Automating All The Things

Sebastian Feldmann

CHECK24

Automation

"Automate until the job is boring as hell"

Automation

- System Configuration
- Software Deployment
- Software Preparation
- Putting it all together

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, Bazzar, Subversion)

Ansible

- Inventory
- Playbook
- Role
- Tasks

Ansible Examples

- System Management
- Software deployment

System Management

Inventory

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

best practice inventory setup

Inventory

```
[webserver]
192.168.1.111

[dbserver]
192.168.1.222
```

ops/inv/integration/hosts

Playbook

```
- hosts: webservers
  sudo: true
  roles:
    - apache
```

ops/install.webserver.yml

Role

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

best 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
  tags: apache
```

ops/roles/apache/tasks/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

Execute Ansible

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

execution example

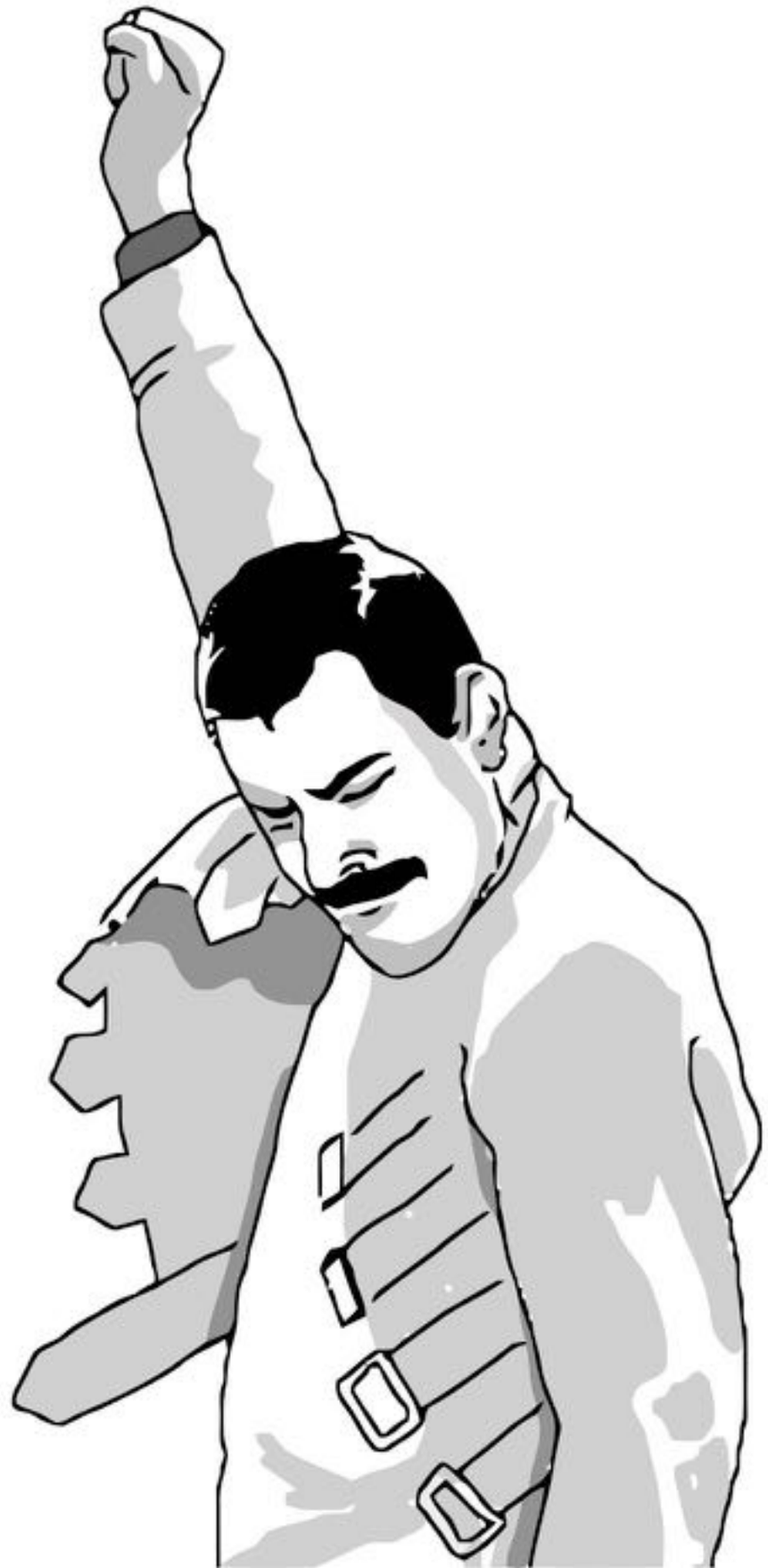
Execute Ansible

```
$ 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 Deployment

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

```yaml
system:
  environment: integration

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

ops/inv/integration/group_vars/webserver

Playbook

```
- hosts: webservers
  sudo: true
  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 webservers
- 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

```
# 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/my-project/{{ 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

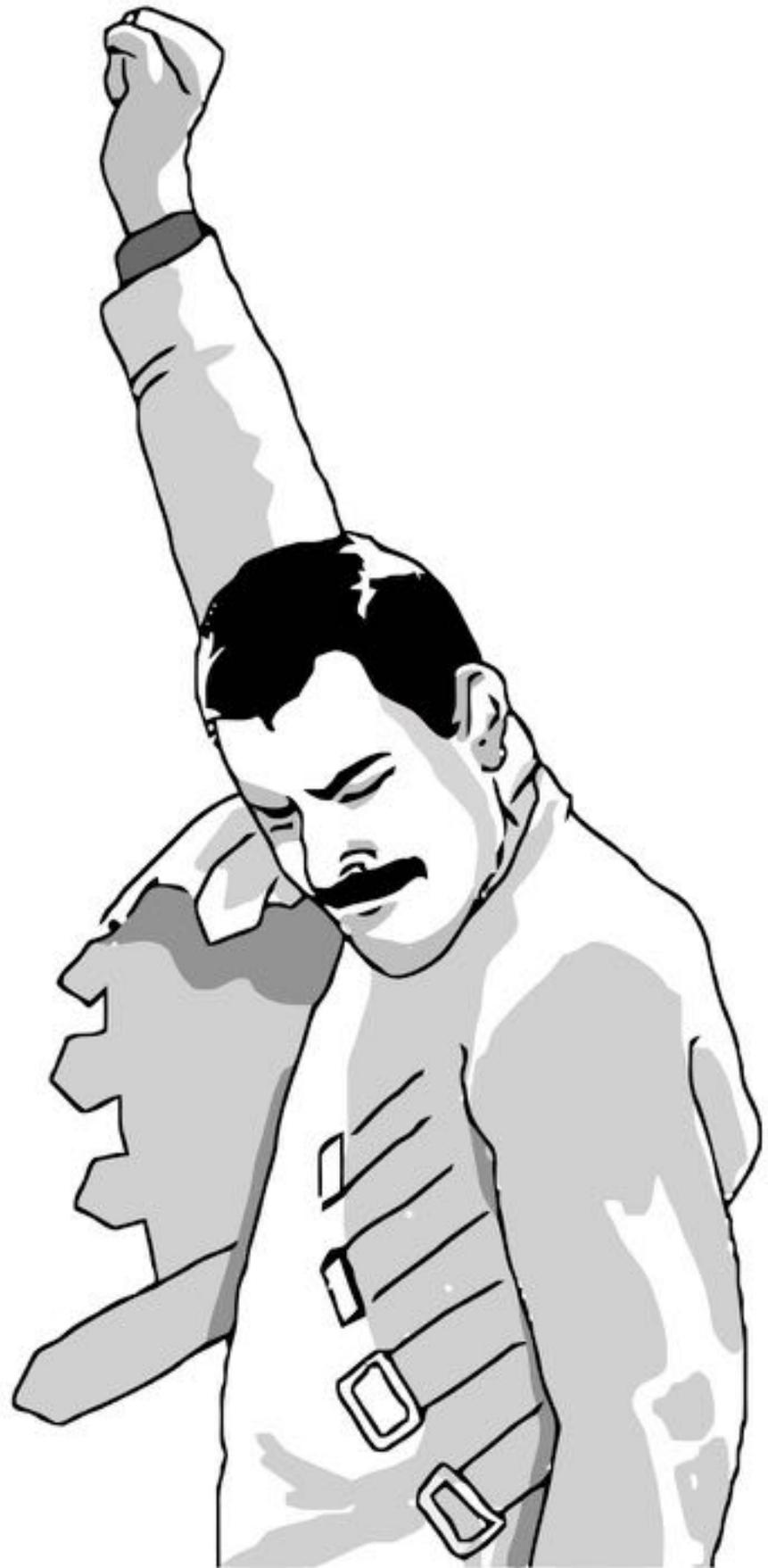
Recap

- 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



Software Preparation

"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

Ant Configuration

```xml
<?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

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…)
- …

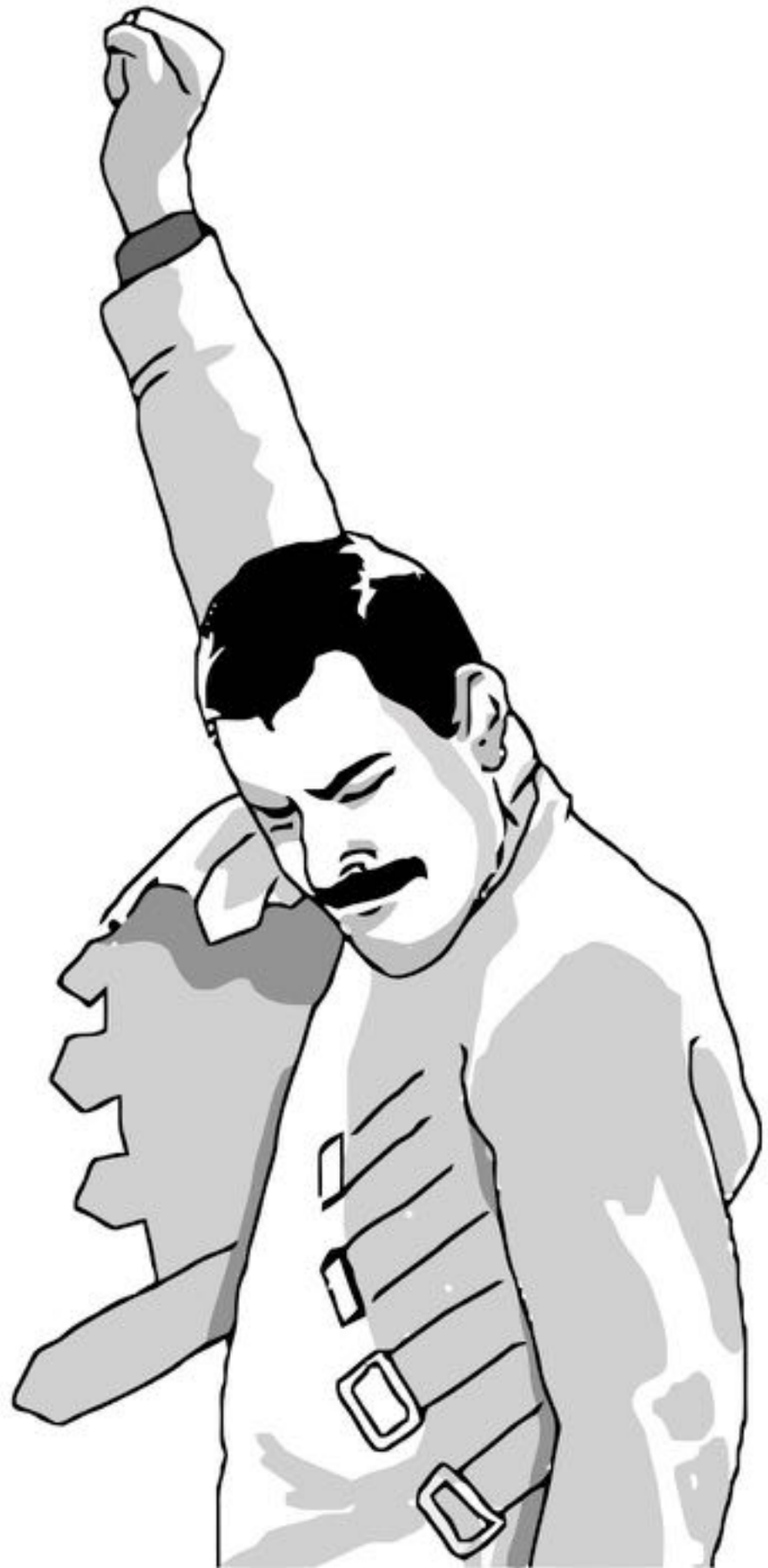
Ant Execution

```
$ ant
```

execute default target

```
$ ant phpunit
```

execute specific target



And finally…

Putting It All Together

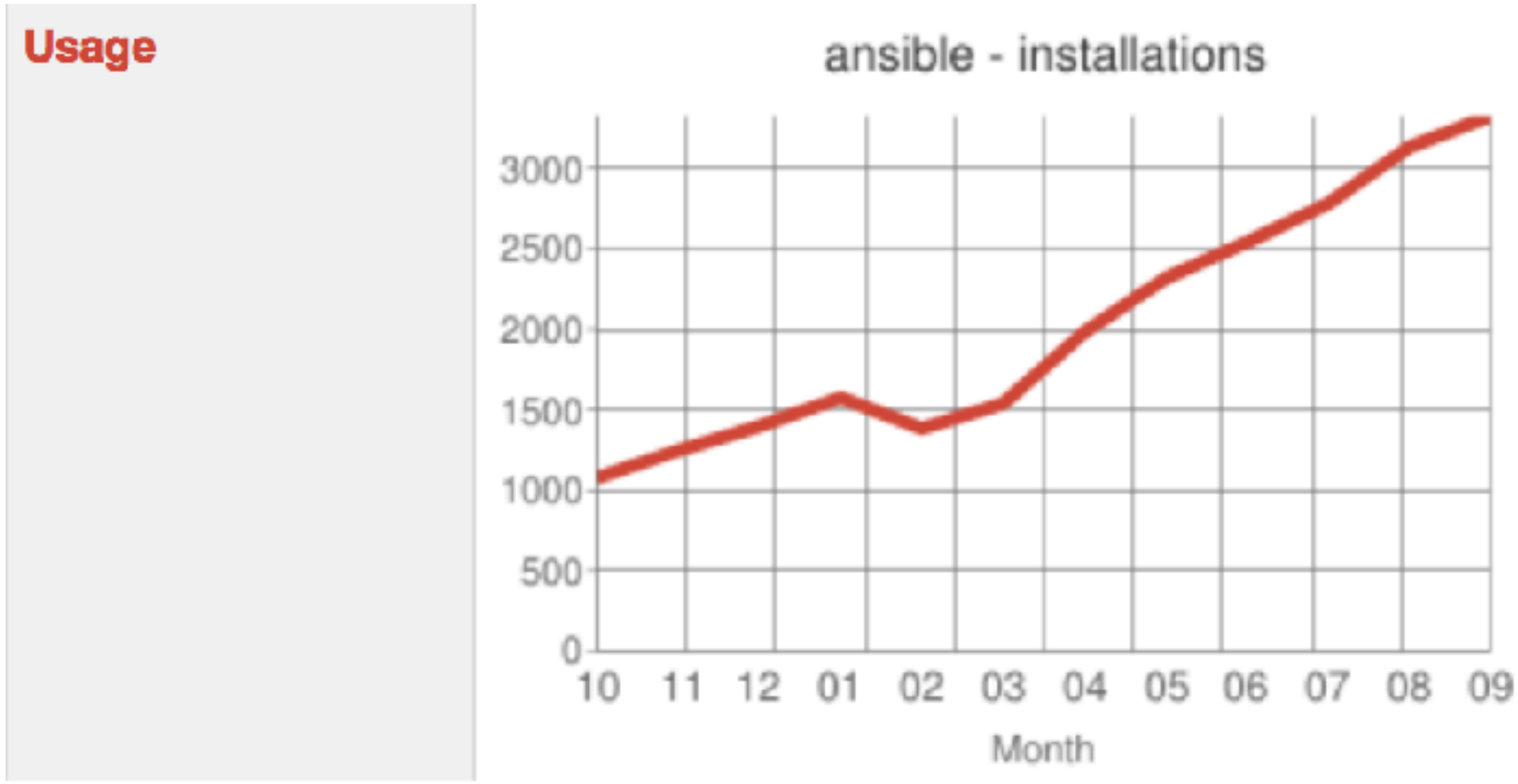
Jenkins

“Build great things at any scale“

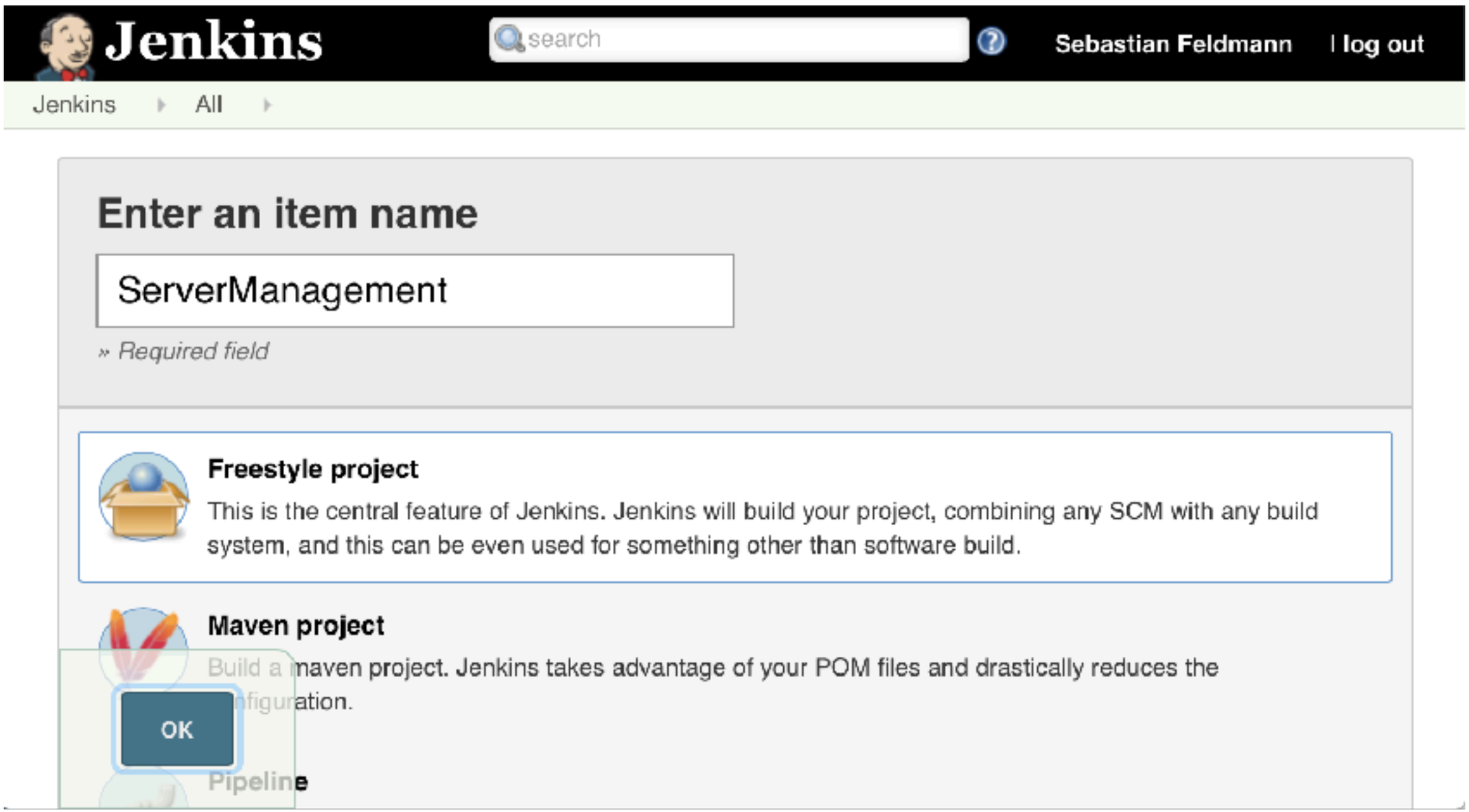
Jenkins

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

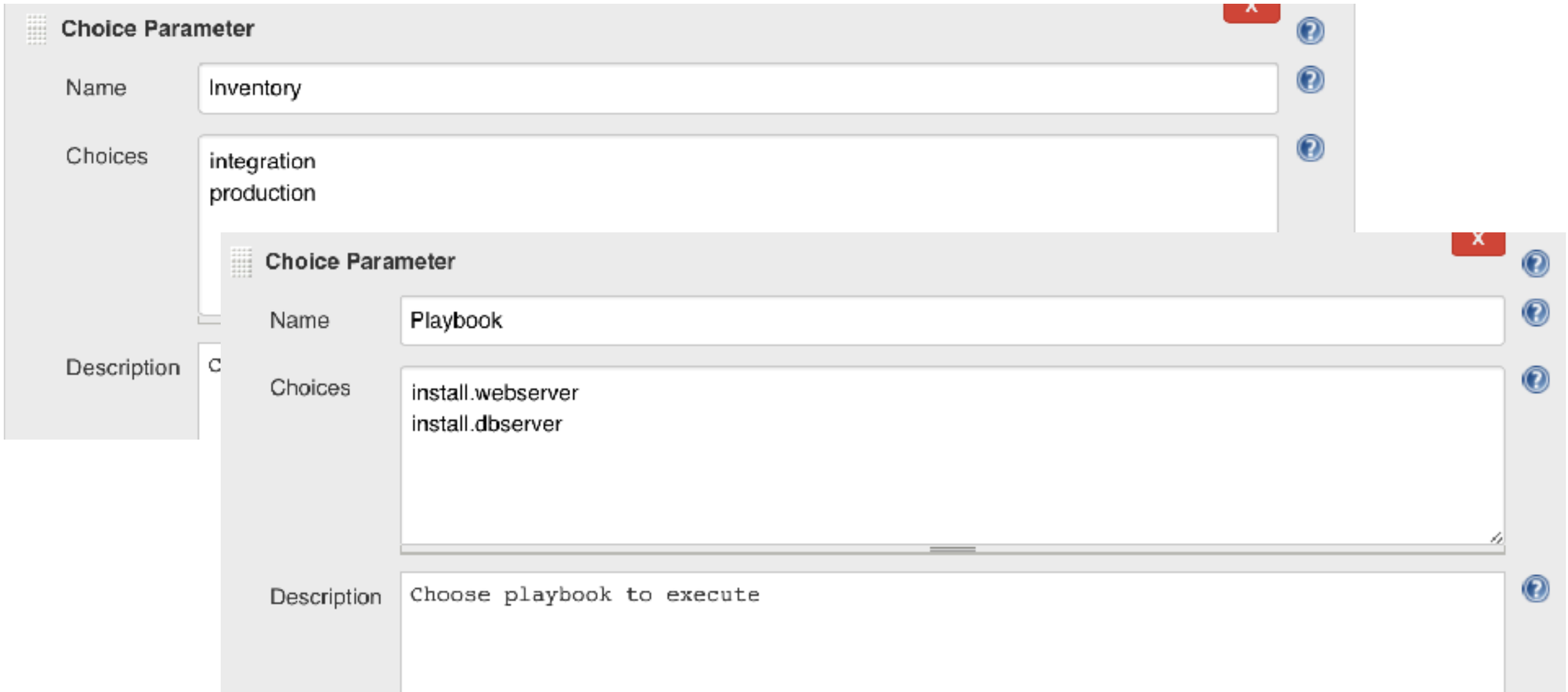
Ansible Plugin



System Management



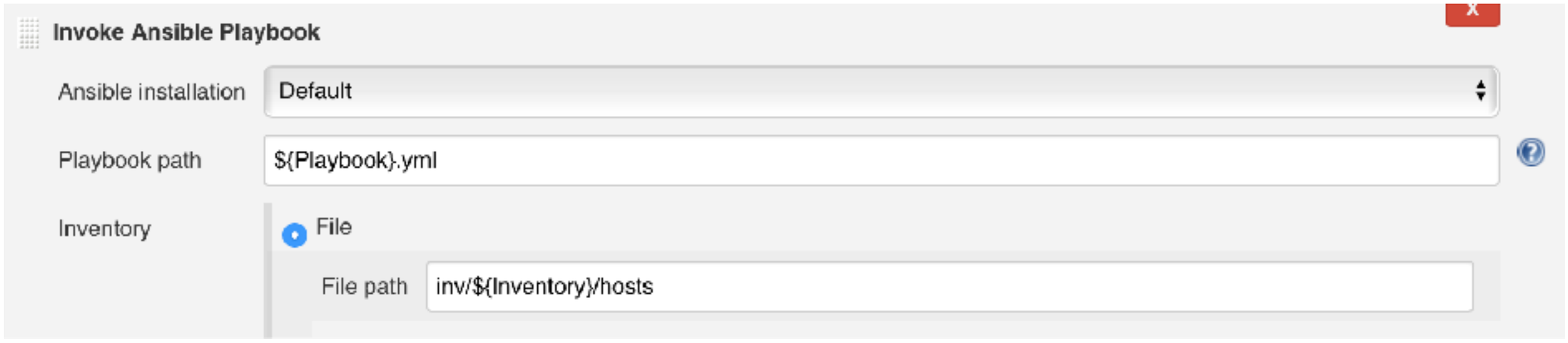
Parameter Setup



SCM Config

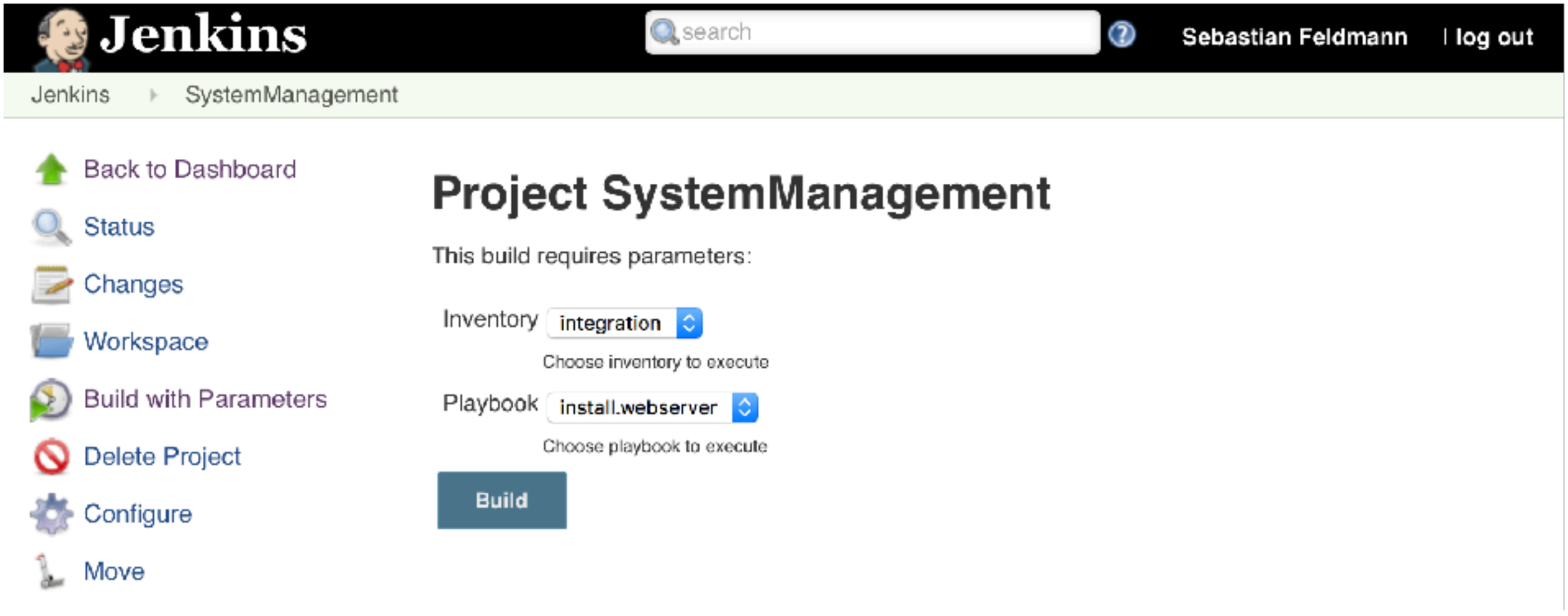
use some SCM to manage your Ansible project

Plugin Config

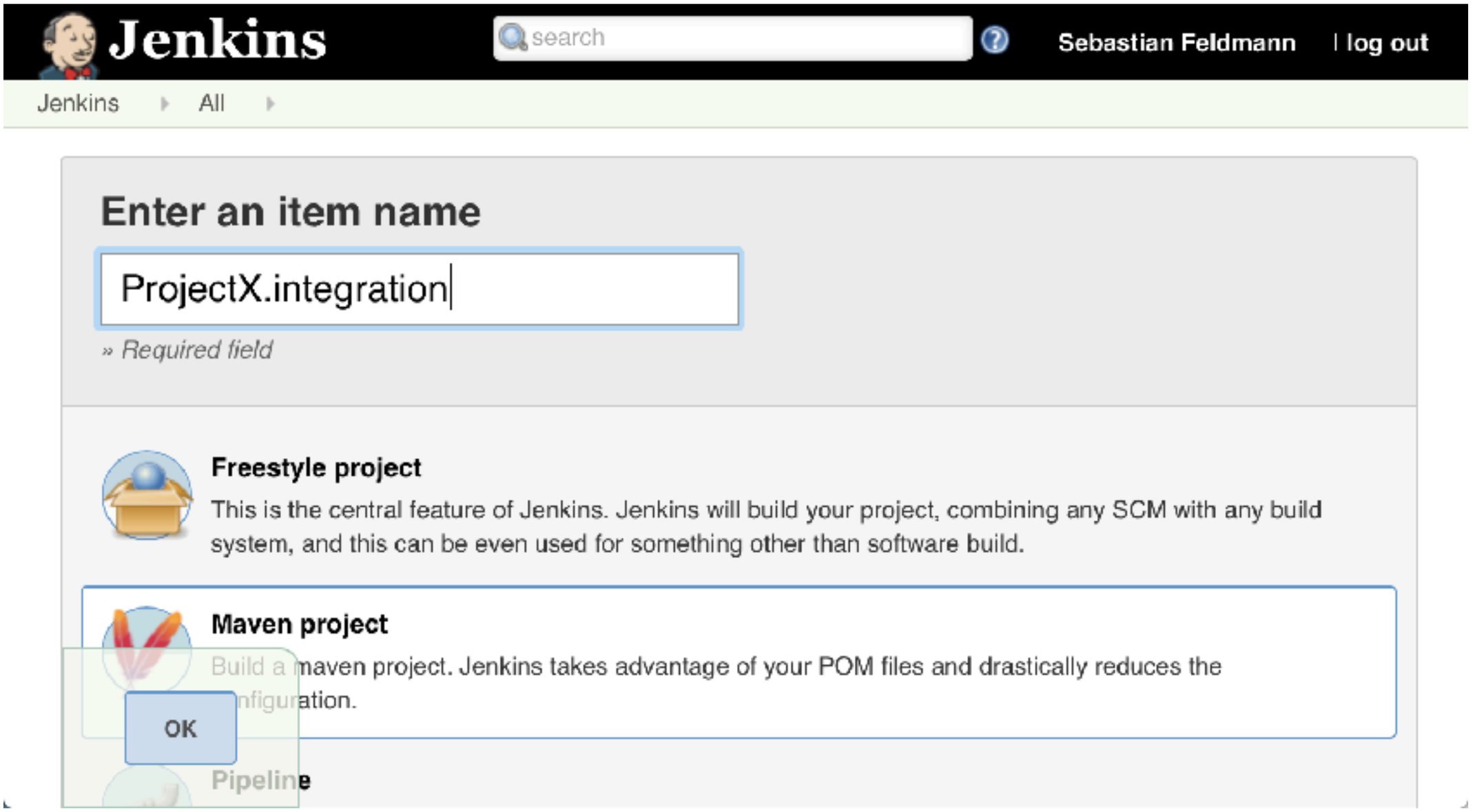


parameter usage

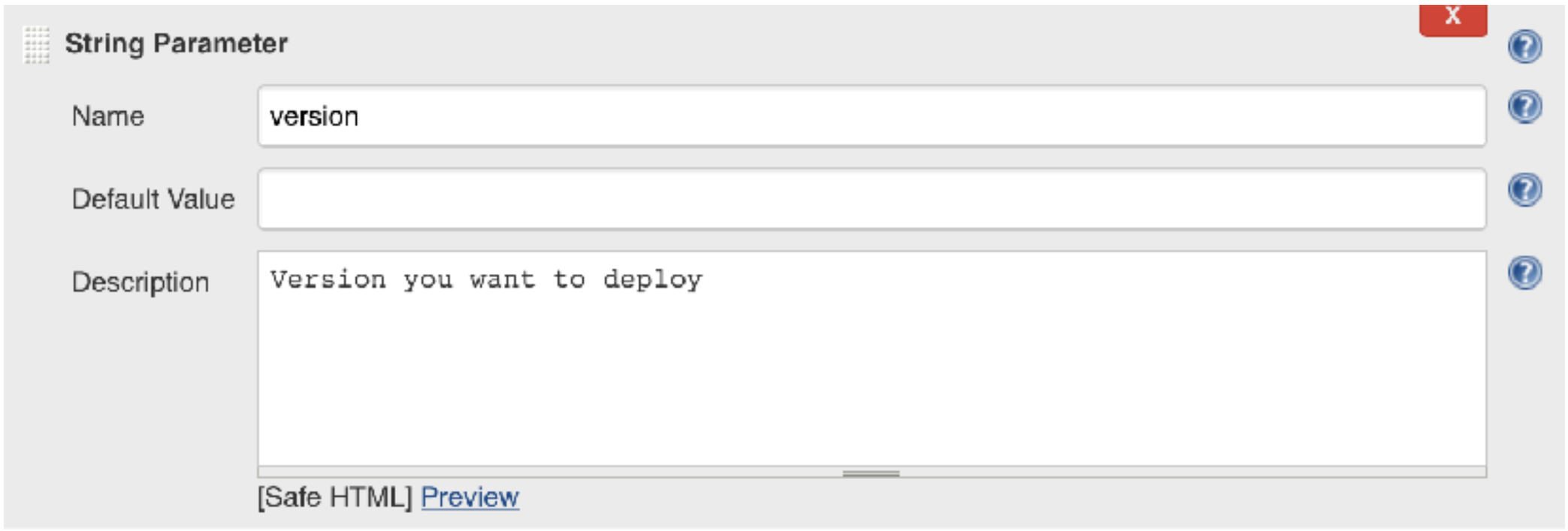
Job Execution



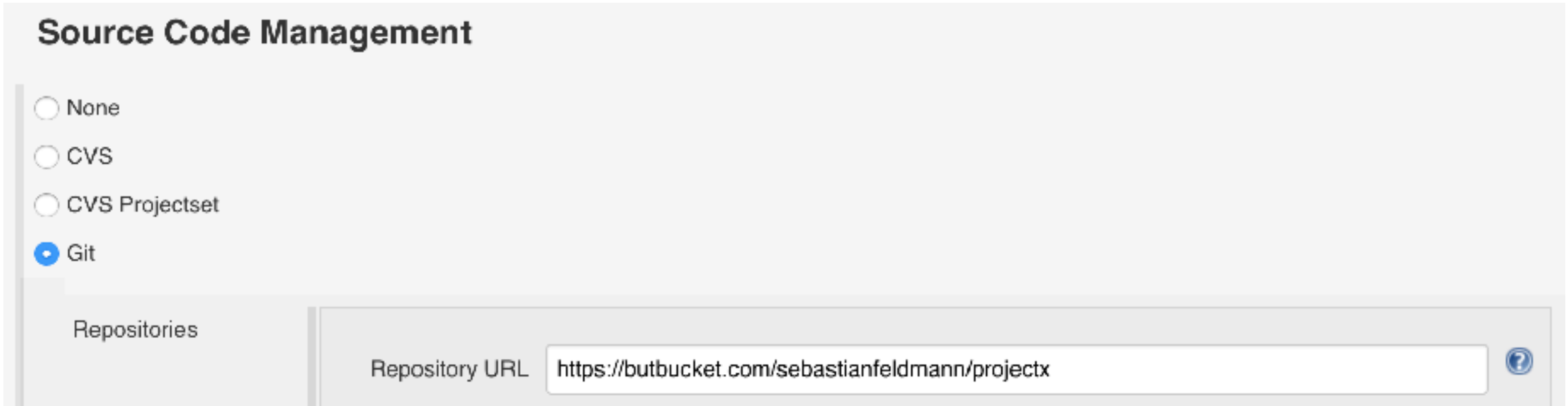
Software Deployment



Parameter Setup

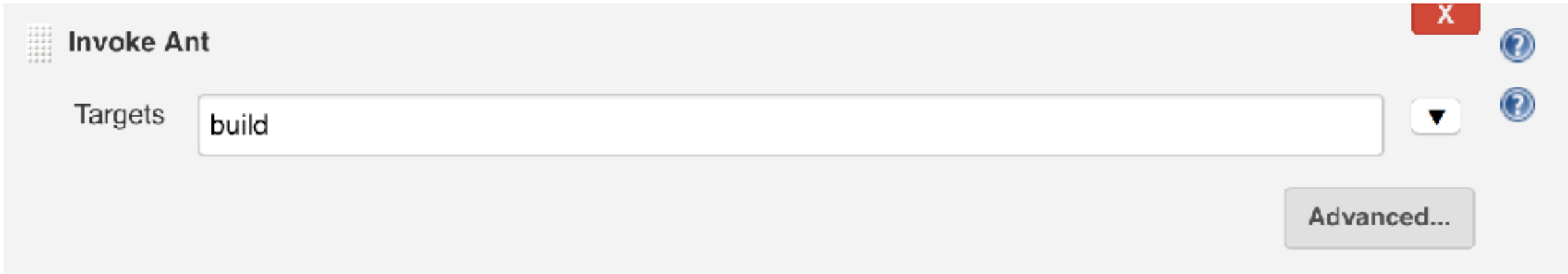


SCM Config



configure projectx SCM

Ant

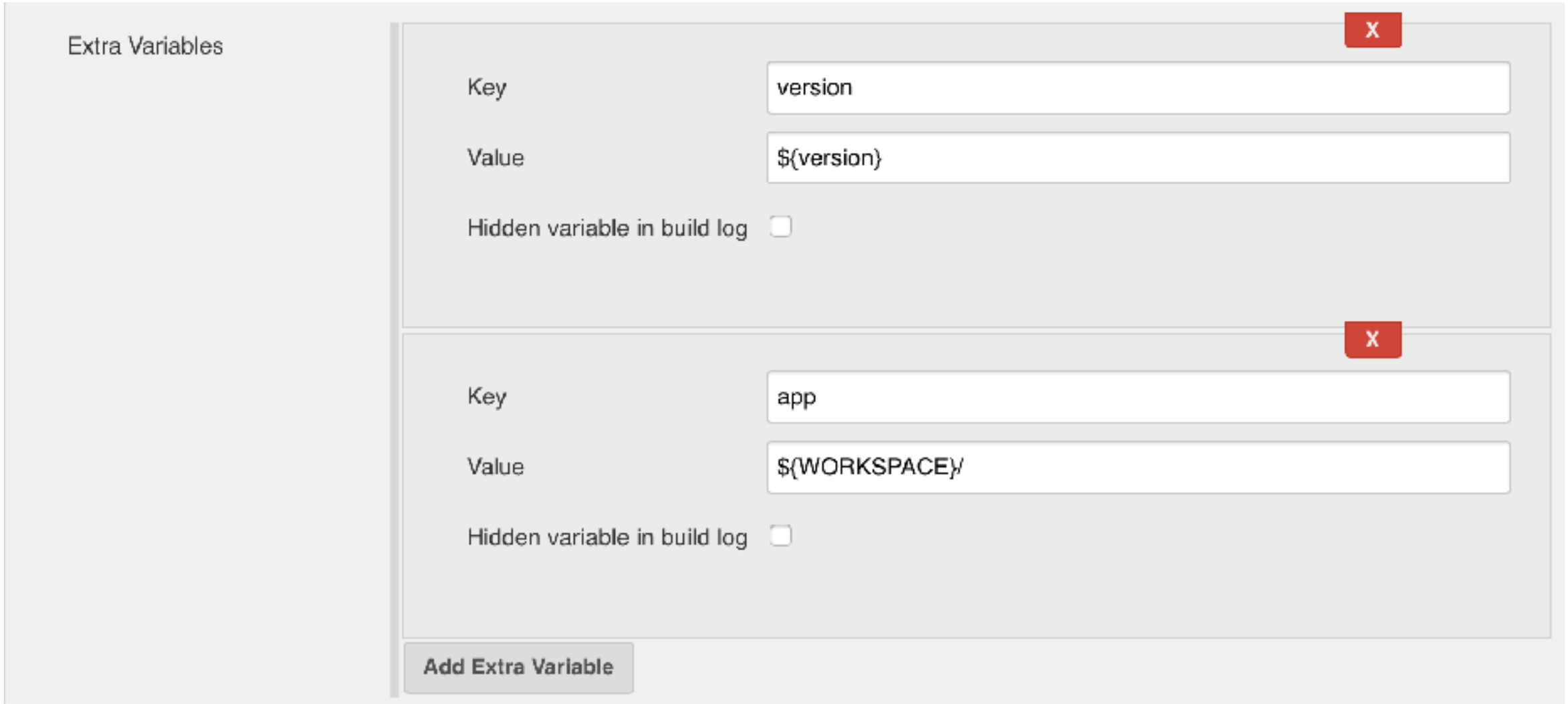


Ant support out of the box

Plugin Config

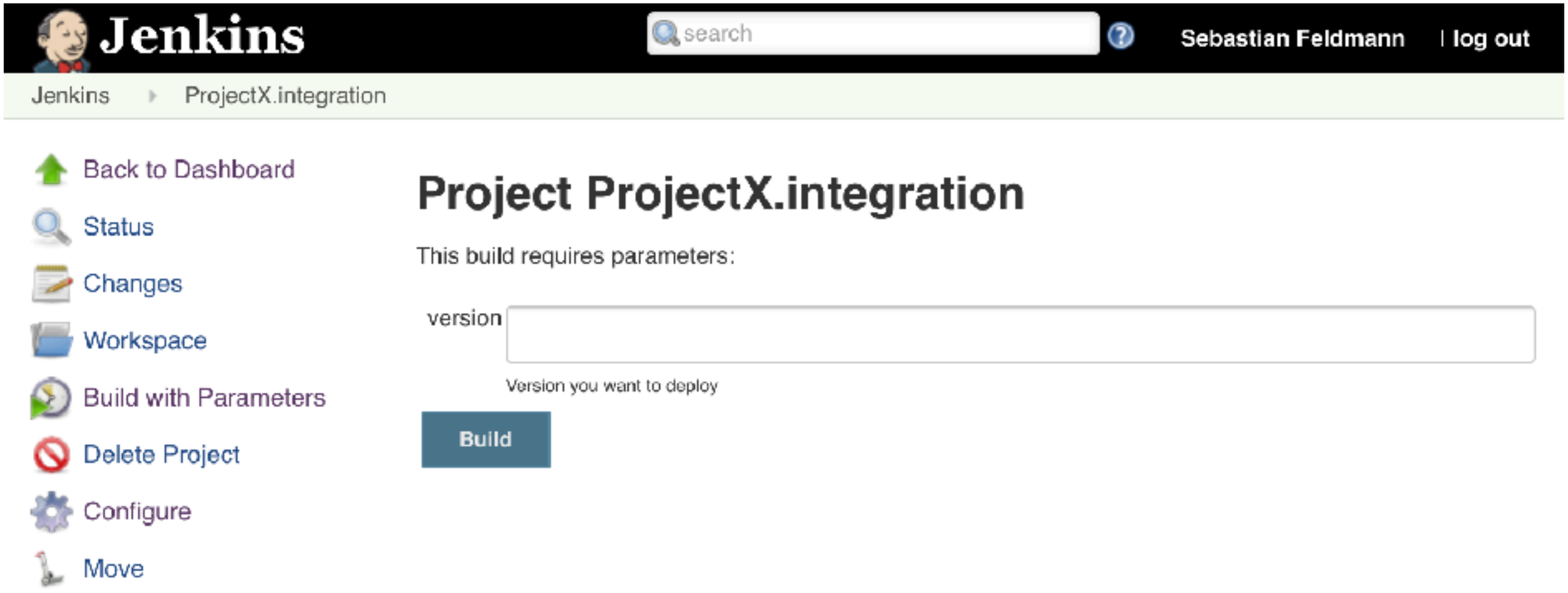
setup ansible playbook and inventory

Ansible Extra Variables



setup version and application directory

Job Execution



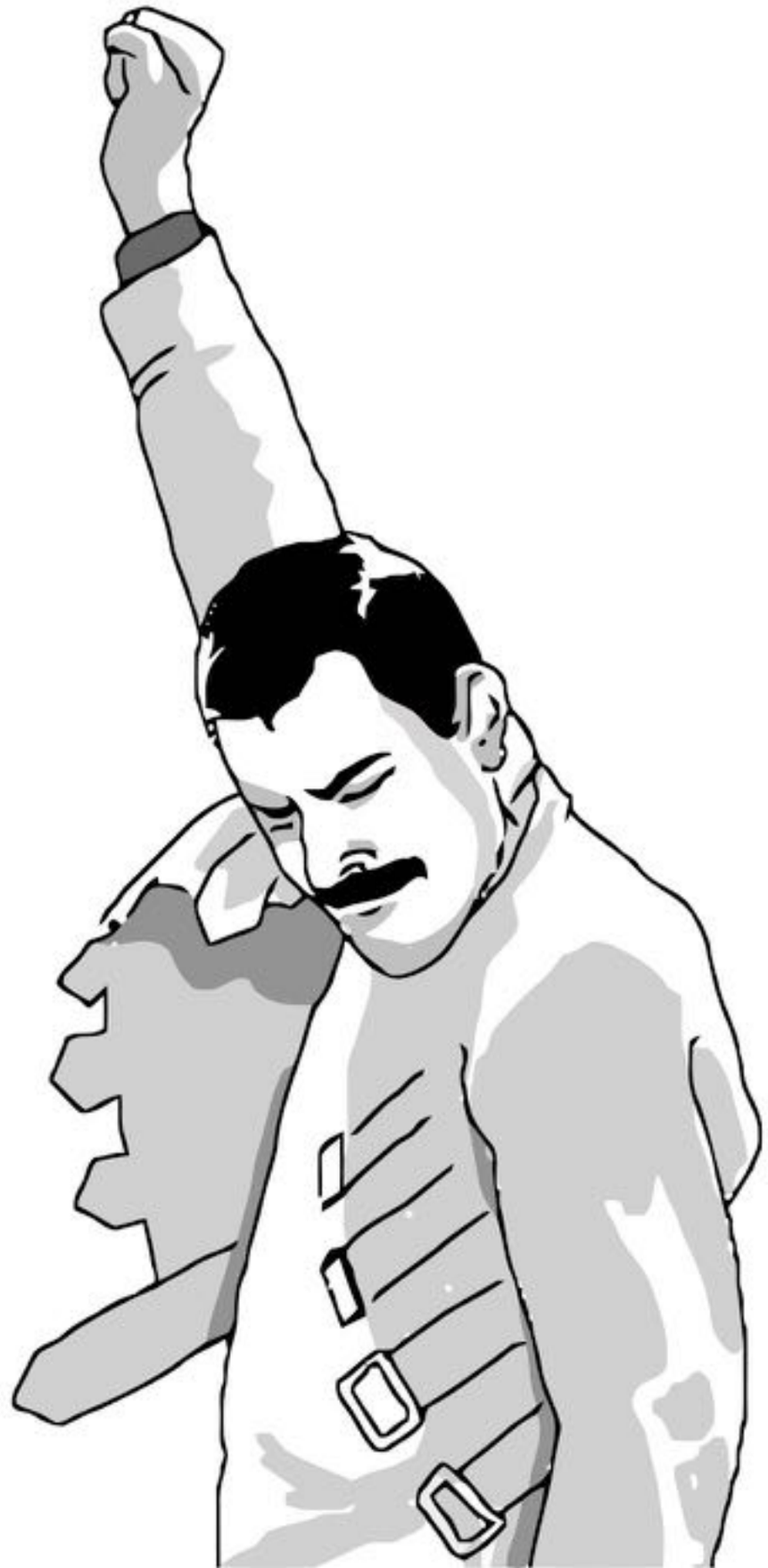
Jenkins Log

```
Buildfile: /var/lib/jenkins/jobs/projectx.integration/workspace/build.xml
...
composer-install:
     [exec] Loading composer repositories with package information
     [exec] Nothing to install or update
     [exec] Generating autoload files
     ...
[workspace] $ ansible-playbook deploy.projectx.yml -i inv/integration/hosts

     PLAYBOOK: deploy.projectx.yml **************************************

     PLAY [webserver] ***************************************************

     TASK [install vhost] ***********************************************
     ok: [webserver]
     ...

BUILD SUCCESSFUL
```

Recap

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

Is it getting boring yet?



thank you

Sebastian Feldmann

phpbu

https://phpbu.de

php User Group Munich

CHECK24

@movetodevnull

sebastianfeldmann

Q & A

09:00 - 10:00

FORUM 7

Mastering git – Insights & Tips & Tricks

Sebastian Feldmann, *CHECK24*