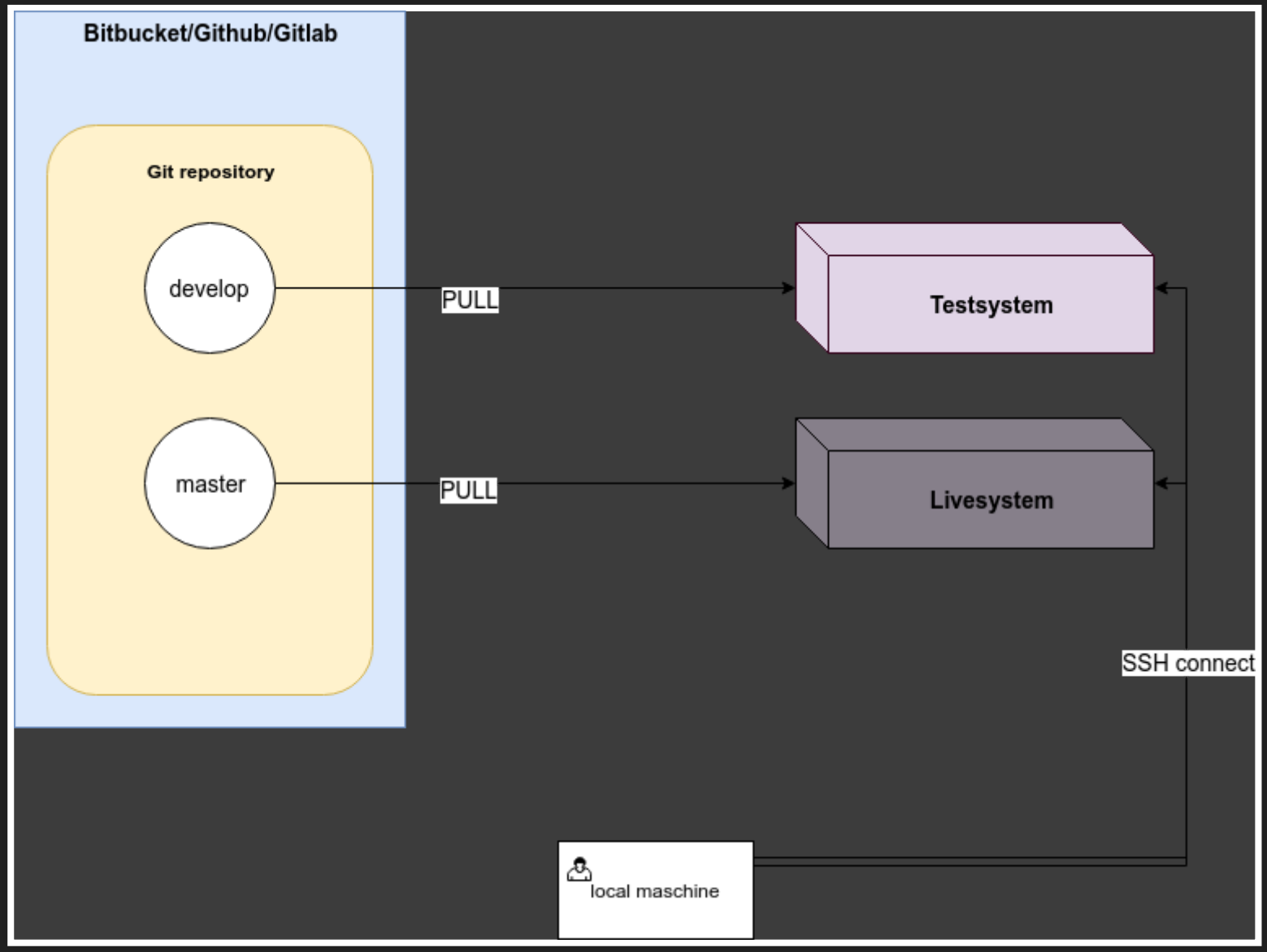
DEPLOYMENT VON WEB APPLICATIONS MIT ANSIBLE

Holger Lösken // Sächsische Dampfschiffahrt

Wohin geht die Reise?



WAS IST ANSIBLE?

Ansible:

- Open Source
- Automatisierung, Orchestrierung von Konfiguration & Administration
- Python (Linux) & Powershell (Windows)
- RedHat, ursprgl. AnsibleWorks Inc.

Ziele:

- sicher: agent-less, nur OpenSSH und Python
- zuverlässig: idempotent, keine Nebenwirkungen*
- leicht erlernbar: YAML und Jinja-Templates
- minimal: keine Abhängigkeit von der Umgebung

Ansible Tower

- Früher *AWX* genannt
- Stellt eine GUI/Dashboard zur Verfügung
- Features
  - Auswertungen
  - Live-Monitoring
  - Scheduling
  - Rollen-basierter Zugriff
  - etc.
- Kostenpflichtig

... für unser Deployment nicht benötigt.

Getting started

1. Installation Ansible eigener Rechner
   - Linux: Paketmanager
   - Mac: Python Pip
2. Installation *OpenSSH* und *Python* Zielsystem

... das war's!

UNSER ERSTES ANSIBLE PLAYBOOK

... und das wird gar nicht weh tun ;)

Playbooks

Es gibt *ad-hoc* tasks

`$ ansible <task>`

... und es gibt *Playbooks*

- `$ ansible-playbook <playbook>`
- Deklarieren von Konfigurationen
- Orchestrierung von Abläufen und Aufgaben

Playbook Beispiel

```
- hosts: webservers
  vars:
    http_port: 80
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Playbook (und Rollen)

Playbook

```
- hosts: "webservers"
  roles:
    - { role: deploy }
    - { role: notify }
```

Rolle

- Läd Tasks, die zu einem bestimmten Playbook gehören
- Gruppierung von wiederkehrenden Aufgaben

TEST VS. LIVE

INVENTORIES TO THE RESCUE

Inventories

- Gruppierung von gleichartigen Maschinen, z. B. web server, database server, etc.
- Unterscheidung nach *environment*, *testing*, *staging*, *production*

.. Bsp ->

Inventory Beispiel

```
[webservers]
test-www.saechsische-dampfschiffahrt.de

[webservers:vars]
env=testing
git_branch=develop

[all:vars]
ansible_connection=ssh
ansible_user=test-www.saechsische-dampfschiffahrt.de
ansible_host=edward.sdsgruppe.de

deploy_user={{ ansible_user }}
deploy_group=www-data
deploy_email_to=edv@sdsgruppe.de
```

RECAP #1

WAS BISHER GESCHAH.. ;-)

Recap 1:

- Ansible im Allgemeinen
- Playbooks
  - Können einzelne Aufgaben enthalten
  - Besser: Gruppieren von Aufgaben in Rollen, die über Playbooks ausgelöst werden
- Roles
  - Gruppieren wiederk. Aufgaben
- Inventories
  - Gruppierungen von *environments*, z. B. `testing`, `production`

ALLE (3) ZUSAMMEN

DEPLOYMENT AM BEISPIEL VON WORDPRESS

Deployment Wordpress

Inventories

```
inventories
  |-- production
      |-- hosts
  |-- testings
      |-- hosts
```

Playbook

```
- hosts: " webservers"
  roles:
    - { role: deploy, tags: ['deploy'] }
```

Deployment Wordpress

Roles

```
roles
  |-- deploy
      |-- tasks
          |-- git.yml
          |-- init.yml
          |-- main.yml
          |-- reboot.yml
  |-- notify
      |-- tasks
          |-- example.yml
```

Deployment Wordpress

main.yml

- Startpunkt jeder Rolle
- Kann Tasks enthalten, oder aufgesplittet s. u.

```
- include_tasks: init.yml // Initialisierung
- include_tasks: git.yml // Git pull
- include_tasks: links.yml // Setzen von Symlinks, Berechtigun
- include_tasks: reboot.yml // Refresh Cache, current Symlink
- include_tasks: post_tasks.yml // E-Mail senden, Slack notify
```

Deployment Wordpress

init.yml

```
- name: Generate release timestamp
  command: date +%Y%m%d%H%M%S
  register: timestamp
  run_once: true

- set_fact: "release_path='{{ releases_path }}/{{ timestamp.st

# This sets the release count to 5, needs 5 + 1 = 6 in total ;
- set_fact: keep_releases={{ 6|int }}
```

Deployment Wordpress

init.yml

```
- name: Generate release timestamp
  command: date +%Y%m%d%H%M%S
  register: timestamp
  run_once: true

- set_fact: "release_path='{{ releases_path }}/{{ timestamp.st

# This sets the release count to 5, needs 5 + 1 = 6 in total ;
- set_fact: keep_releases={{ 6|int }}
```

Deployment Wordpress

links.yml

```
- name: Remove shared folder
  file: "path={{ release_path }}/src/wp-content/{{ item }} sta
  with_items:
    - "uploads"
    - "cache"

- name: Symlink shared content
  file: "src={{ shared_path }}/src/{{ item }} dest={{ release_
  with_items:
    - "wp-config.php"
    - "wp-content/uploads"
    - "wp-content/cache"
```

Deployment Wordpress

reboot.yml

```
- name: Make main folder writeable for webserver
  file: "path={{ release_path }}/{{ item }} owner={{deploy_use
  with_items:
    - "src"
    - "src/wp-content"
    - "src/wp-content/plugins"

- name: Set permission for wflogs/
  file: "path={{ release_path }}/src/wp-content/wflogs owner={

- name: Update current symlink
  file: "state=link path={{ current_path }} src={{ release_pat

- name: Refresh cache
  command: "wp cache flush"
```

Deployment Wordpress

post_tasks.yml

```
- name: Clean old releases
  shell: "ls -tp | grep '/$' | tail -n +{{ keep_releases|strin
  args:
      chdir: "{{ releases_path }}"

- name: Inform Bugsnag
  uri:
    url: https://notify.bugsnag.com/deploy
    method: POST
    body: "apiKey={{ bugsnag_api_key }}&releaseStage={{ env }}
```

Deployment Wordpress

Deployment Testsystem

```
$ ansible-playbook -i inventories/testing deploy.yml
```

Deployment Livesystem

```
$ ansible-playbook -i inventories/production deploy.yml
```

... das war's (schonwieder) ¯\(ツ)/¯

RECAP #2

- Inventories, Roles, Playbook im Detail
- Verschiedene Inventories für versch. Environments
- Unterteilen von Rollen für versch. Bereiche
- => Ein Kommando zum Deployen

HANDY STUFF

DATABASE DUMP, ...

DATABASE DUMP

Playbook:

```
- hosts: "webservers"
  roles:
    - { role: mysqldump }
```

Database dump

Role (alles in main.yml)

```
- set_fact:
    db_filename: "dump-{{ ansible_env.DB_DATABASE }}-{{ ansibl

- name: Create MySql dump
  mysql_db:
    state: dump
    name: "{{ ansible_env.DB_DATABASE }}"
    login_user: "{{ ansible_env.DB_USERNAME }}"
    login_password: "{{ ansible_env.DB_PASSWORD }}"
    target: "shared/{{ db_filename }}"

- name: Make a gzipped archive
  archive:
    path: "{{ shared_path }}/{{ db_filename }}"
    dest: "{{ shared_path }}/{{ db_filename }}.gz"
```

Database dump

Database dump ziehen - Testsystem

```
$ ansible-playbook -i inventories/testing mysqldump.yml
```

Extras/Vergessene Sachen ;-)

- Globale Variable über Gruppenvariable, z. B. Git-Repository
- Rechte setzen über _authorized_keys_ file
- Rollback
  - Entweder: eine Release-Version zurück
  - Oder: Übergabe eines Git-Hashes/-Tag und nochmal komplettes Deployment
  - Andere Ideen?

... wird im Repository enthalten sein

VIELEN DANK!

Slides & Code-Beispiele auf Github

Fragen?!

MAIL: post@codedge.de

TWITTER: cod2edge