CONTINUOUS DELIVERY OF CONTINUOUS DELIVERY

Gerd Aschemann

JAVA FORUM STUTTGART 2016

- Gerd Aschemann
- **gerd@aschemann.net**
- **http://aschemann.net**
- twitter: **@GerdAschemann**

FRAGESTUNDE

- Wer nutzt Continuous Integration?
- Wer betreibt Continuous Delivery?
- Wer setzt auf *Infrastructure as Code* (IaC)?
- Wer baut seine CIP/CDP per IaC?
- Wem sagt das alles gar nichts?

AGENDA

- Continuous Delivery
- Platform as Code (PaC)
- Lösungsarchitektur
- Umsetzung
- Fazit

WAS IST CONTINUOUS DELIVERY?

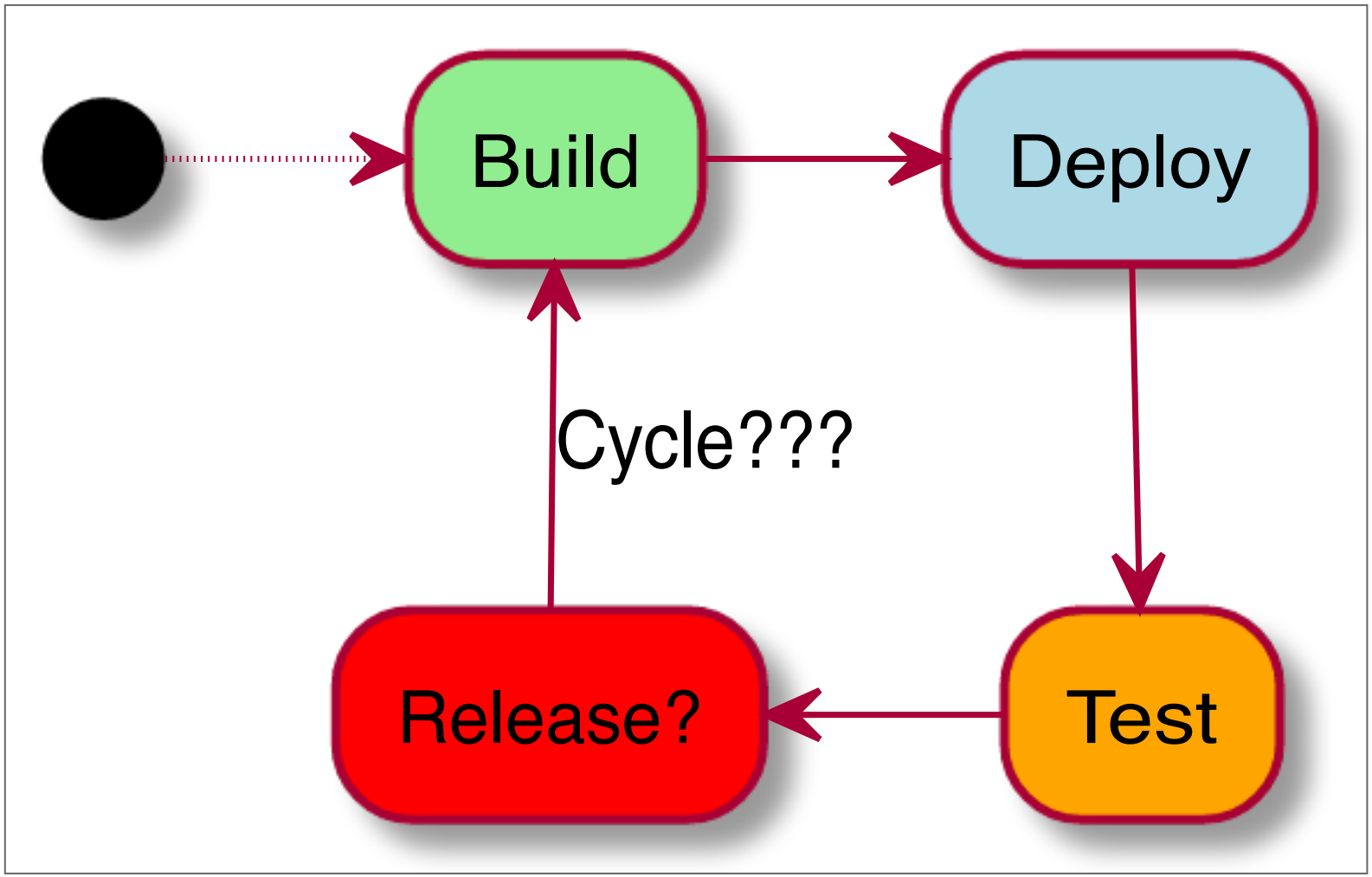
Frei nach **Martin Fowler**

- Software ist jederzeit deploybar
- Deploybarkeit hat Vorrang vor Erweiterungen
- Aussagefähigkeit über Production-readiness nach jedem Change
- Push-button Deployment beliebiger Versionen in beliebige Umgebungen

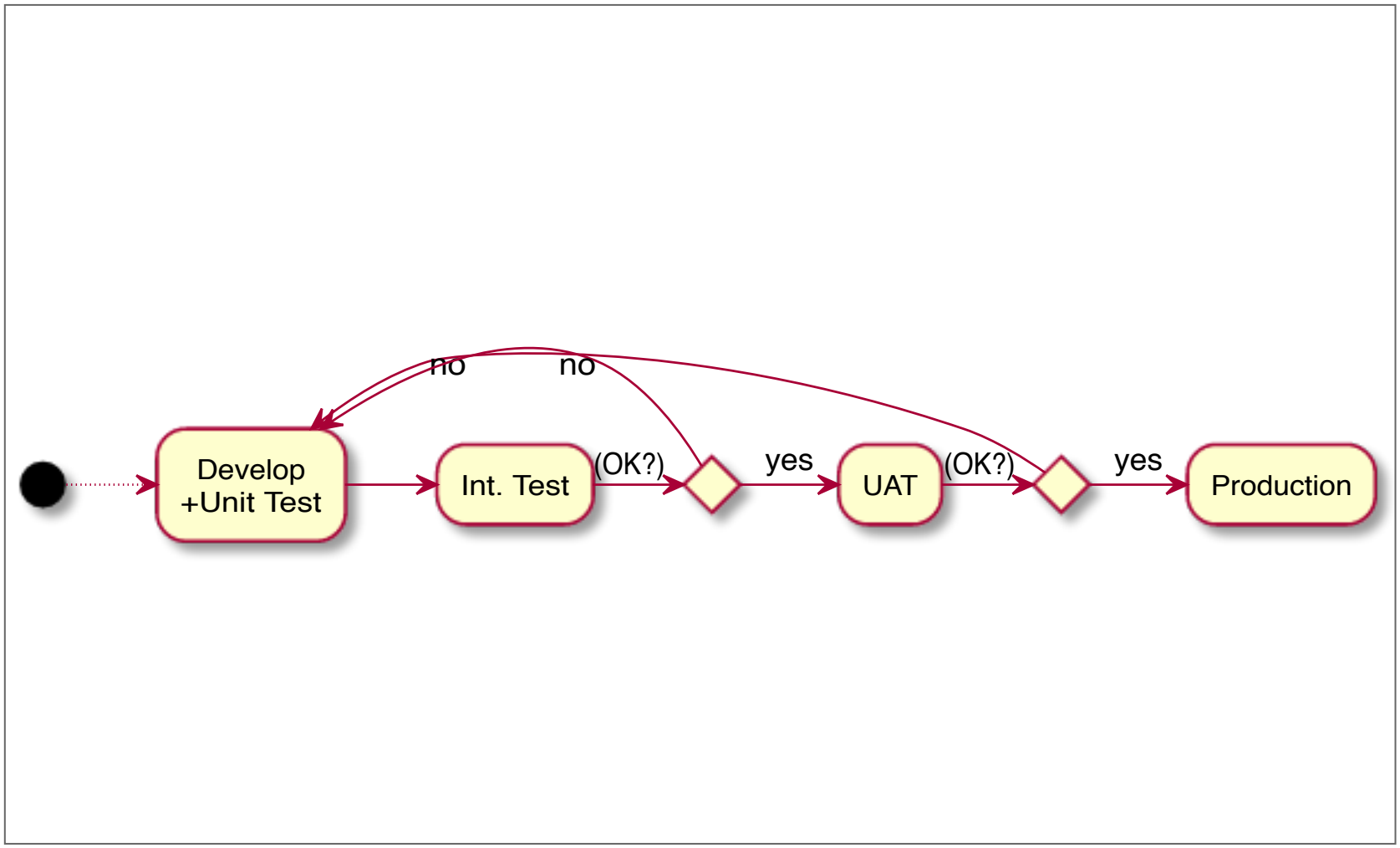
❗ Hohe Automatisierung

⚠ Continuous Delivery ist *NICHT* Continuous Deployment!

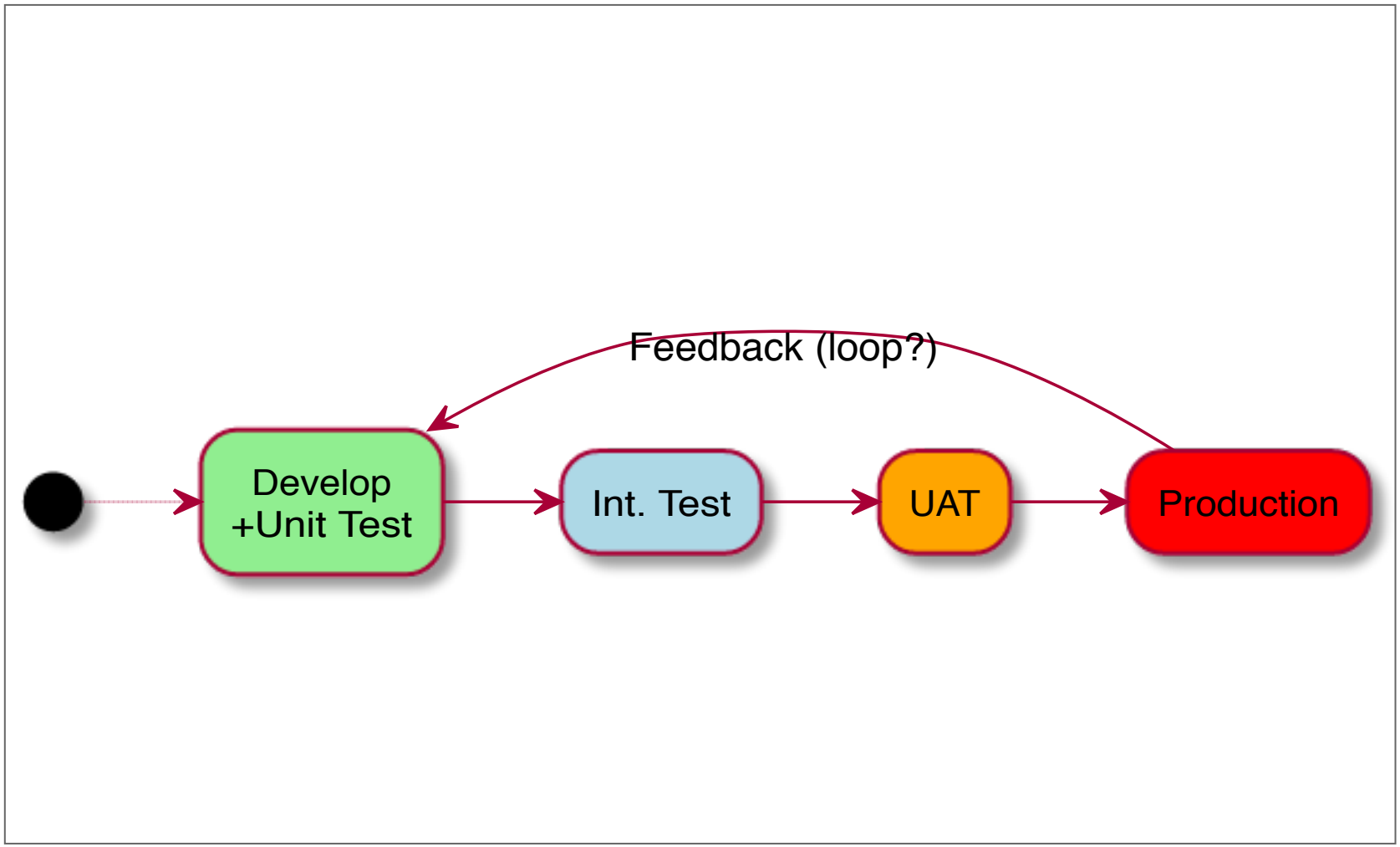
CONTINUOUS DELIVERY CYCLE (?)



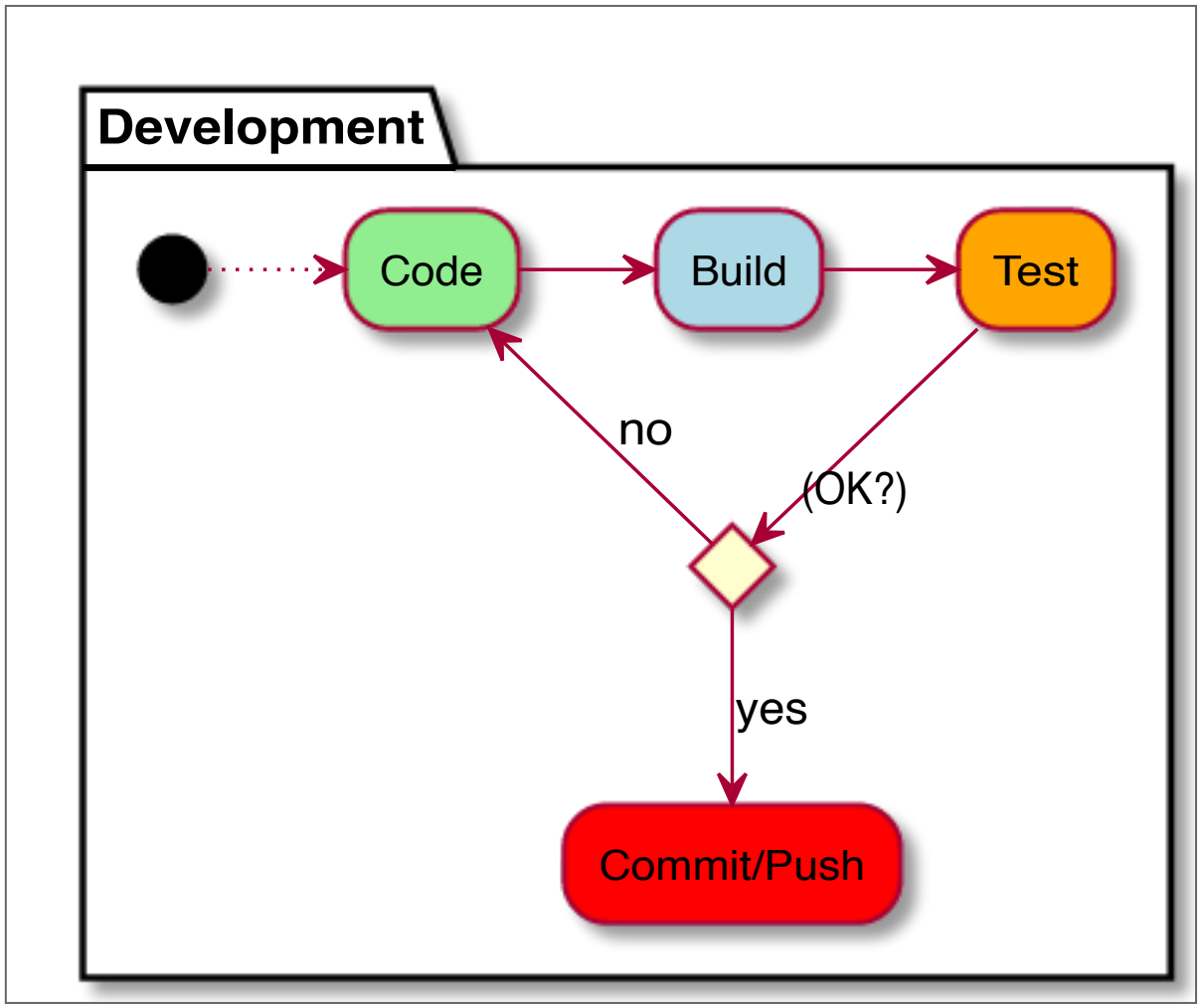
CONTINUOUS DELIVERY CASCADE (???)



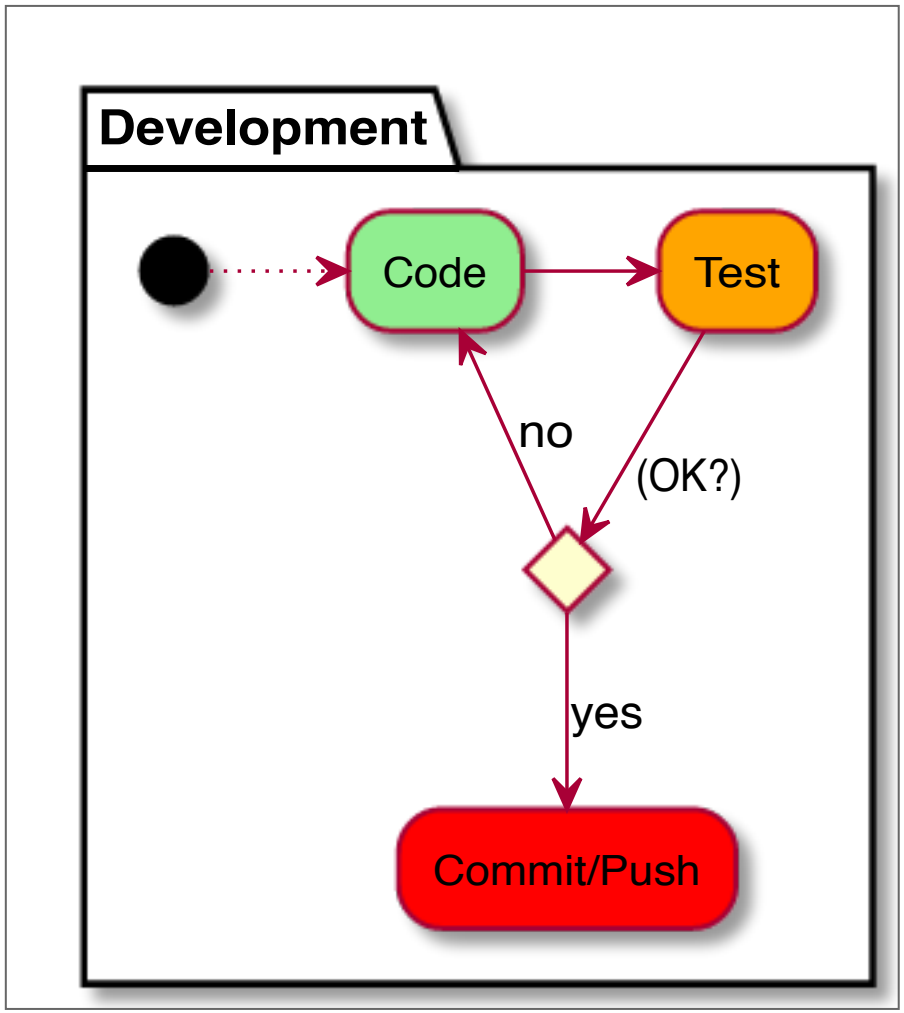
CONTINUOUS DELIVERY LOOP (!)



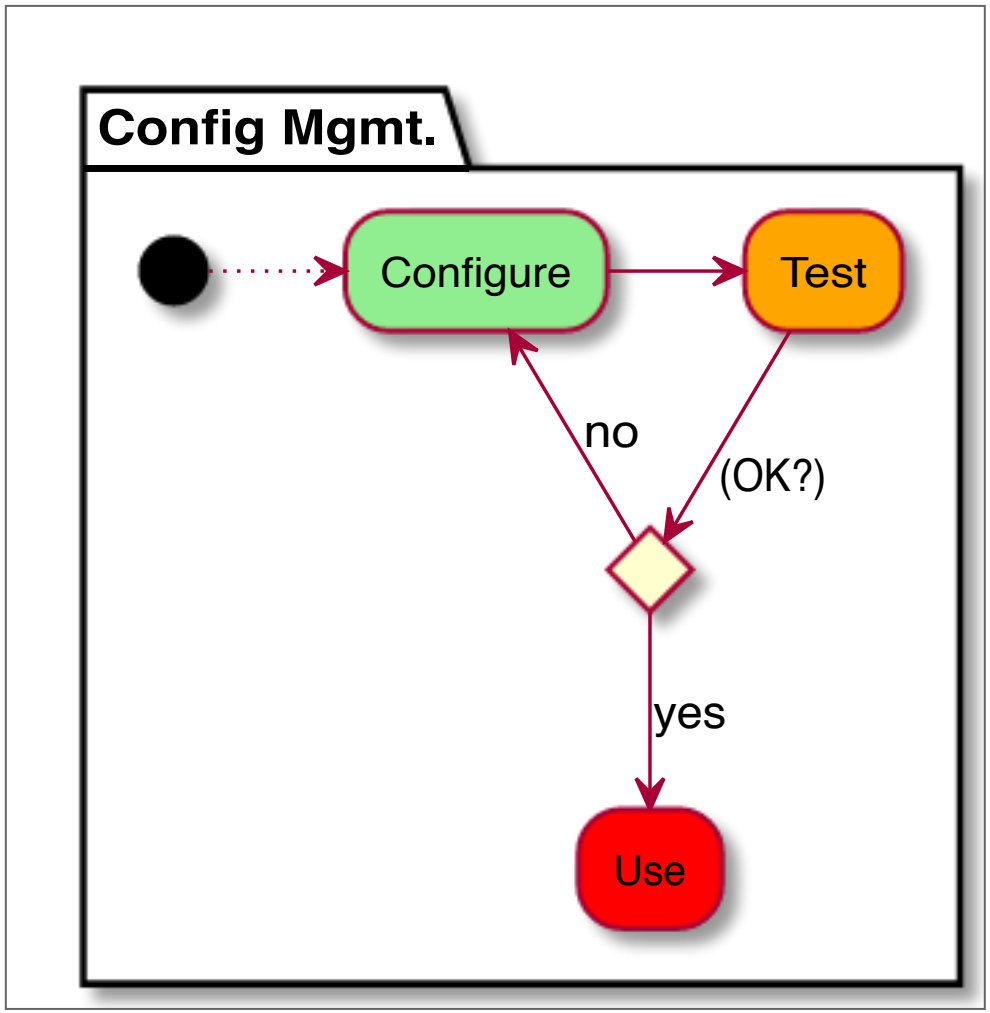
RÜCKBLICK: DEVELOPMENT



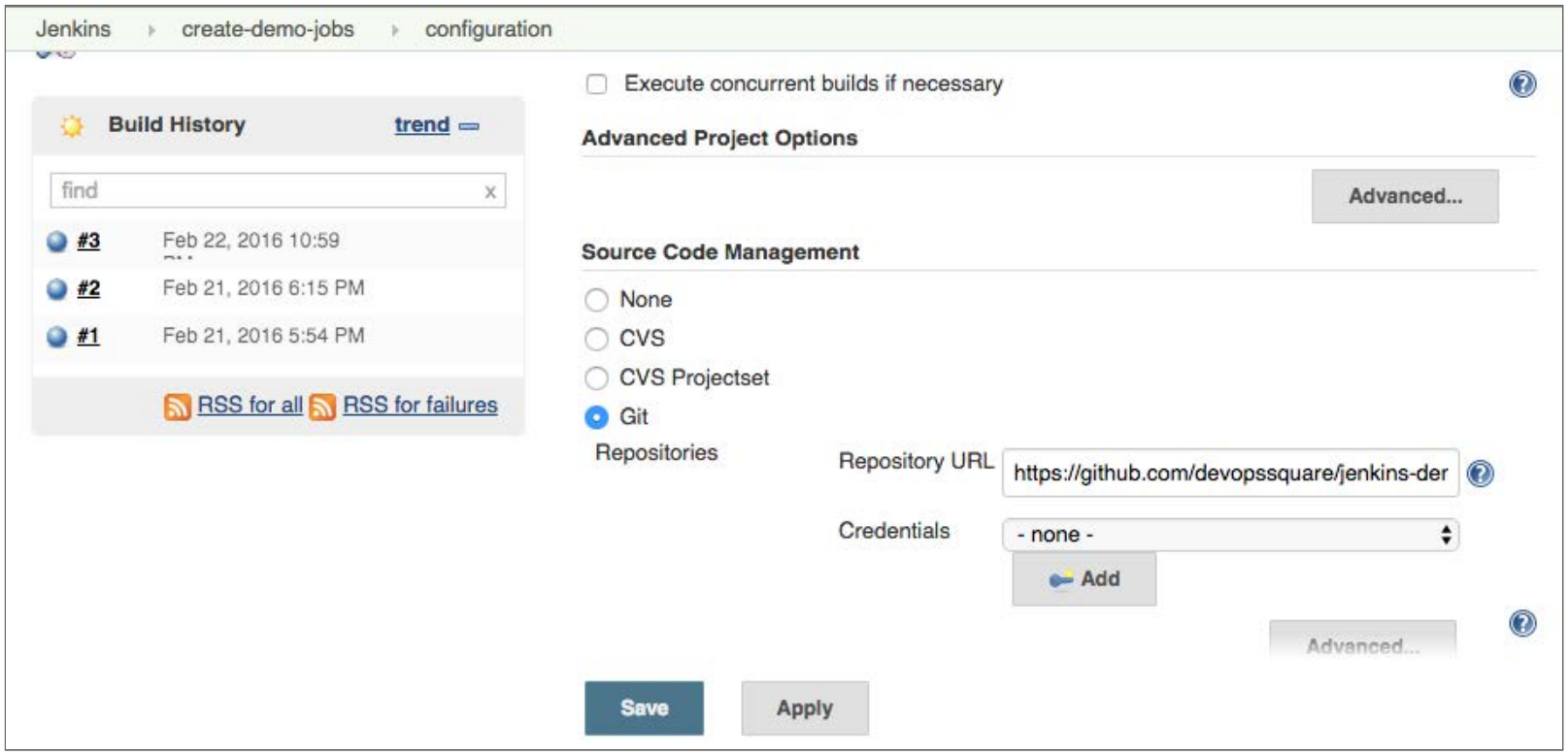
DEVELOPMENT NOCH EINFACHER



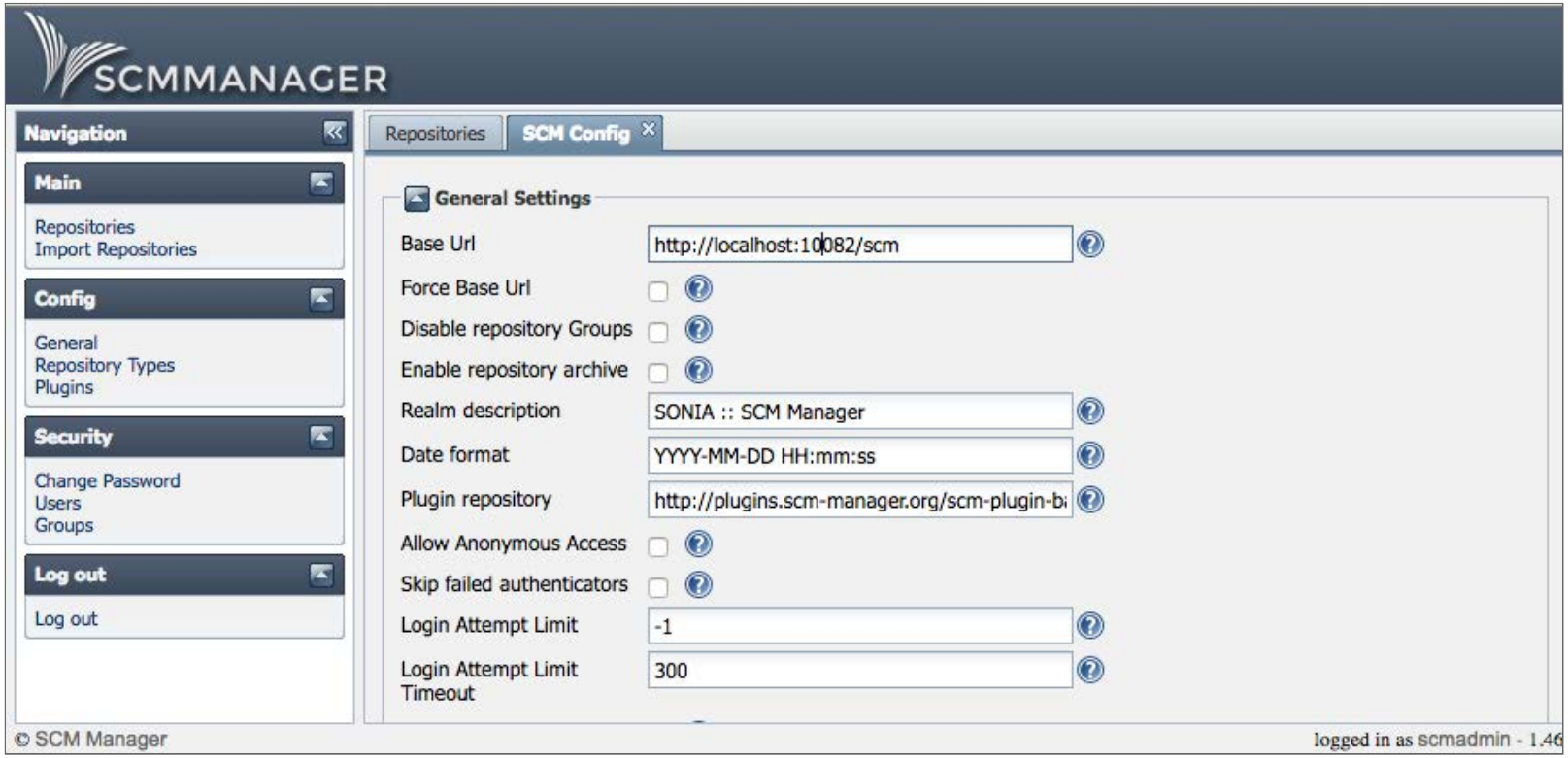
CONFIGURATION MANAGEMENT



JENKINS


Jenkins
create-demo-jobs
configuration
Build History
trend
find
#3 Feb 22, 2016 10:59
#2 Feb 21, 2016 6:15 PM
#1 Feb 21, 2016 5:54 PM
RSS for all
RSS for failures
Execute concurrent builds if necessary
Advanced Project Options
Advanced...
Source Code Management
None
CVS
CVS Projectset
Git
Repositories
Repository URL
https://github.com/devopssquare/jenkins-der
Credentials
- none -
Add
Advanced...
Save
Apply

SCM-MANAGER


SCMMANAGER
Navigation
Main
Repositories
Import Repositories
Config
General
Repository Types
Plugins
Security
Change Password
Users
Groups
Log out
Log out
Repositories
SCM Config
General Settings
Base Url
http://localhost:10082/scm
Force Base Url
Disable repository Groups
Enable repository archive
Realm description
SONIA :: SCM Manager
Date format
YYYY-MM-DD HH:mm:ss
Plugin repository
http://plugins.scm-manager.org/scm-plugin-b
Allow Anonymous Access
Skip failed authenticators
Login Attempt Limit
-1
Login Attempt Limit Timeout
300
© SCM Manager
logged in as scmadmin - 1.46

SO NICHT: PET

BESSER SO: CATTLE

CM SYSTEMATISCH: WAS FEHLT?

- Configuration Management am "lebenden Herzen"
- Feedback-Loop ist sehr kurz!
- Feedback-Loop ist sehr lang!?
- "Blood, Sweat, and Tears"-Management
- Keine systematische Qualitätssicherung!

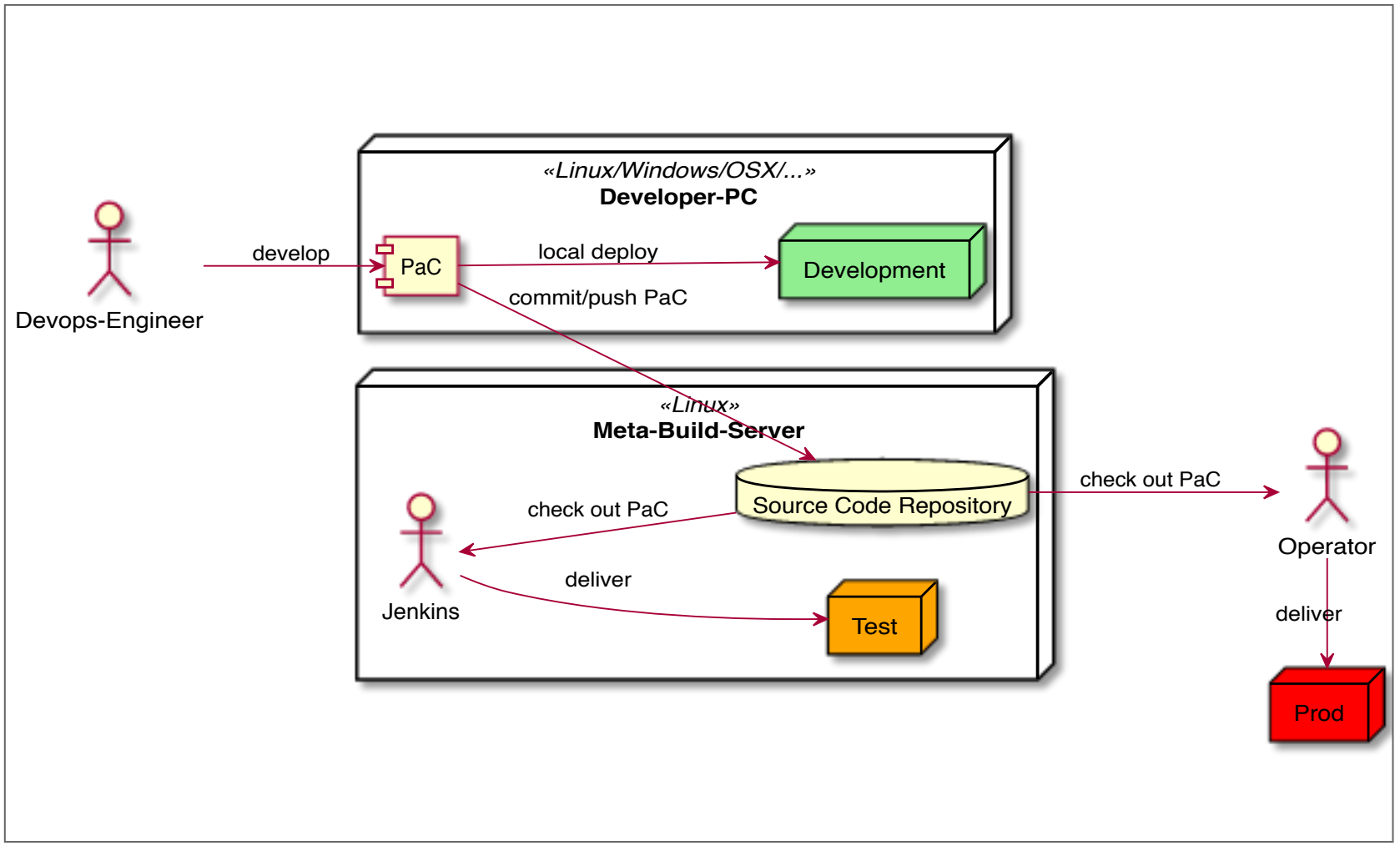
CM EBENEN

- *Infrastruktur*: **OS/Linux** (Windows, OSX, Android, Embedded)
- *Plattform*:
  - Build Server (**Jenkins**, Bamboo, Travis, ...)
  - Source Code Repo (Apache/SVN, **SCM-Manager**, Gitlab, Github, ...)
  - Artifact Repo (**Nexus**, Artifactory, ...)
  - QS-Management (Findbugs/Checkstyle/PMD, Surefire/Failsafe, ... **Sonar Qube**)
- *Application*: **Maven**, Gradle, ...

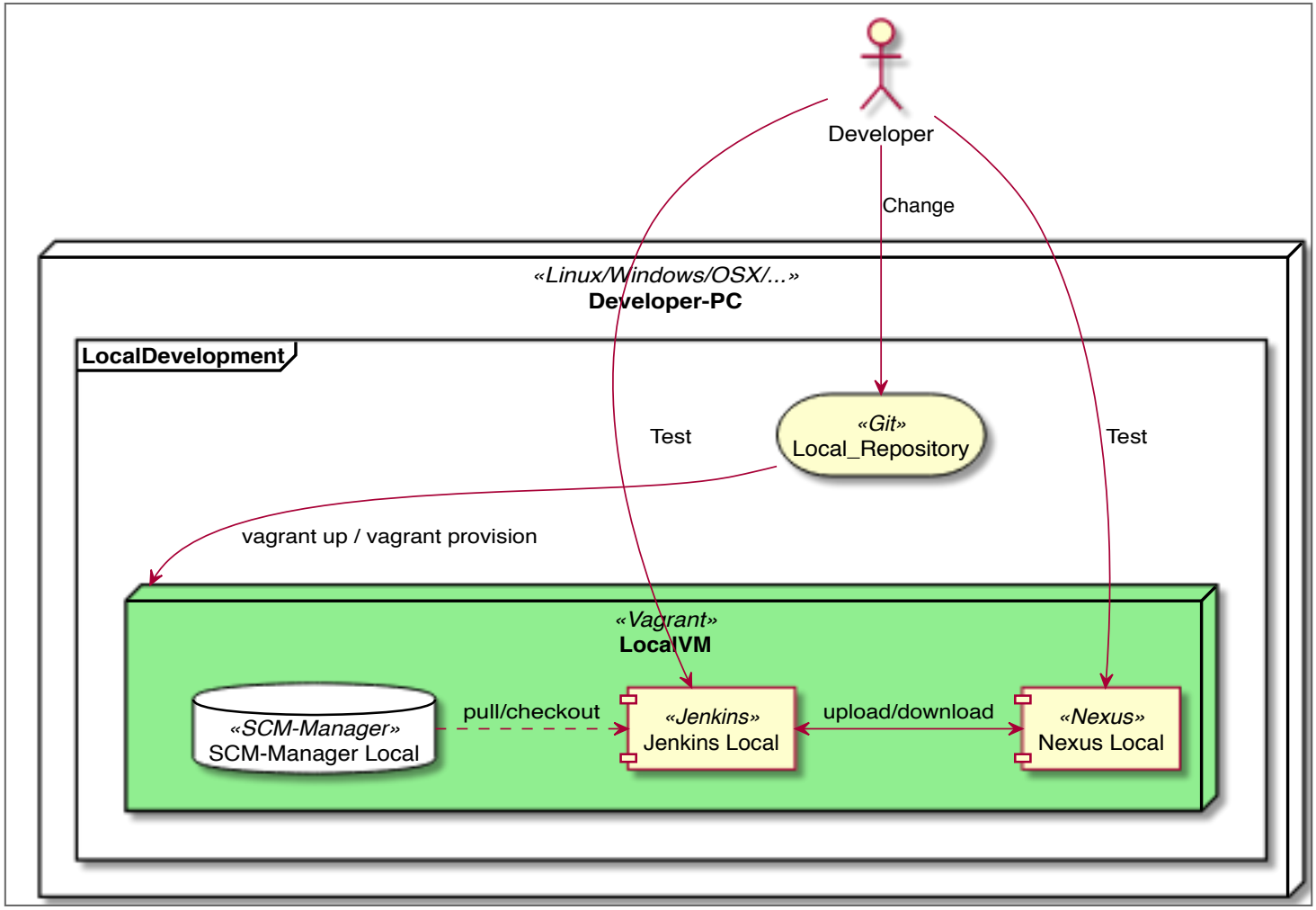
PLATFORM AS CODE (PAC)

- Vagrant
- Puppet (+ Shell + Perl)
- Jenkins (+ Maven)
- Docker/Nexus

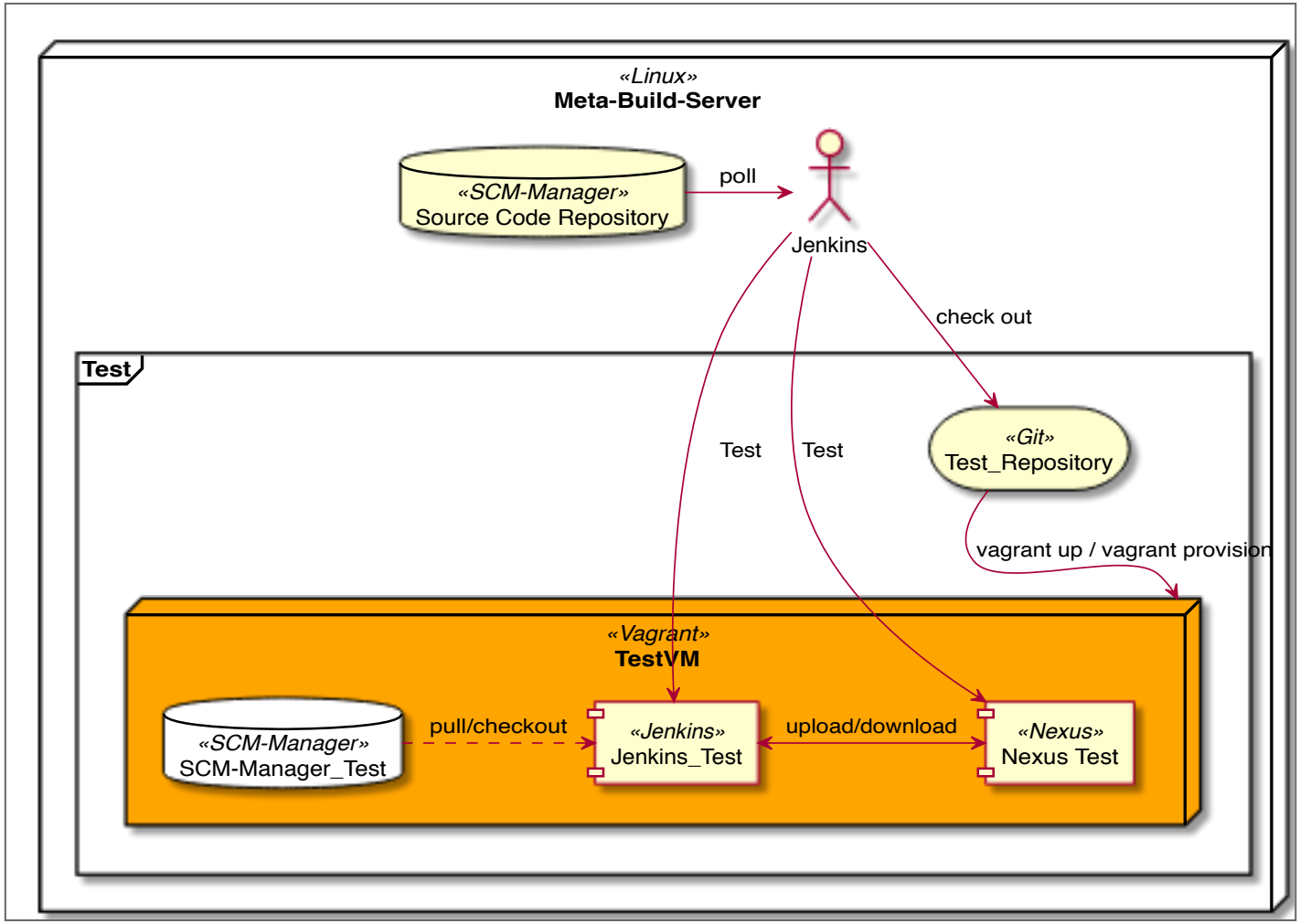
ZIEL-ARCHITEKTUR (VEREINFACHT)



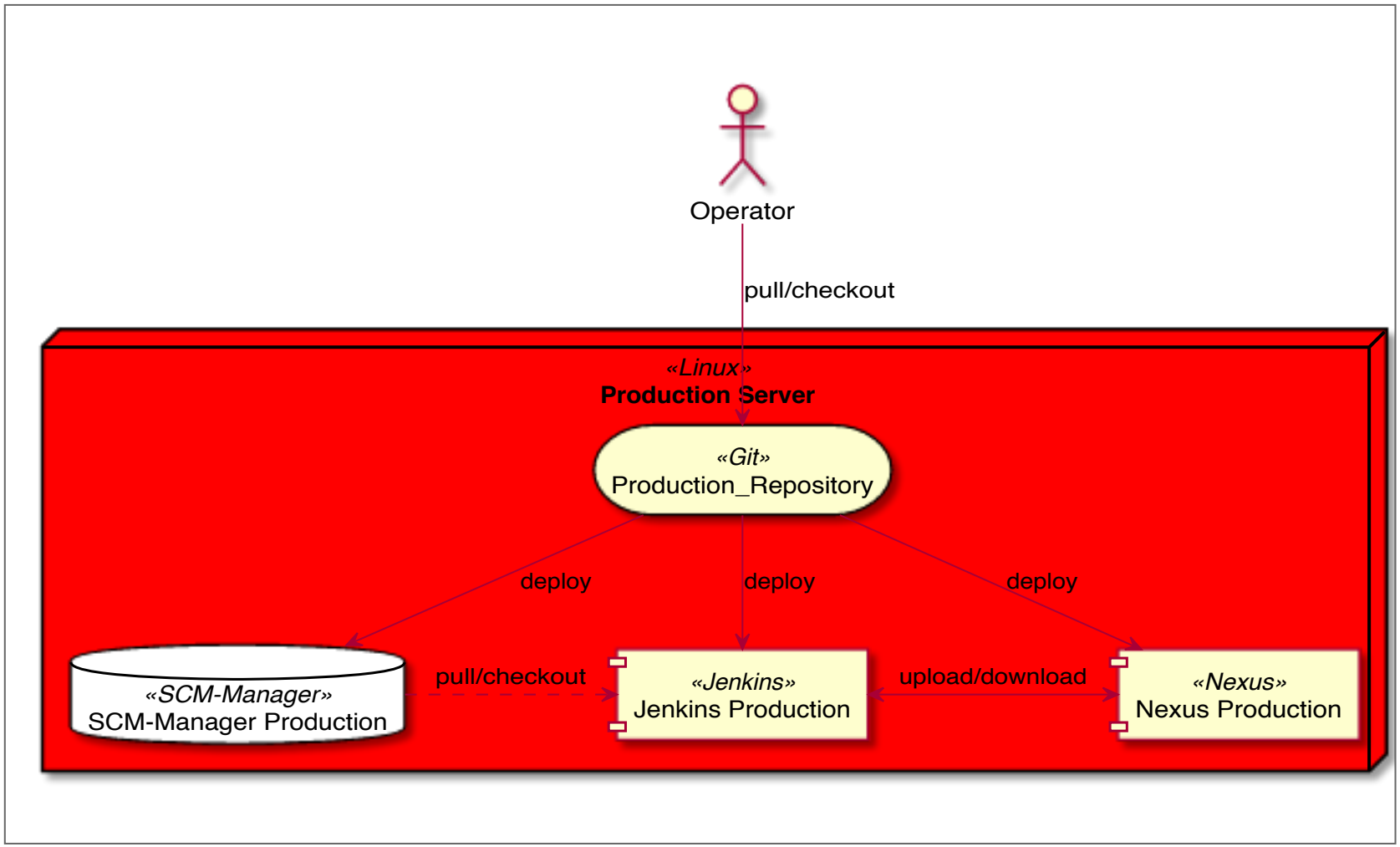
DETAIL-VIEW: DEVELOPER



DETAIL-VIEW: META-BUILD-SERVER



DETAIL-VIEW: PRODUKTION



VAGRANT

- Setup virtueller Maschinen
- Provider
  - **VirtualBox**, libvirt (Qemu/KVM), Parallels, VMware, ...
  - Cloud: AWS
- Provisioner
  - **Shell**
  - Infrastruktur CM-Tools: **Puppet**, Chef, Ansible, SaltStack?, ...

VAGRANTFILE

- Ruby-Syntax
- Provider + Box
- Konfiguration
  - Name
  - Speicher
  - Netzwerk (Port forwards)
  - Dateisystem(e): `/vagrant`
- Provisioner

VAGRANTFILE (DEMO)

```
Vagrant.configure(2) do |config|
  # ...
  config.vm.box = "ubuntu/trusty64"
  config.vm.provider "virtualbox" do |vbox, override|
    vbox.name = "devopssquare-jfs2016"
    vbox.memory = 2048
    override.vm.network "private_network", ip: "192.168.50.17", virtualbox__intnet: tru
    # Jenkins
    override.vm.network "forwarded_port", guest: 8080, host: "17080"
    # Nexus
    override.vm.network "forwarded_port", guest: 8081, host: "17081"
    ...
```

MODULARES SETUP

- Module
- Kompositionen

EXKURS: PUPPET

- Eigentlich Master/Slave-CM (Polling)
- CM-Logik in sog. Manifesten
- Puppet-Module
- Hier: Rein lokale Nutzung
  - Modul-Installation in VM
  - Manifest für Provisionierung

BASE

- `modules/(base/)`
  - `scripts/init.sh`
  - `puppet/init.pp`

BASE/SCRIPTS/INIT.SH

- Install puppet
- Install puppet modules
  - stdlib
  - etckeeper

INIT.SH (DEMO)

```
...
dir=`dirname $0`

basedir="${dir}/.."

$sudo apt-get update
$sudo apt-get install -qq puppet

test -r /etc/puppet/modules/etckeeper || $sudo puppet module install thomasvandoren-etc
test -r /etc/puppet/modules/stdlib || $sudo puppet module install puppetlabs-stdlib

# Avoid warning from initial puppet run
test -r /etc/puppet/hiera.yaml || $sudo touch /etc/puppet/hiera.yaml

$sudo puppet apply "$basedir/puppet/init.pp"
```

BASE/PUPPET/INIT.PP

- Purge packages (NFS, ...)
- Install packages (git, perl-test, ...)
- Setup hiera
- Setup etckeeper!

INIT.PP (DEMO)

```
# Delete some default packages ...
package { 'nfs-common':
  ensure  => "purged"
}
package { 'rpcbind':
  ensure  => "purged"
}

# The minimal set of packages we would like to see!
package { "git": }
package { "screen": }
package { "apticron": }

# Some Packages required for testing
package { "libwww-mechanize-perl": }
package { "libtest-html-content-perl": }
#...
# Tweak etckeeper
file_line { 'etckeeper:git':
  path    => '/etc/etckeeper/etckeeper.conf',
  line    => 'VCS="git"',
}
file_line { 'etckeeper:no-nightly-commit':
  path    => '/etc/etckeeper/etckeeper.conf',
  line    => 'AVOID_DAILY_AUTOCOMMITS=1',
```

COMPOSITES

- composites/
  - scripts/run.sh
  - lists/minimal

RUN.SH (DEMO)

```
dir=`dirname $0`

modulesdir=/vagrant/modules
test -d ${modulesdir} || modulesdir=${dir}/../../modules

compositesdir=/vagrant/composites
test -d ${compositesdir} || compositesdir=${dir}/..

for comp in "$*"
do
    echo "Applying '${comp}'"
    modules=`cat ${compositesdir}/lists/${comp}`

    for module in ${modules}
    do
        ${modulesdir}/${module}/scripts/init.sh
    done
done
```

`VAGRANTFILE`: ERWEITERUNGEN/TIPPS

- Environment
  - Boxname
  - Memory
  - IP/Port Ranges
- Caching + Persistenz

EXTENDED VAGRANTFILE

```
name = ENV['VAGRANT_NAME'] || "devopssquare-jfs2016-devel"
memory = ENV['VAGRANT_MEMORY'] || 2048
ip_unique = ENV['VAGRANT_IP_UNIQUE'] || "17"
port_unique = ENV['VAGRANT_PORT_UNIQUE'] || "#{ip_unique}"
...
  config.vm.hostname = name
  config.vm.synced_folder "cache/apt-archives", "/var/cache/apt/archives"
  config.vm.provider "virtualbox" do |vbox, override|
    vbox.name = name
    vbox.memory = memory
    override.vm.network "private_network", ip: "192.168.50.#{ip_unique}", virtualbox__i
    ...
    override.vm.network "forwarded_port", guest: 8055, host: "#{port_unique}055"
    ...
```

JENKINS

- Neu: `modules/jenkins-native`
  - `scripts/init.sh`: Wie gehabt
  - `puppet/init.pp` !!!
- `composite/lists`: `jenkins-only`
- NEU: `scripts/test.pl`

JENKINS PUPPET (DEMO)

```
include jenkins

# Jenkins needs some packages for misc. Jobs
$packages = [
  'git',
]

package { $packages :
  ensure        =>        installed,
}

$plugins = [
  'ansicolor',
  'build-pipeline-plugin',
  'conditional-buildstep',
  'credentials',
  'disk-usage',
  'envinject',
  'git',
  'git-client',
  'github',
  'github-api',
  'javadoc',
  'jobConfigHistory',
```

JENKINS TEST (DEMO)

```
#!/usr/bin/env perl

use strict;
use warnings;

use WWW::Mechanize;
use Test::HTML::Content (tests => 1);

# Create a new mechanize object
my $mech = WWW::Mechanize->new();
my $url = 'http://admin:admin@localhost:8080/';
# Associate the mechanize object with a URL
$mech->get($url);
# Test for the footer in the content of the URL
xpath_ok($mech->content, '/html/body/footer', 'Jenkins HTML contains footer');
```

⚠ Das ist ein Unit-Test!

❗ Ist das ein Unit-Test?

JENKINS INTEGRATION TEST

- Neu: `modules/it-jenkins-minimal`
- `scripts`, `composites`: Wie gehabt
- `puppet/init.sh`
  - Seed Job (Jenkins Job DSL)
  - HelloWorld
- Integrations-Test: Wurden die Jobs ausgeführt?

SEED JOB (INTEGRATION TEST)

```
mavenJob ("helloworld-install") {
    scm {
        github ("devopssquare/helloworld", "master")
    }
    triggers {
        scm("H/10 * * * *")
    }
    goals ('clean install')
}
```

JENKINS INTEGRATION TEST (DEMO)

```
#!/usr/bin/env perl

use strict;
use warnings;

use WWW::Mechanize;
use Test::HTML::Content (tests => 2);

# Create a new mechanize object
my $mech = WWW::Mechanize->new();

# First test case
my $url1 = 'http://admin:admin@localhost:8080/job/helloworld-seed';
# Associate the mechanize object with a URL
$mech->get($url1);
# Test for the footer in the content of the URL
xpath_ok($mech->content, '/html/body/footer', 'Jenkins Job "helloworld-seed" HTML con

my $url2 = 'http://admin:admin@localhost:8080/job/helloworld-install';
# Associate the mechanize object with a URL
$mech->get($url2);
# Test for the footer in the content of the URL
xpath_ok($mech->content, '/html/body/footer', 'Jenkins Job "helloworld-install" HTML
```

INTERMEZZO: DOCKER

- Komponenten ohne *viel* Konfiguration (→ Plug'n'Play):
  - Nexus
  - SCM-Manager
  - SonarQube (+ PostgreSQL)
- Volumes + Volume-Container für Persistenz!

DOCKER SETUP

- Neu: `modules/data`
- Neu: `modules/docker`

```
class { 'docker': }
```

NEXUS ALS DOCKER-CONTAINER

- Neu: `modules/nexus`

```
docker::run { 'nexus':
  image    => 'sonatype/nexus',
  volumes  => ['/data/nexus/sonatype:/sonatype-work'],
  ports    => ['8081:8081'],
  require  => File['/data/nexus/sonatype']
}

file { '/data/nexus':
  ensure   => directory,
  owner    => root,
  group    => root,
  mode     => 0777,
}
file { '/data/nexus/sonatype':
  ensure   => directory,
  owner    => root,
  group    => root,
  mode     => 0777,
  require  => File['/data/nexus']
}
```

❗ Puppet Docker-Module erzeugt OS-Service (Nexus)

JENKINS + NEXUS INTEGRATION TEST

- Neu: `modules/jenkins-settings-nexus-localhost`
- Neu: `modules/it-jenkins-nexus-minimal`
- Neu: `composites/lists/it-jenkins-nexus-minimal`

USE CASES

- Kontinuierliche Fortentwicklung der CDP
- Automatisierter regelmäßiger (Neu) Aufbau der CDP
- Testen neuer Komponenten(versionen)
- Testen neuer Prozesse
- Schulung/Einarbeitung

ENVIRONMENTS / CD STAGES

- Development/Continuous Integration (Automatisierte Tests)
- User Acceptance Test (Exploratives Testen)
- (Performance/Exception/... Tests)
- Produktion

OFFENE FRAGEN

- Was ist mit dem Zustand (in Produktion)?
  - Administrativer Zustand → **PaC**
  - Operativer Zustand → *it depends*
    - Repositories
    - Alte Instanz(en) vorhalten?
    - Backup?
- Partielle Deployments?
- Blue/Green-Deployments?

FAZIT

- CD-Plattform als PaC realisiert
- Nutzung von Virtualisierung und Containern
- Modularisierung und Aggregation
- Puppet ist oft *hakelig* (Ansible auch!)
- Ausblick:
  - Feedback-Loops/Monitoring fehlt
  - Unabhängigkeit von Puppet/Ansible/Salt/Chef???

LINKS

- Beispiele:
  - **https://github.com/devopssquare/demo-jsf2016**
  - **https://github.com/devopssquare/helloworld**
  - **https://github.com/devopssquare/helloworld-seed**
- Slides: **http://aschemann.net/gerd/publications/cdofcd-jsf16.pdf**

DANKE!