

# Automate all the things



Sebastian Feldmann



"Automate until the job is boring as hell"

# System Configuration

Software packaging

Software deployment

Combining all to work seamlessly together

# System Configuration



ANSIBLE

*“App deployment, configuration management and orchestration  
all from one system. Powerful automation that you can learn quickly.”*



# About Ansible

- Written in Python
- Using SSH
- No agents
- YAML

# Ansible Modules

- Package management (apt, yum...)
- Command execution (command, shell)
- Service management (start, stop...)
- File handling (copy, template...)
- SCM (git, Bazaar, Subversion)

Inventory

Roles

Playbook

# System configuration with Ansible

# Inventory

```
ops/  
+- inv/  
  +- integration/  
  +- local/  
  +- production/
```

good practice inventory setup

# Inventory

```
[webserver]  
192.168.1.111
```

```
[dbserver]  
192.168.1.222  
192.168.1.223
```

ops/inv/integration/hosts



# Playbook

```
- hosts: webservers
  roles:
    - apache
```

ops/install.webservers.yml

# Role

```
ops/  
+- roles/  
| +- apache/  
|   +- files/  
|   +- handlers/  
|   +- tasks/  
|   +- templates/  
+- install.webserver.yml
```

good practice role setup

# Tasks

```
- name: install apache
  action: apt
          pkg=apache2
          state=present
          force=yes

- name: copy mpm prefork conf
  copy: dest=/etc/apache2/mods-available/mpm_prefork.conf
        src=mpm_prefork.conf
  notify: restart apache
```

ops/roles/apache/tasks/main.yml

# Tasks

```
- name: install apache
  action: apt
          pkg=apache2
          state=present
          force=yes

- name: copy mpm prefork conf
  copy: dest=/etc/apache2/mods-available/mpm_prefork.conf
        src=mpm_prefork.conf
  notify: restart apache
```

ops/roles/apache/tasks/main.yml

# Tasks

```
- name: install apache
  action: apt
    pkg=apache2
    state=present
    force=yes

- name: copy mpm prefork conf
  copy: dest=/etc/apache2/mods-available/mpm_prefork.conf
        src=mpm_prefork.conf
  notify: restart apache
```

ops/roles/apache/tasks/main.yml

# Tasks

```
- name: install apache
  action: apt
    pkg=apache2
    state=present
    force=yes

- name: copy mpm prefork conf
  copy: dest=/etc/apache2/mods-available/mpm_prefork.conf
        src=mpm_prefork.conf
  notify: restart apache
```

ops/roles/apache/tasks/main.yml

# Handler

```
- name: restart apache  
  action: service name=apache2 state=restarted
```

ops/roles/apache/handlers/main.yml

# Handler

```
- name: restart apache  
  action: service name=apache2 state=restarted
```

ops/roles/apache/handlers/main.yml



# Recap

- Create inventories
- Install playbook
  - Apache role
    - Install tasks / handlers

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts install.webserver.yml
```

execution example

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts install.webserver.yml
```

execution example

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts install.webserver.yml
```

execution example

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts install.webserver.yml
```

execution example

PLAYBOOK: install.webserver.yml

\*\*\*\*\*

1 plays in ansible/install.webserver.yml

PLAY [webserver]

\*\*\*\*\*

TASK [apache : install apache]

\*\*\*\*\*

ok: [192.168.1.111]

TASK [apache-c24 : copy mpm prefork conf]

\*\*\*\*\*

ok: [192.168.1.111]

NOTIFIED HANDLER restart apache

PLAY RECAP \*\*\*\*\*

192.168.1.111 : ok=2 changed=2 unreachable=0 failed=0

Finished: SUCCESS

PLAYBOOK: install.webserver.yml

\*\*\*\*\*

1 plays in ansible/install.webserver.yml

PLAY [webserver]

\*\*\*\*\*

TASK [apache : install apache]

\*\*\*\*\*

ok: [192.168.1.111]

TASK [apache-c24 : copy mpm prefork conf]

\*\*\*\*\*

ok: [192.168.1.111]

NOTIFIED HANDLER restart apache

PLAY RECAP \*\*\*\*\*

192.168.1.111 : ok=2 changed=0 unreachable=0 failed=0

Finished: SUCCESS

# Execute Ansible

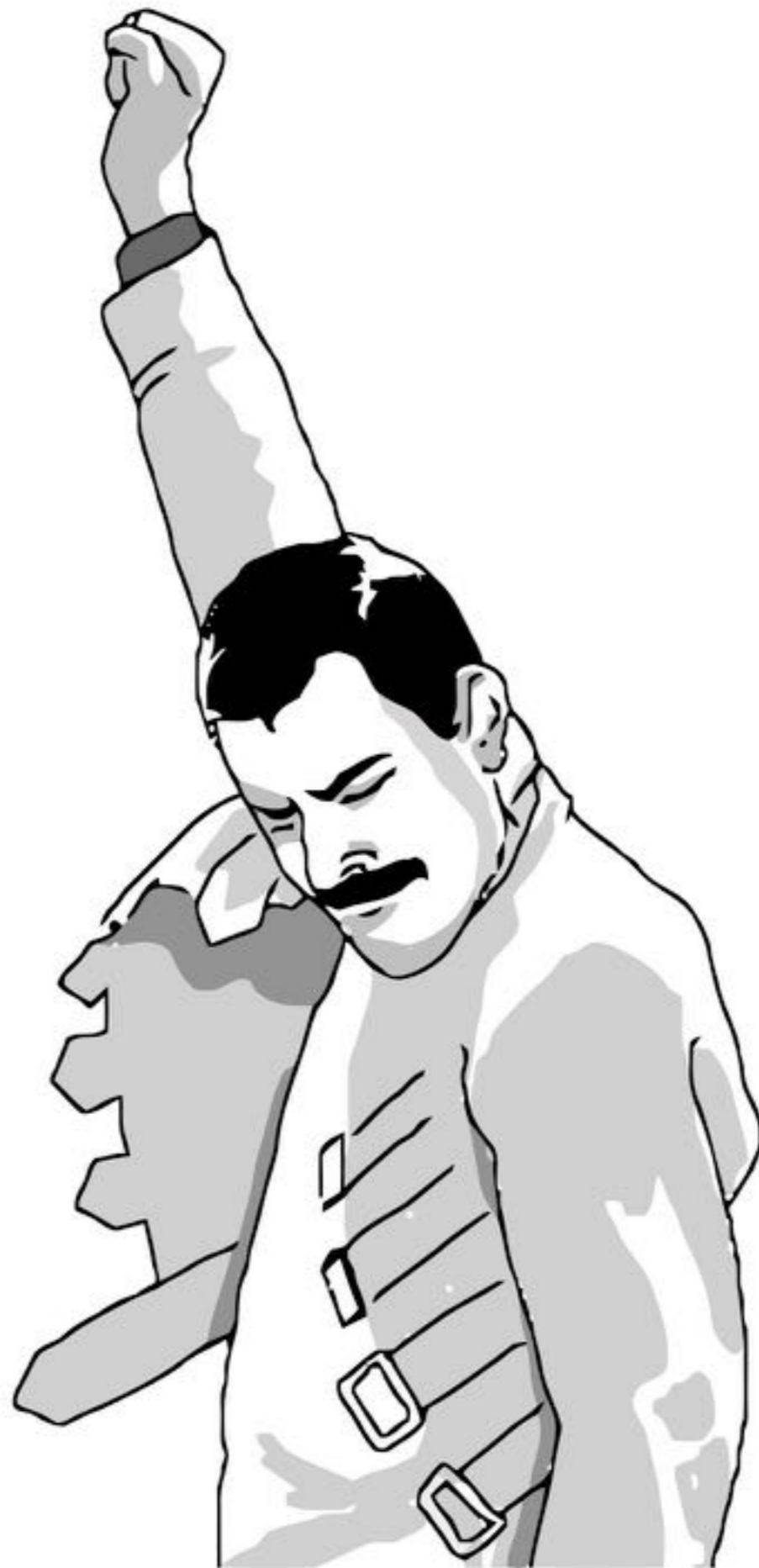
```
$ ansible-playbook -i inv/integration/hosts --check install.webserver.yml
```

dry run with --check

```
$ ansible-playbook -i inv/integration/hosts --limit 192.168.1.111 install.webserver.yml
```

restrict execution with --limit





# Software Packaging



"Ant can be used to pilot any type of process which can be described in terms of targets and tasks"

# Ant

- Written in Java
- XML configuration
- Targets (tasks)
- Bundled with Java

# Software packaging with Ant

# Ant Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="projectx" default="build">
  <target name="build" depends="target1, target2, target3" />
  <target name="target1">...</target>
  <target name="target2">...</target>
  ...
</project>
```

build.xml

# Ant Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="projectx" default="build">
  <target name="build" depends="target1, target2, target3" />
  <target name="target1">...</target>
  <target name="target2">...</target>
  ...
</project>
```

build.xml

# Ant Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="projectx" default="build">
  <target name="build" depends="target1, target2, target3" />
  <target name="target1">...</target>
  <target name="target2">...</target>
  ...
</project>
```

build.xml



# Ant Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="projectx" default="build">
  <target name="build" depends="target1, target2, target3" />
  <target name="target1">...</target>
  <target name="target2">...</target>
  ...
</project>
```

build.xml

# Composer

```
<target name="composer-install">  
  <exec executable="composer" failonerror="true">  
    <arg value="install" />  
    <arg value="--no-interaction" />  
  </exec>  
</target>
```

build.xml

# Composer

```
<target name="composer-install">  
  <exec executable="composer" failonerror="true">  
    <arg value="install" />  
    <arg value="--no-interaction" />  
  </exec>  
</target>
```

build.xml

# Composer

```
<target name="composer-install">  
  <exec executable="composer" failonerror="true">  
    <arg value="install" />  
    <arg value="--no-interaction" />  
  </exec>  
</target>
```

build.xml

# Composer

```
<target name="composer-install">  
  <exec executable="composer" failonerror="true">  
    <arg value="install" />  
    <arg value="--no-interaction" />  
  </exec>  
</target>
```

build.xml

# Composer

```
<target name="composer-install">  
  <exec executable="composer" failonerror="true">  
    <arg value="install" />  
    <arg value="--no-interaction" />  
  </exec>  
</target>
```

build.xml

# PHPUnit

```
<target name="phpunit" description="Run unit tests">  
  <exec executable="phpunit" failonerror="true"></exec>  
</target>
```

build.xml

# PHPUnit

```
<target name="phpunit" description="Run unit tests">  
  <exec executable="phpunit" failonerror="true"></exec>  
</target>
```

build.xml



# PHPUnit

```
<target name="phpunit" description="Run unit tests">  
  <exec executable="phpunit" failonerror="true"></exec>  
</target>
```

build.xml

# PHPUnit

```
<target name="phpunit" description="Run unit tests">  
  <exec executable="phpunit" failonerror="true"></exec>  
</target>
```

build.xml

# Ant Tasks

- Minify Assets
- Cache warmup (Templates, DIContainer...)
- ...

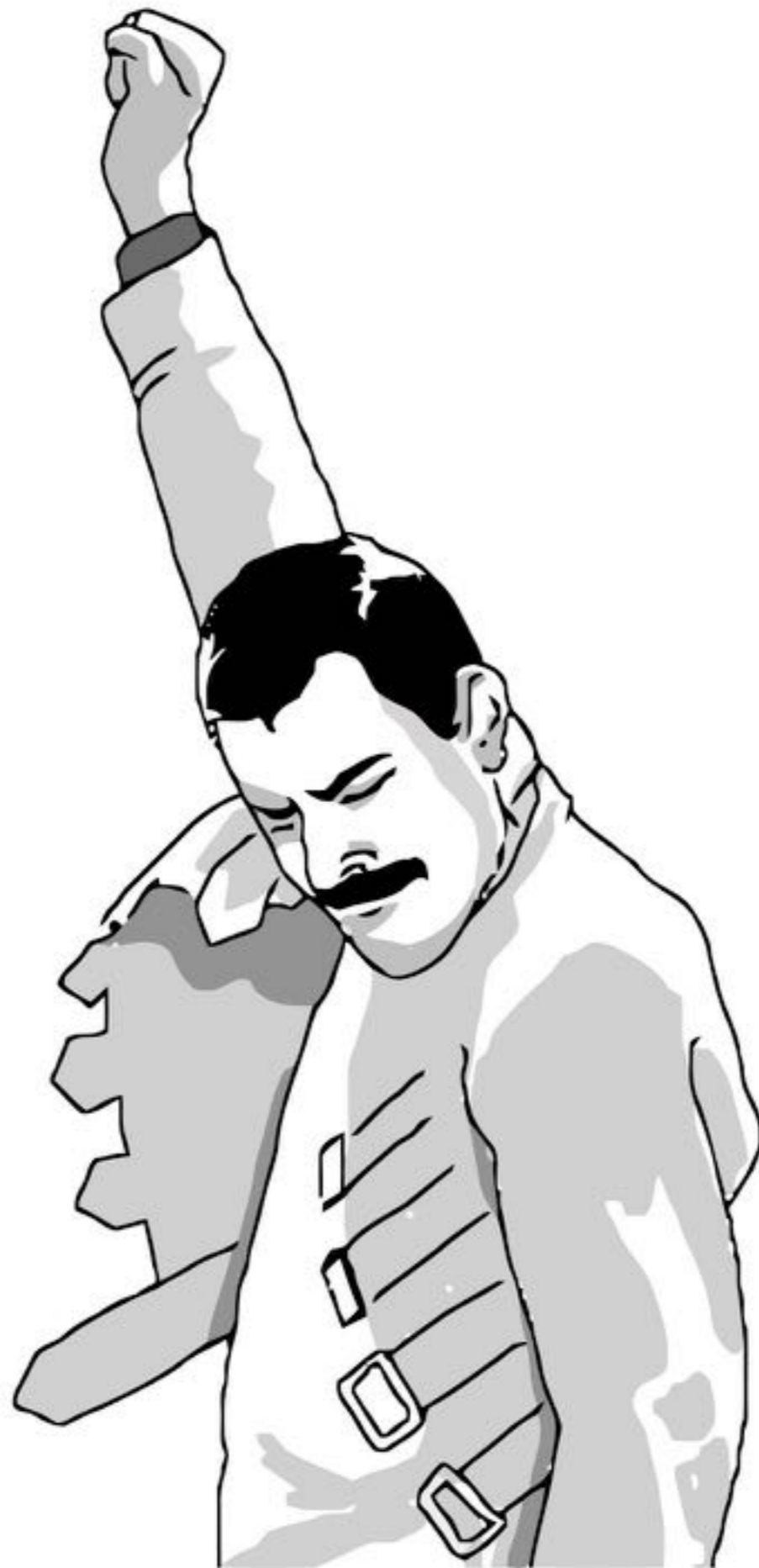
# Ant Execution

```
$ ant
```

execute default target

```
$ ant phpunit
```

execute specific target



# Software Deployment

# Software deployment with Ansible

# Software Deployment

- Copy app to server
- Activate deployed app
- Allow quick rollbacks



# Half Way There

- Enhance the inventory
- New playbook
- New role

# Inventory

```
ops/  
+- inv/  
  +- integration/  
    +- group_vars/  
    +- host_vars/
```

add variables

# Variables

```
system:  
  environment: integration  
  
projectx:  
  domain: www.projectx.int
```

ops/inv/integration/group\_vars/webserver

# Playbook

```
- hosts: webservers
  roles:
    - projectx
```

ops/deploy.projectx.yml

# Role

```
ops/  
+- roles/  
|   +- projectx/  
|     +- files/  
|     +- handler/  
|     +- tasks/  
|     +- templates/  
+- deploy.projectx.yml
```

best practice role setup

# Tasks

```
# create the directory for the current app version
- name: Create version directory
  file: path=/var/www/projectx/{{ version }}
       state=directory

# copy the prepared app to your web servers
- name: Copy files to server
  synchronize: src={{ app }}
               dest=/var/www/projectx/{{ version }}
               perms=yes
               recursive=yes
               delete=yes
               owner=no
               group=no
```

ops/roles/projectx/main.yml

# Tasks

```
# create the directory for the current app version
- name: Create version directory
  file: path=/var/www/projectx/{{ version }}
       state=directory

# copy the prepared app to your web servers
- name: Copy files to server
  synchronize: src={{ app }}
               dest=/var/www/projectx/{{ version }}
               perms=yes
               recursive=yes
               delete=yes
               owner=no
               group=no
```

ops/roles/projectx/main.yml

# Tasks

```
# create the directory for the current app version
- name: Create version directory
  file: path=/var/www/projectx/{{ version }}
        state=directory

# copy the prepared app to your web servers
- name: Copy files to server
  synchronize: src={{ app }}
               dest=/var/www/projectx/{{ version }}
               perms=yes
               recursive=yes
               delete=yes
               owner=no
               group=no
```

ops/roles/projectx/main.yml



# Tasks

```
# create the directory for the current app version
- name: Create version directory
  file: path=/var/www/projectx/{{ version }}
        state=directory

# copy the prepared app to your web servers
- name: Copy files to server
  synchronize: src={{ app }}
                dest=/var/www/projectx/{{ version }}
                perms=yes
                recursive=yes
                delete=yes
                owner=no
                group=no
```

ops/roles/projectx/main.yml

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
        src=projectx.conf
        dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
        src=projectx.conf
        dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
    src=projectx.conf
    dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
        src=projectx.conf
        dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
    src=projectx.conf
    dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Tasks

```
# create a symlink to activate the vhost
# could be done with "command: a2ensite projectx.conf"
# but this would always trigger an apache restart
# even if nothing changes
- name: Activate vhost configuration
  file: dest=/etc/apache2/sites-enabled/010-projectx.conf
       src=/etc/apache2/sites-available/projectx.conf
       state=link
  notify: restart apache
```

ops/roles/projectx/tasks/main.yml

# Tasks

```
# create a symlink to activate the vhost
# could be done with "command: a2ensite projectx.conf"
# but this would always trigger an apache restart
# even if nothing changes
- name: Activate vhost configuration
  file: dest=/etc/apache2/sites-enabled/010-projectx.conf
       src=/etc/apache2/sites-available/projectx.conf
       state=link
  notify: restart apache
```

ops/roles/projectx/tasks/main.yml



# Tasks

```
# create a symlink to activate the vhost
# could be done with "command: a2ensite projectx.conf"
# but this would always trigger an apache restart
# even if nothing changes
- name: Activate vhost configuration
  file: dest=/etc/apache2/sites-enabled/010-projectx.conf
       src=/etc/apache2/sites-available/projectx.conf
       state=link
  notify: restart apache
```

ops/roles/projectx/tasks/main.yml

# Tasks

```
# create a symlink to activate the vhost
# could be done with "command: a2ensite projectx.conf"
# but this would always trigger an apache restart
# even if nothing changes
- name: Activate vhost configuration
  file: dest=/etc/apache2/sites-enabled/010-projectx.conf
       src=/etc/apache2/sites-available/projectx.conf
       state=link
  notify: restart apache
```

ops/roles/projectx/tasks/main.yml

# Tasks

```
# create a symlink to activate the vhost
# could be done with "command: a2ensite projectx.conf"
# but this would always trigger an apache restart
# even if nothing changes
- name: Activate vhost configuration
  file: dest=/etc/apache2/sites-enabled/010-projectx.conf
       src=/etc/apache2/sites-available/projectx.conf
       state=link
  notify: restart apache
```

ops/roles/projectx/tasks/main.yml

# Template

```
<VirtualHost *:80>
    ServerName {{ projectx.domain }}
    DocumentRoot /var/www/projectx/{{ version }}/htdocs

    setenv APP.ENVIRONMENT {{ system.environment }}

    <Directory /var/www/projectx/{{ version }}/htdocs>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

ops/roles/projectx/templates/projectx.conf

# Variables

```
system:  
  environment: integration
```

```
projectx:  
  domain: www.projectx.int
```

ops/inv/integration/group\_vars/webserver

# Template

```
<VirtualHost *:80>
  ServerName {{ projectx.domain }}
  DocumentRoot /var/www/projectx/{{ version }}/htdocs

  setenv APP.ENVIRONMENT {{ system.environment }}

  <Directory /var/www/projectx/{{ version }}/htdocs>
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

ops/roles/projectx/templates/projectx.conf

# Recap

- Inventory variables
- Deploy playbook
- Project role
- Deployment tasks

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution



# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution

# Execute Ansible

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.1 app=/some/local/dir/app/"  
  deploy.projectx.yml
```

deployment execution

# Tasks

```
# deploy the vhost configuration for our project
- name: Deploy vhost configuration
  action: template
    src=projectx.conf
    dest=/etc/apache2/sites-available/projectx.conf
  notify: restart apache
  tags: rollback
```

ops/roles/projectx/main.yml

# Template

```
<VirtualHost *:80>
  ServerName {{ projectx.domain }}
  DocumentRoot /var/www/projectx/{{ version }}/htdocs

  setenv APP.ENVIRONMENT {{ system.environment }}

  <Directory /var/www/projectx/{{ version }}/htdocs>
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

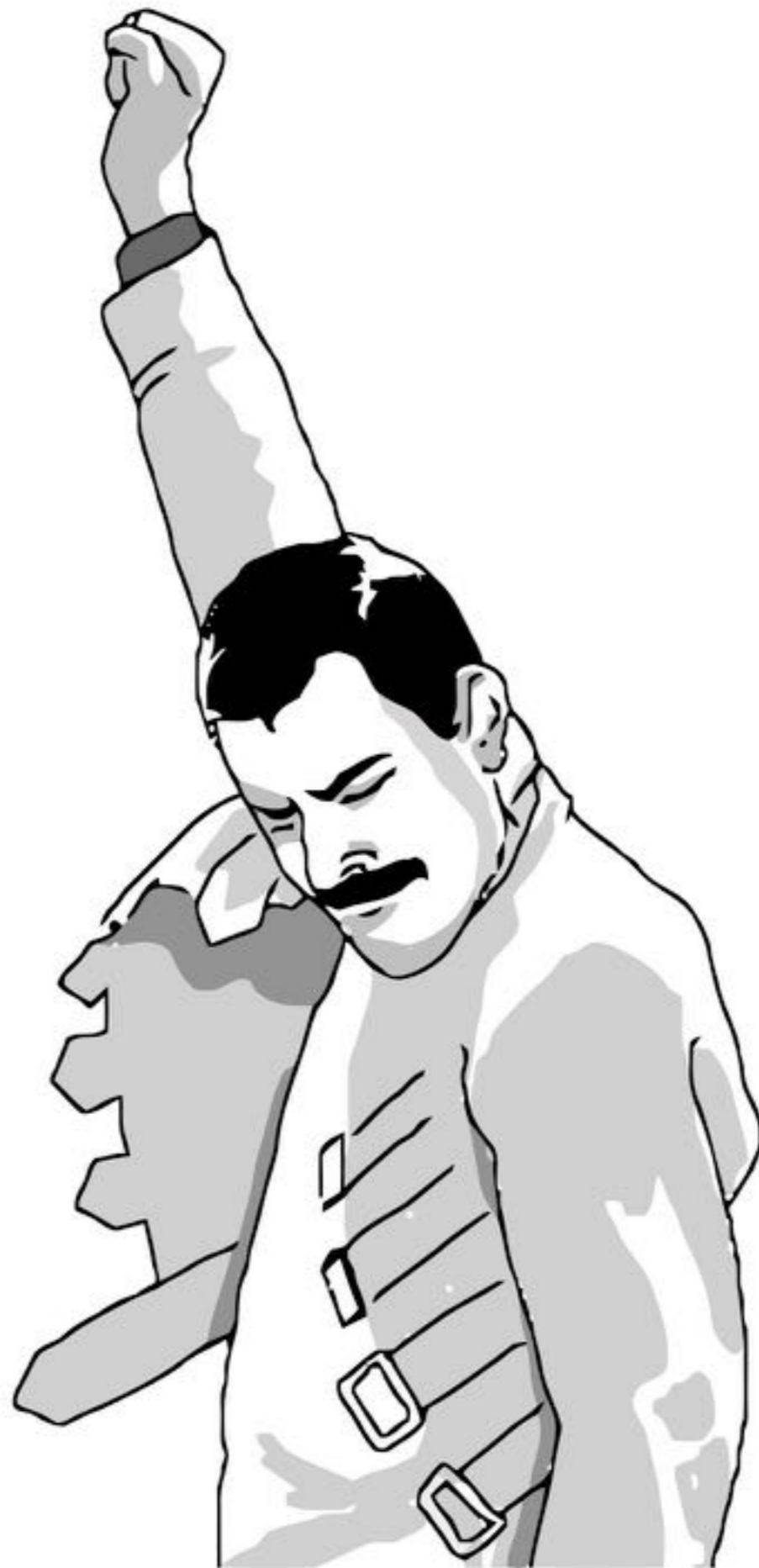
ops/roles/projectx/templates/projectx.conf

# Rollback

```
$ ansible-playbook -i inv/integration/hosts  
  --extra-vars "version=1.0.0"  
  --tags rollback  
  deploy.projectx.yml
```

rollback execution





# System Config

- Clone repository
- Execute Ansible

# Deployment

- Tag version
- Checkout version
- Execute Ant
- Execute Ansible

Not boring!

NOT BAD

But I only want to click a button

And everything should work



# Jenkins

*“Build great things at any scale”*



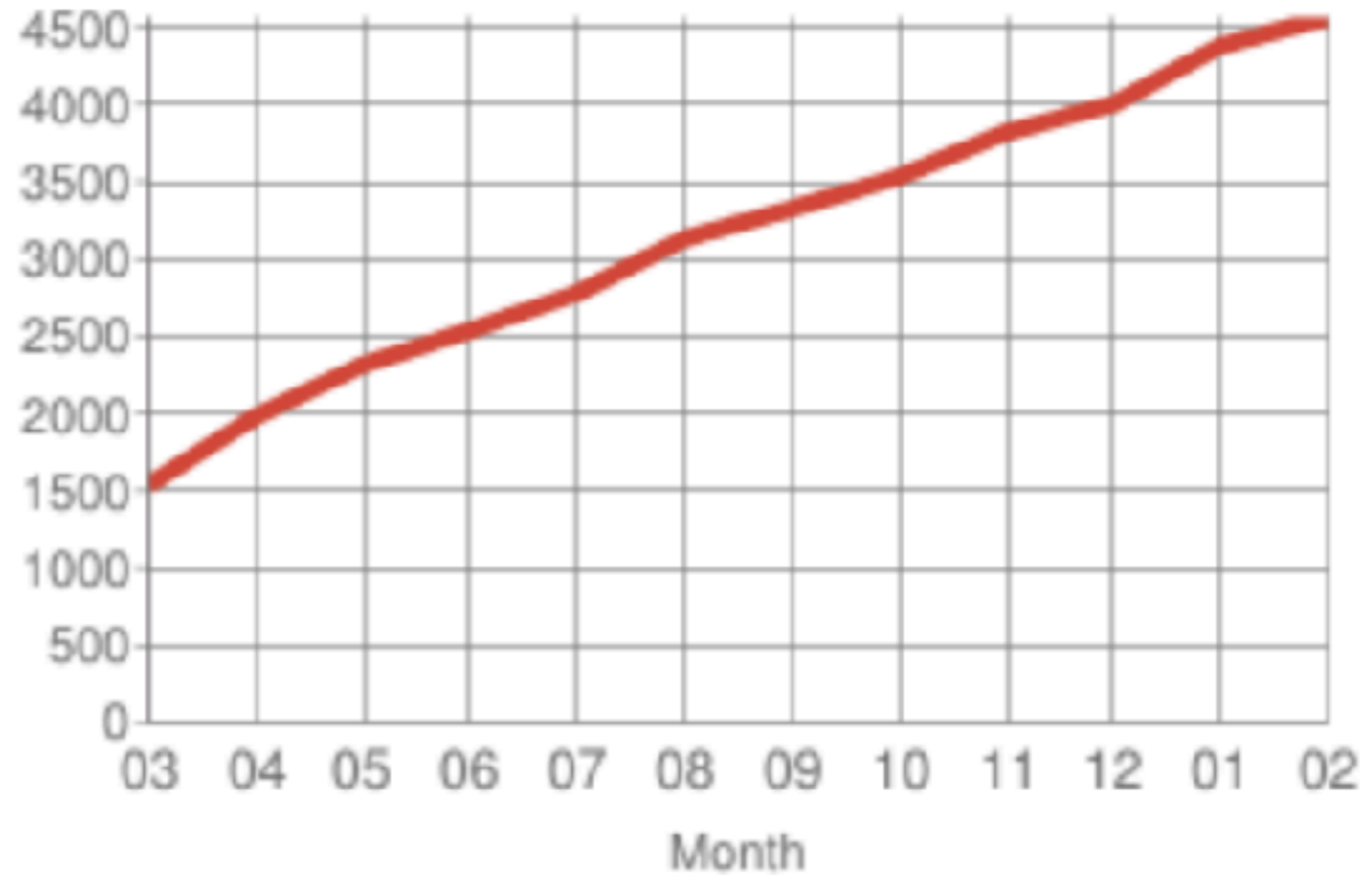
# Jenkins

- Written in Java
- Build server
- Projects / Jobs
- Jetty (build in webserver)

# Ansible Plugin

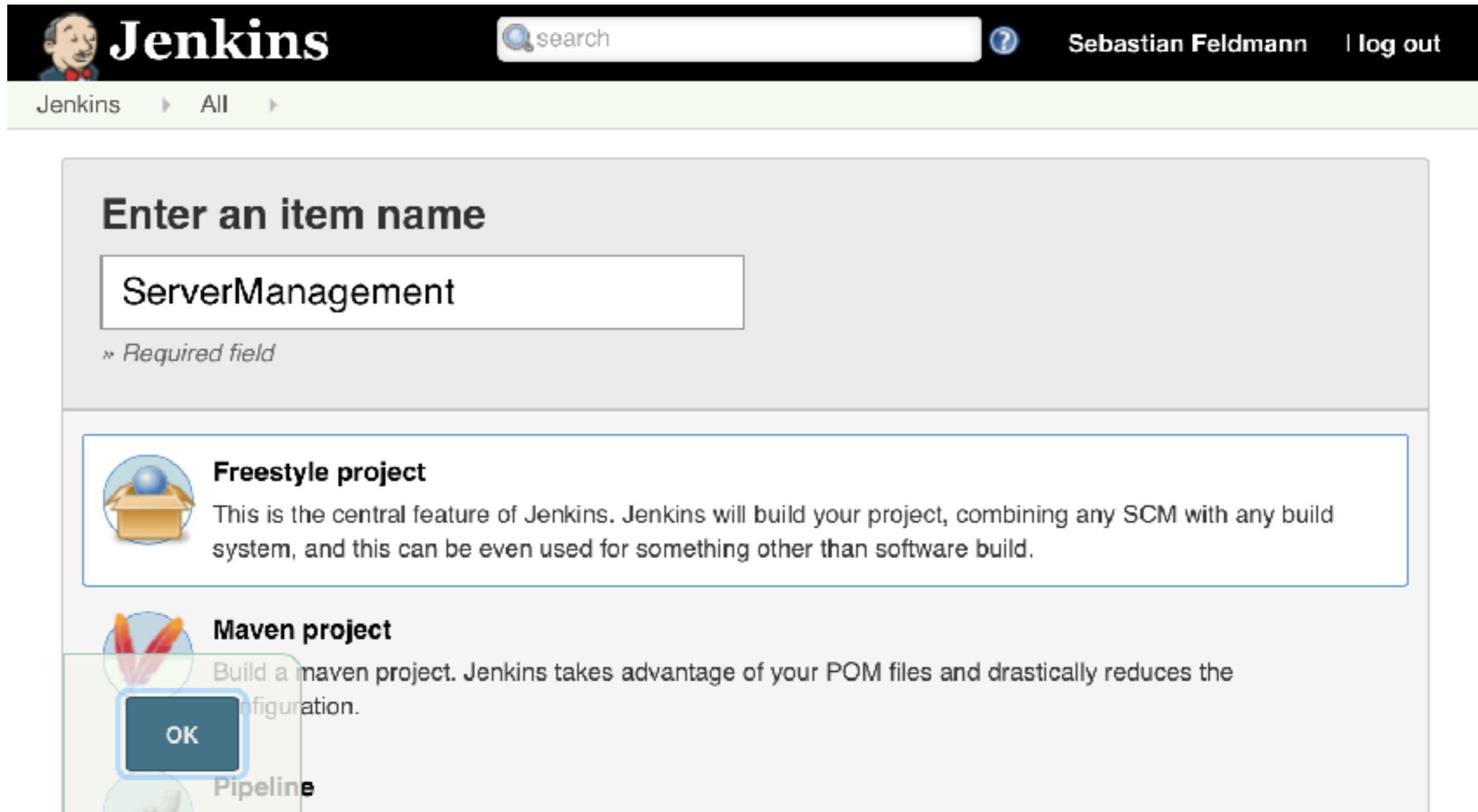
Usage


ansible - installations



System configuration with Ansible and Jenkins

# System Management





 **Jenkins**  [?](#) Sebastian Feldmann | [log out](#)


Jenkins > All >

**Enter an item name**

» *Required field*

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **Pipeline**

# SCM Config

**Source Code Management**

None  
 CVS  
 CVS Projectset  
 Git

Repositories

Repository URL  

use some SCM to manage your Ansible project

# Parameter Setup

The image shows two overlapping windows for configuring 'Choice Parameters'. Each window has a title bar with a close button (X) and a help button (?).

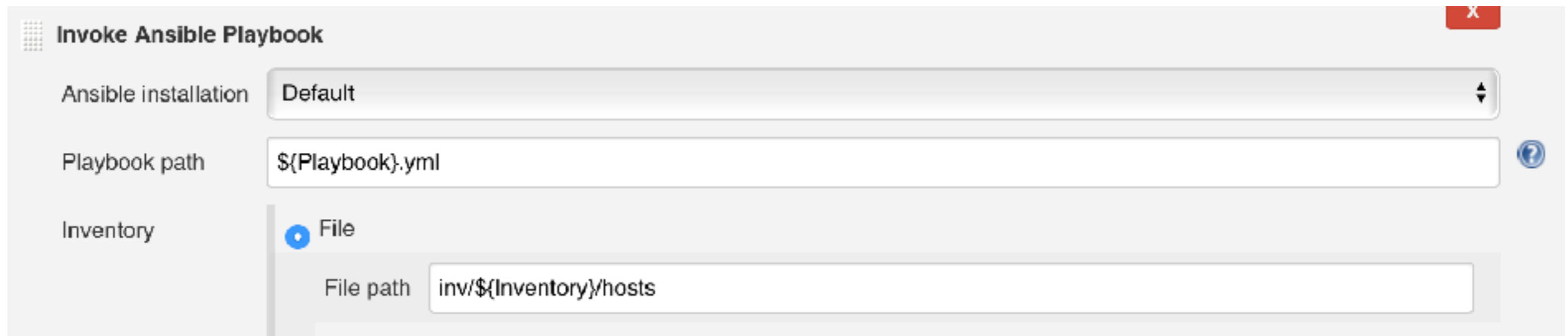
**Top Window: Choice Parameter**

- Name:** Inventory
- Choices:** integration, production
- Description:** (partially obscured)

**Bottom Window: Choice Parameter**

- Name:** Playbook
- Choices:** install.webserver, install.dbserver
- Description:** Choose playbook to execute

# Ansible Plugin Config



The screenshot shows a configuration window titled "Invoke Ansible Playbook". It contains the following fields and options:

- Ansible installation:** A dropdown menu with "Default" selected.
- Playbook path:** A text input field containing "\${Playbook}.yml".
- Inventory:** A section with a radio button selected for "File".
- File path:** A text input field containing "inv/\${Inventory}/hosts".

The use of "\${Playbook}" and "\${Inventory}" in the paths demonstrates parameter usage.

parameter usage

# Job Execution

The screenshot displays the Jenkins web interface. At the top, there is a black navigation bar with the Jenkins logo on the left, a search bar in the center, and the user name 'Sebastian Feldmann' and a 'log out' link on the right. Below this is a light green breadcrumb trail showing 'Jenkins' and 'SystemManagement'. On the left side, there is a vertical sidebar with several menu items, each with an icon: 'Back to Dashboard' (green arrow), 'Status' (magnifying glass), 'Changes' (notepad), 'Workspace' (folder), 'Build with Parameters' (play button), 'Delete Project' (red prohibition sign), 'Configure' (gear), and 'Move' (hand with arrow). The main content area is titled 'Project SystemManagement' and contains the text 'This build requires parameters:'. Below this text are two dropdown menus: 'Inventory' with the value 'integration' and 'Playbook' with the value 'install.webserver'. Each dropdown has a small blue arrow icon on the right. Below the dropdowns is a dark blue 'Build' button.

Jenkins  Sebastian Feldmann | log out

Jenkins > SystemManagement

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Delete Project
- Configure
- Move

## Project SystemManagement

This build requires parameters:

Inventory

Choose inventory to execute

Playbook

Choose playbook to execute

**Build**

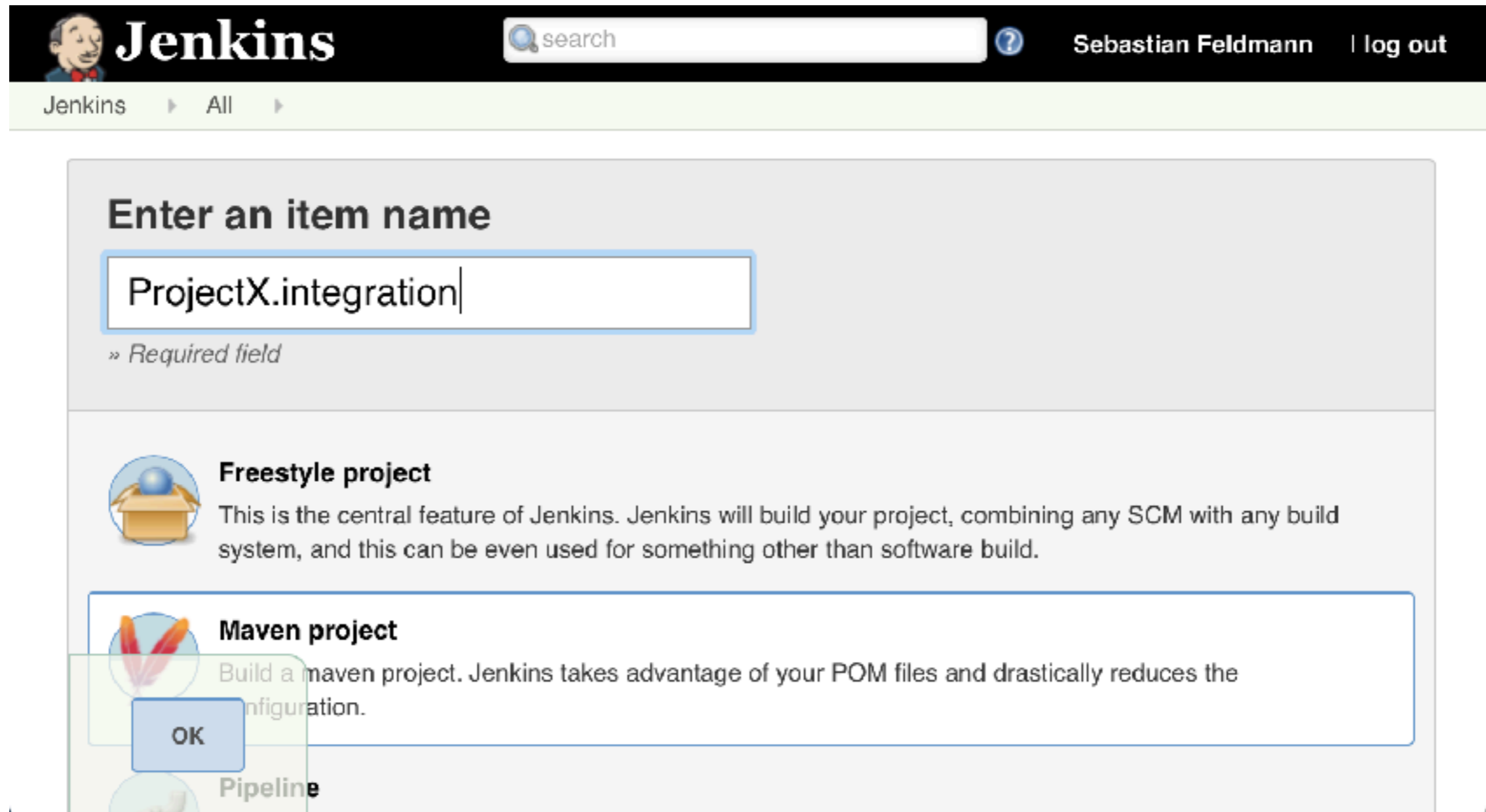


# System Configuration

- Push code to git repository
- Push a button

# Software packaging and deployment with Ant, Ansible and Jenkins

# Software Deployment



The screenshot shows the Jenkins web interface. At the top, there is a black header with the Jenkins logo (a cartoon man) and the word "Jenkins" in white. To the right of the logo is a search bar with the text "search" and a magnifying glass icon. Further right, the user's name "Sebastian Feldmann" and a "log out" link are visible. Below the header is a light green breadcrumb trail: "Jenkins > All >".

The main content area is a light gray box. At the top of this box is the heading "Enter an item name". Below this heading is a text input field containing the text "ProjectX.integration". Below the input field is a small red asterisk and the text "» Required field".

Below the input field are three project type options, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration. (This option is highlighted with a blue border and a blue "OK" button is overlaid on it.)
- Pipeline**: (Partially visible at the bottom)

# Parameter Setup

String Parameter X ?

Name	<input type="text" value="version"/>	?
Default Value	<input type="text"/>	?
Description	<input type="text" value="Version you want to deploy"/>	?

[Safe HTML] [Preview](#)

# SCM Config


**Source Code Management**

None  
 CVS  
 CVS Projectset  
 Git

**Repositories**

Repository URL  ?

Credentials  Add

 **Advanced...**

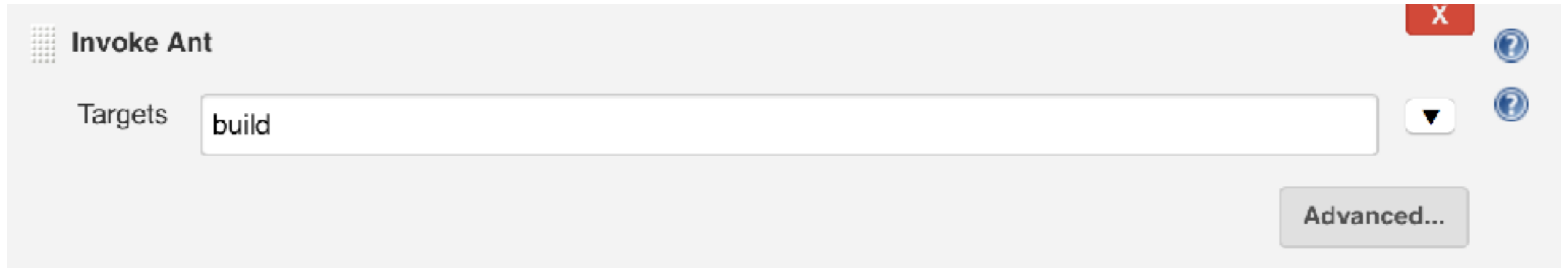
**Add Repository**

**Branches to build**

Branch Specifier (blank for 'any')  ?

**Add Branch**

# Ant



Ant support out of the box

# Ansible Plugin Config



The screenshot shows a configuration window titled "Invoke Ansible Playbook". It contains three main sections: "Ansible installation" with a dropdown menu set to "Default"; "Playbook path" with a text input field containing "deploy.projectx.yml"; and "Inventory" with a radio button selected for "File" and a sub-section "File path" with a text input field containing "inv/integration/hosts".

setup ansible playbook and inventory

# Ansible Extra Variables

Extra Variables

Key	version
Value	\$(version)
Hidden variable in build log	<input type="checkbox"/>


Key	app
Value	\$(WORKSPACE)
Hidden variable in build log	<input type="checkbox"/>

Add Extra Variable









setup version and application directory



# Job Execution

 **Jenkins**  Sebastian Feldmann | log out

Jenkins > ProjectX.integration

-  [Back to Dashboard](#)
-  [Status](#)
-  [Changes](#)
-  [Workspace](#)
-  [Build with Parameters](#)
-  [Delete Project](#)
-  [Configure](#)
-  [Move](#)

## Project ProjectX.integration

This build requires parameters:

version

Version you want to deploy

**Build**

# App Deployment

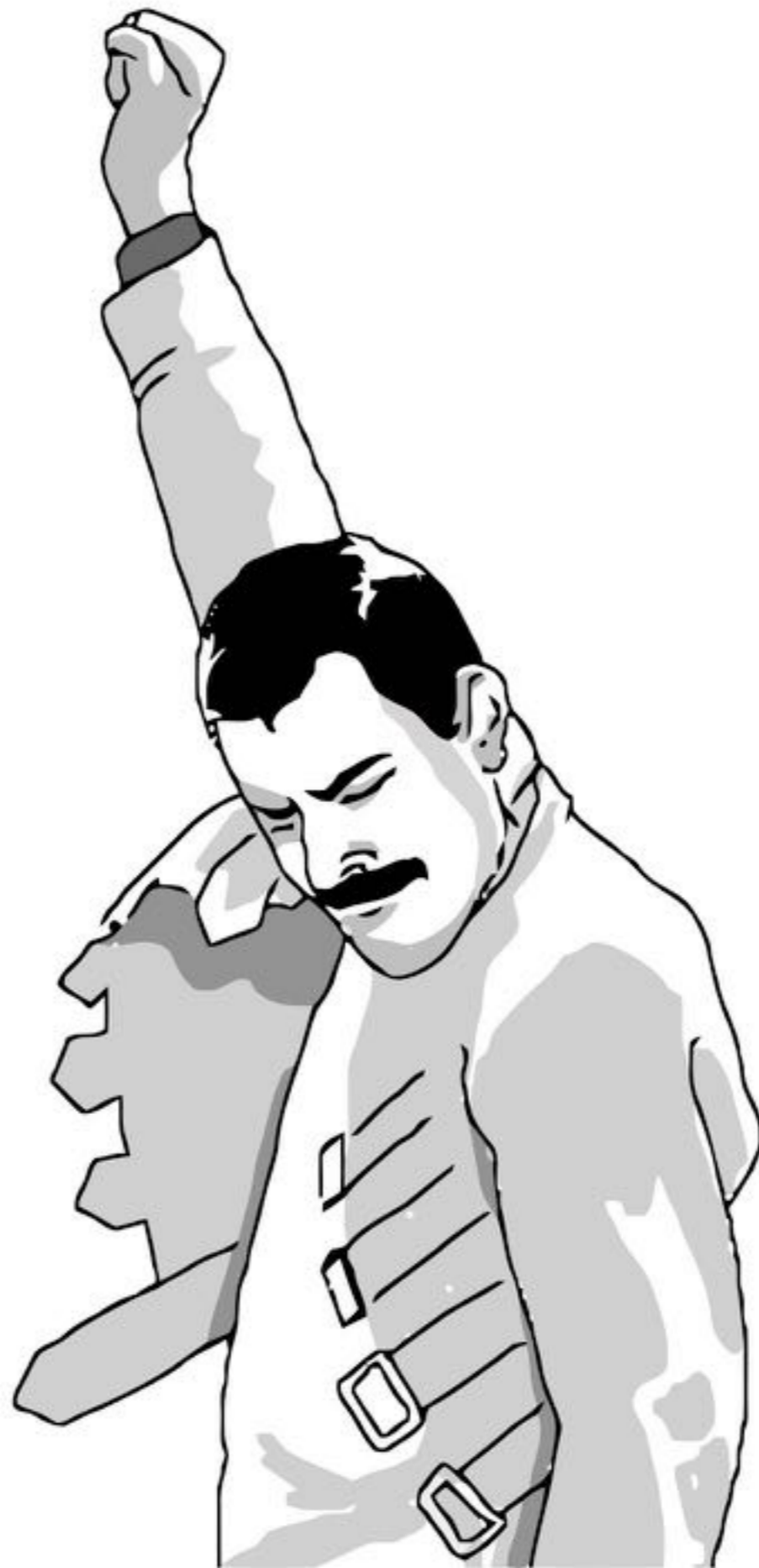
- Tag version in git repository
- Push button

# Recap

- System Management Build
- Software Deployment Build
- Execute everything with a single click

Is it getting boring yet?

At least a little right ;)





# Sebastian Feldmann

phpbu

<https://phpbu.de>



User Group  
Munich



**CHECK24**



@movetodevnull



sebastianfeldmann