localhost 11211') do
  it { should return_stdout /^STORED$/ }
end
describe command('echo -e "get Test\r" | nc localhost 11211') do
  it { should return_stdout /^Hello World$/ }
end
```

Serverspec – Example Run

Test Run for memcached, Part 1

```
$ rake spec SPEC_OPTS="-fd"
(in /home/b1user/serverspec)
/usr/bin/ruby -S rspec spec/default/memcached_spec.rb

Package "memcached"
  should be installed

Service "memcached"
  should be enabled (FAILED - 1)
  should be running

Port "11211"
  should be listening
  should be listening
```

Test Run for memcached, Part 2

```
File "/etc/sysconfig/memcached"
  should be readable
  content
    should match /PORT="11211"/

Command "echo -e "add Test 0 60 11\r\nHello World\r" | nc localhost 11211"
  should return stdout /^STORED$/

Command "echo -e "get Test\r" | nc localhost 11211"
  should return stdout /^Hello World$/
[...]
Failures:
  1) Service "memcached" should be enabled
     Failure/Error: it { should be_enabled   }
       sudo chkconfig --list memcached | grep 3:on
       expected Service "memcached" to be enabled
     # ./spec/default/memcached_spec.rb:8
Finished in 11.36 seconds
12 examples, 1 failure
[...]
```

What's Next?

After successful acceptance test

1. Promotion of packages from Dev to Test/Integration software channel (Spacewalk)
2. Integration Test of packages
3. Promotion of packages from Test to Prod
4. Canary Testing $\rightarrow$ does it blow up in Prod?
5. Profit

Thank you for your attention!

If you have further questions please contact info@b1-systems.de or call +49 (0)8457 - 931096