Automated deployment with Ansible

Juergen Brendel

juergen@brendel.com
@BrendelConsult

Summary

- Configuration management background
- Ansible intro
- Learn by example
- Unified test and deployment environments

juergen@brendel.com
@BrendelConsult

Configuration Management: Why and how

Configuration Management?

I thought we'd talk about deployment?

juergen@brendel.com
@BrendelConsult

Configuring servers

How do you configure a server?

juergen@brendel.com
@BrendelConsult

Configuring servers

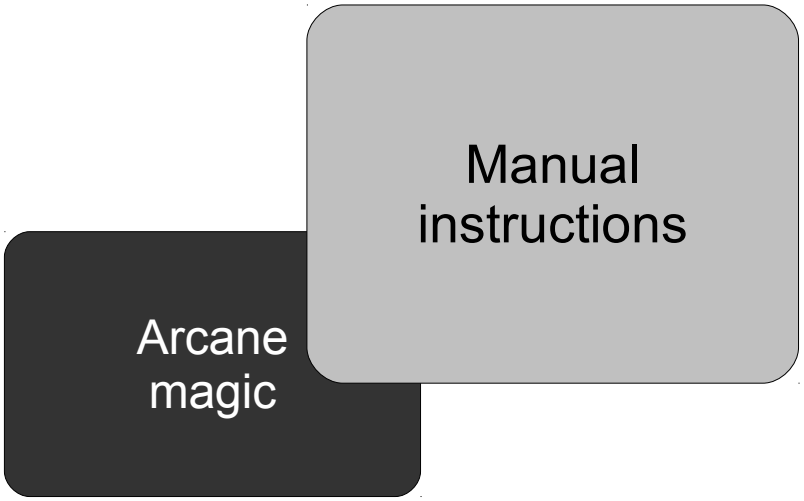
How do you configure a server?

Arcane
magic

juergen@brendel.com
@BrendelConsult

Configuring servers

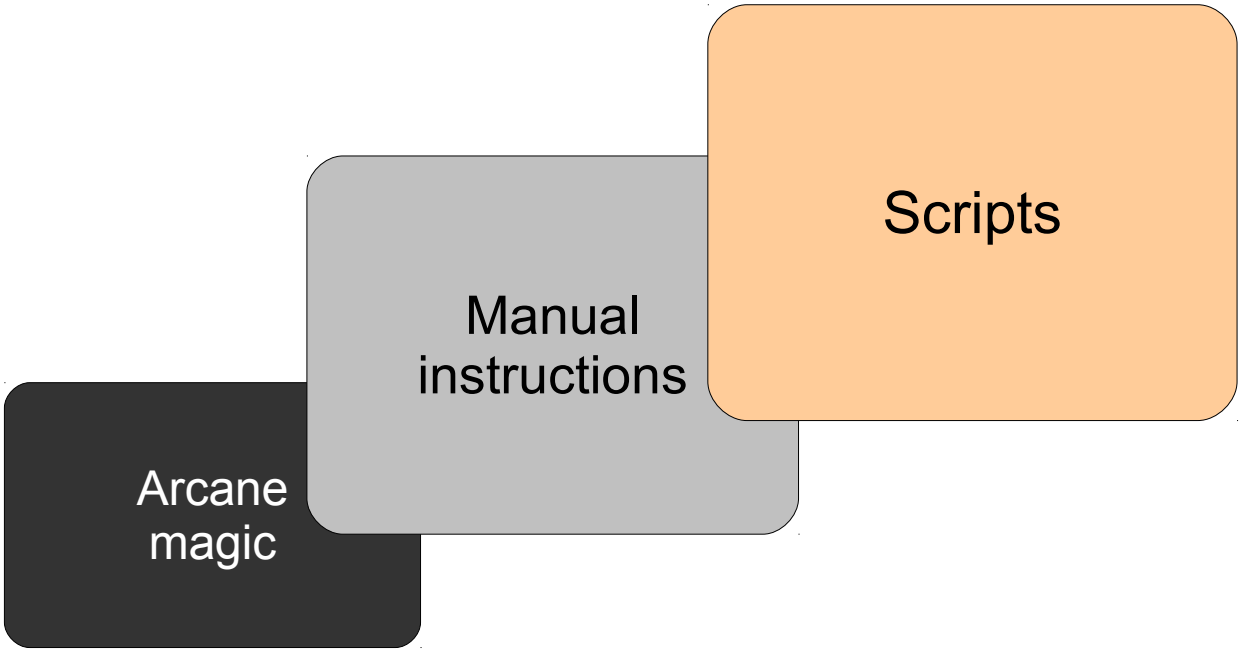
How do you configure a server?



juergen@brendel.com
@BrendelConsult

Configuring servers

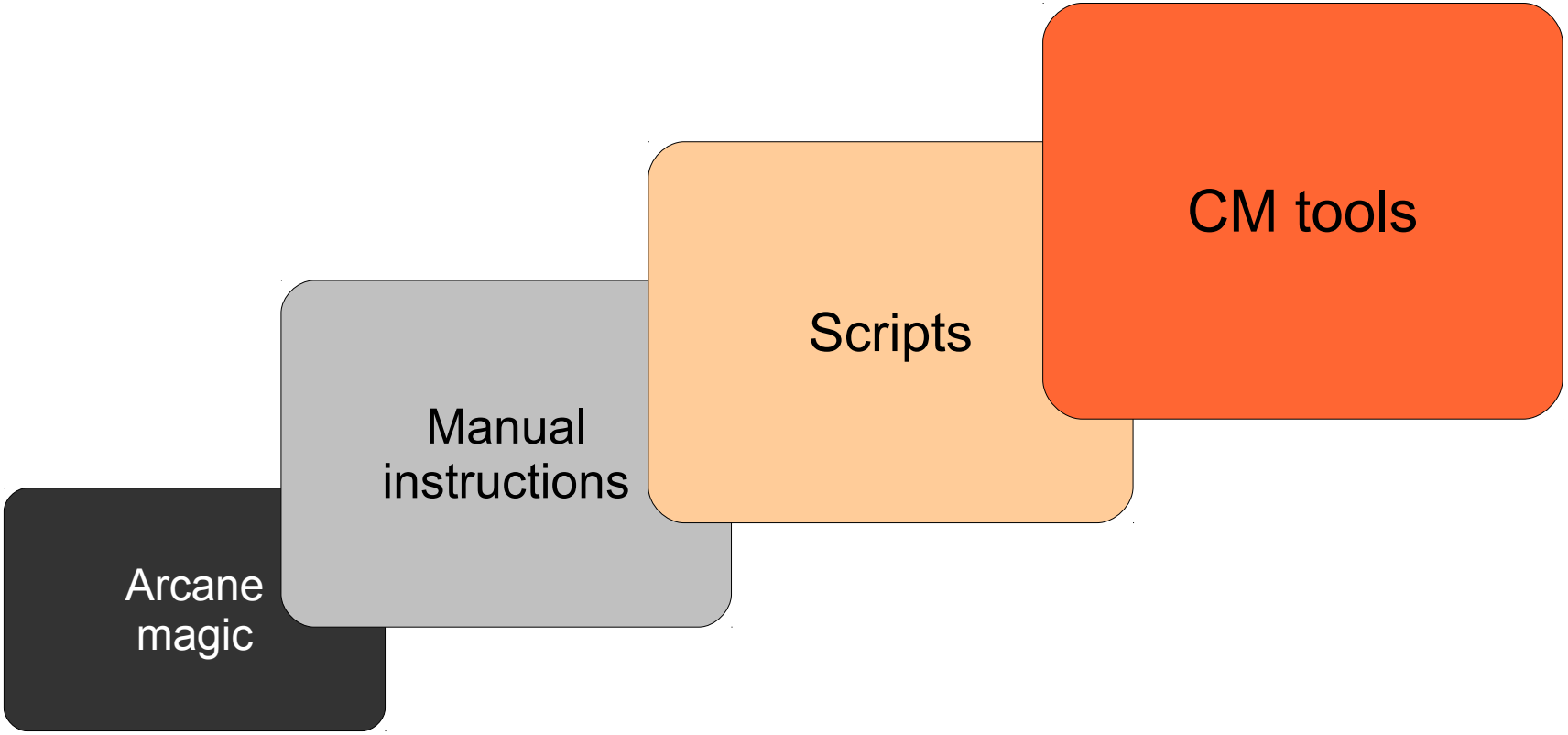
How do you configure a server?



juergen@brendel.com
@BrendelConsult

Configuring servers

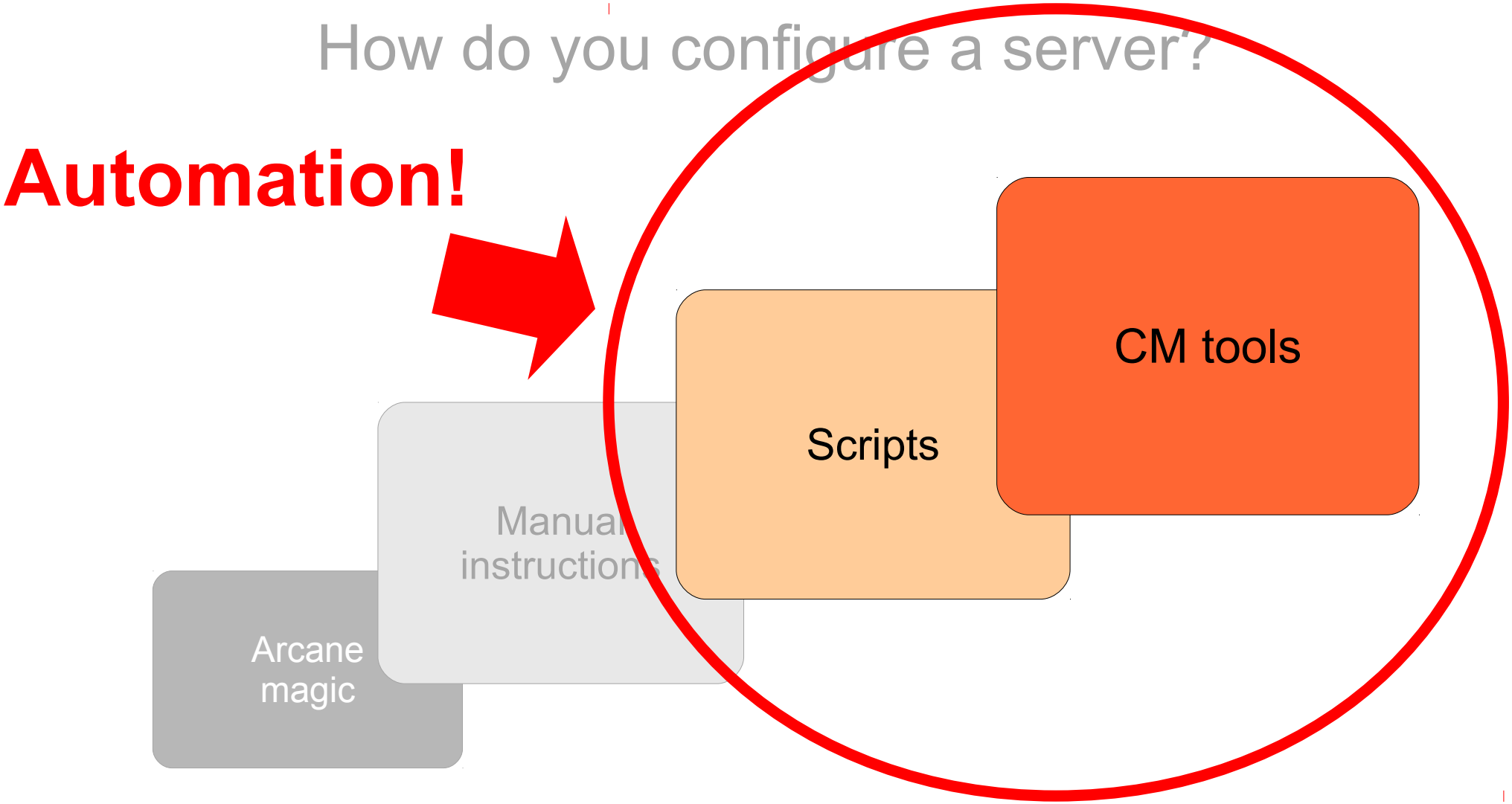
How do you configure a server?



juergen@brendel.com
@BrendelConsult

Configuring servers

How do you configure a server?



juergen@brendel.com
@BrendelConsult

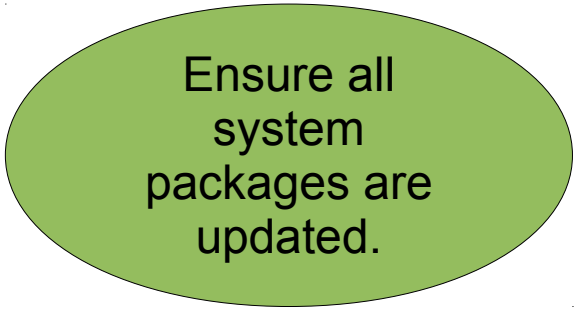
CM Tools

Describe the desired state

juergen@brendel.com
@BrendelConsult

CM Tools

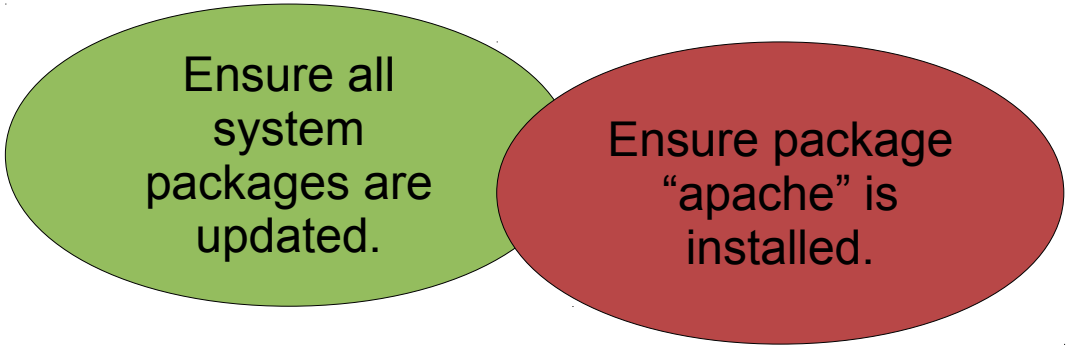
Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools

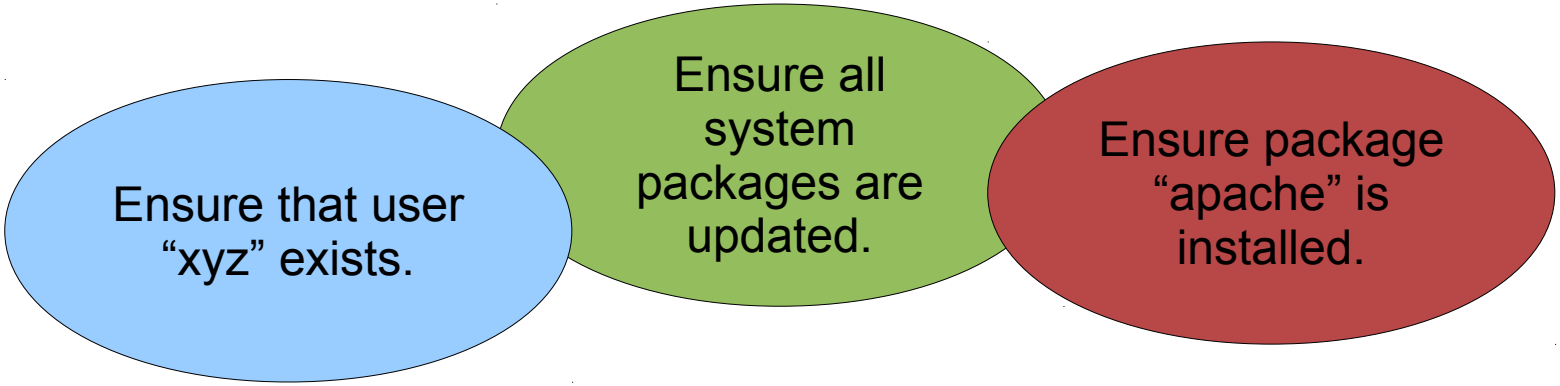
Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools

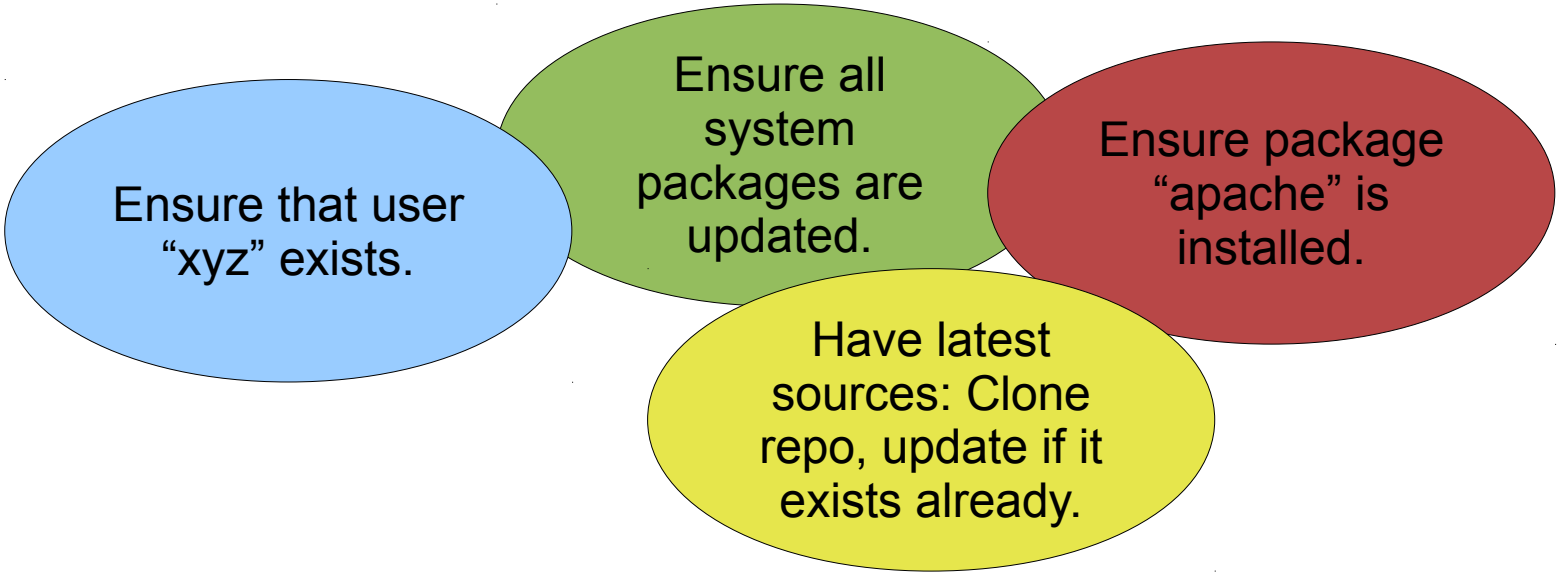
Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools

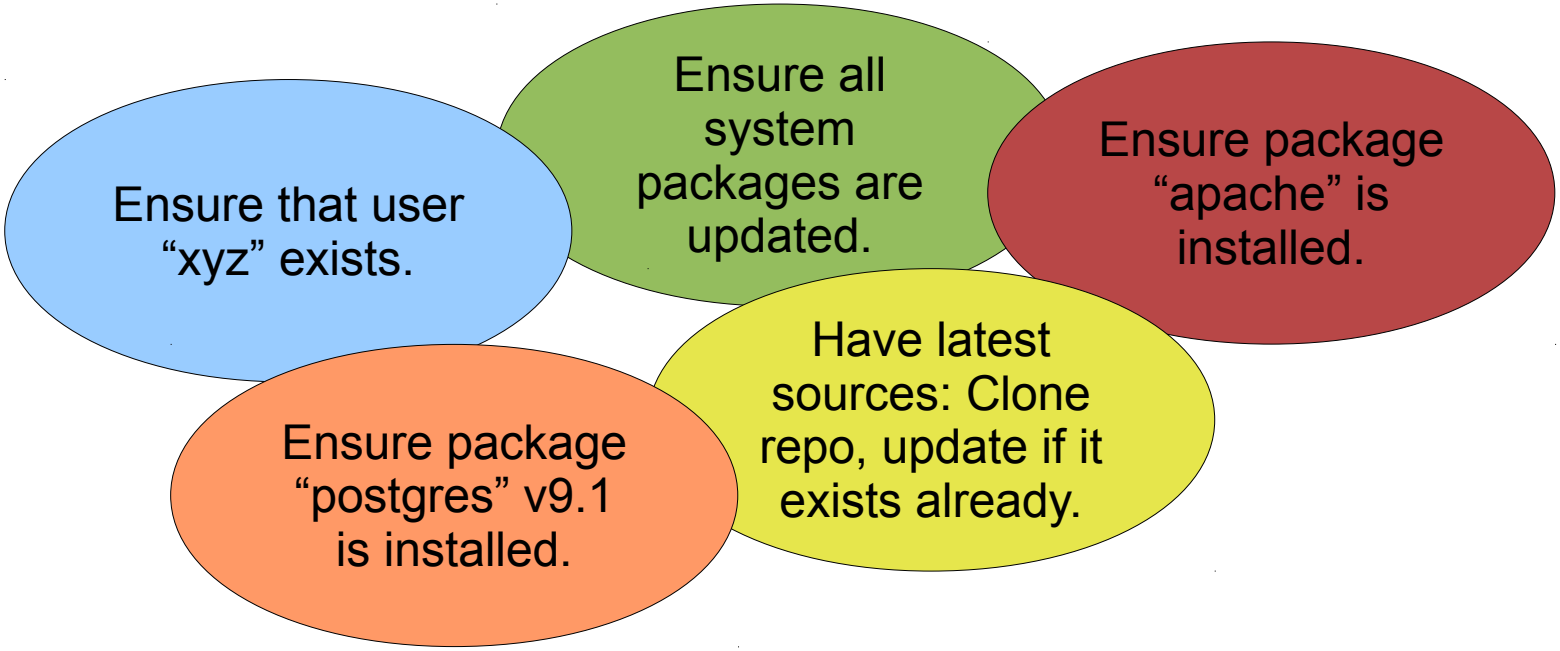
Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools

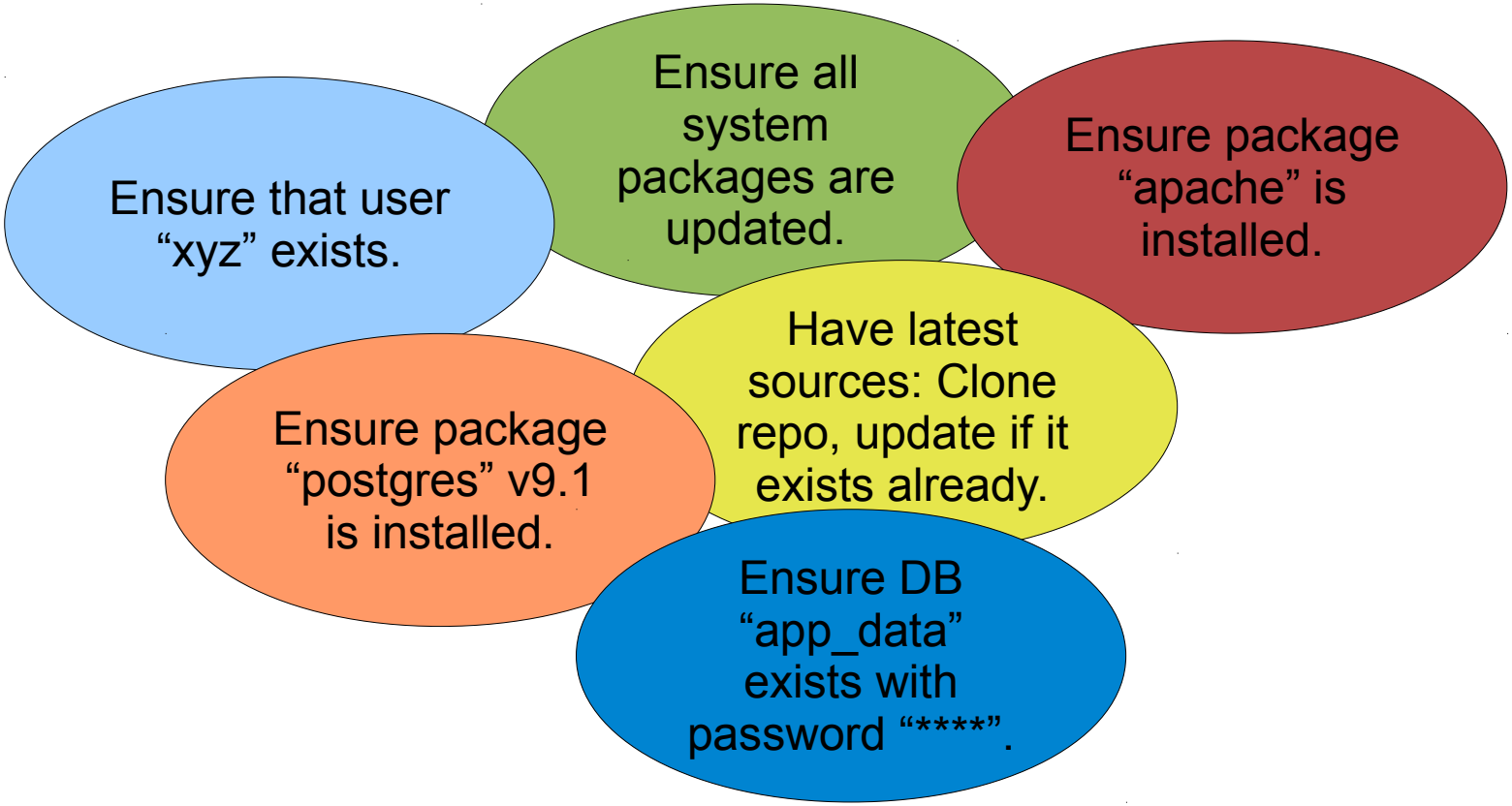
Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools

Describe the desired state



juergen@brendel.com
@BrendelConsult

CM Tools variety

juergen@brendel.com
@BrendelConsult

CM Tools variety

- Puppet *(2005)*
- Chef *(2009)*

“powerful, feature-rich, enterprisy”

juergen@brendel.com
@BrendelConsult

CM Tools variety

- Puppet *(2005)*
- Chef *(2009)*

“powerful, feature-rich, enterprisy”

- Salt *(2011)*
- Ansible *(2012)*

“simple, fast, good for most things”

juergen@brendel.com
@BrendelConsult

CM Tools variety

- Puppet *(2005)*
- Chef *(2009)*

"powerful, feature-rich, enterprisy"

- Salt *(2011)*
- Ansible *(2012)*

"simple, fast, good for most things"

- Fabric
- Scripts

"not really CM tools"

juergen@brendel.com
@BrendelConsult

Ansible: Intro and key concepts

Ansible overview

- “Orchestration engine” for CM and deployment
- Written in Python
- Uses YAML
- “Playbooks”
- Config specs or explicit commands

juergen@brendel.com
@BrendelConsult

Ansible simplicity

Key points:

- No central configuration server
- No key management
- No agent to install on target machine
- Explicit order

Requirements:

- Need SSH access (with key or password)
- Need Python installed on target machine

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

- Command
- Shell
- Script

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

- Command
- Shell
- Script

- Copy
- Sync
- Templates
- Line ops

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

- Command
- Shell
- Script

- Copy
- Sync
- Templates
- Line ops

- Install packages
- Users and groups
- Networking
- Services

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

- Command
- Shell
- Script

- Copy
- Sync
- Templates
- Line ops

- Install packages
- Users and groups
- Networking
- Services

- Repositories
- Message queues
- Monitoring
- Notification

juergen@brendel.com
@BrendelConsult

Modules

Hundreds of them. Know how to do stuff...

- Command
- Shell
- Script

- Copy
- Sync
- Templates
- Line ops

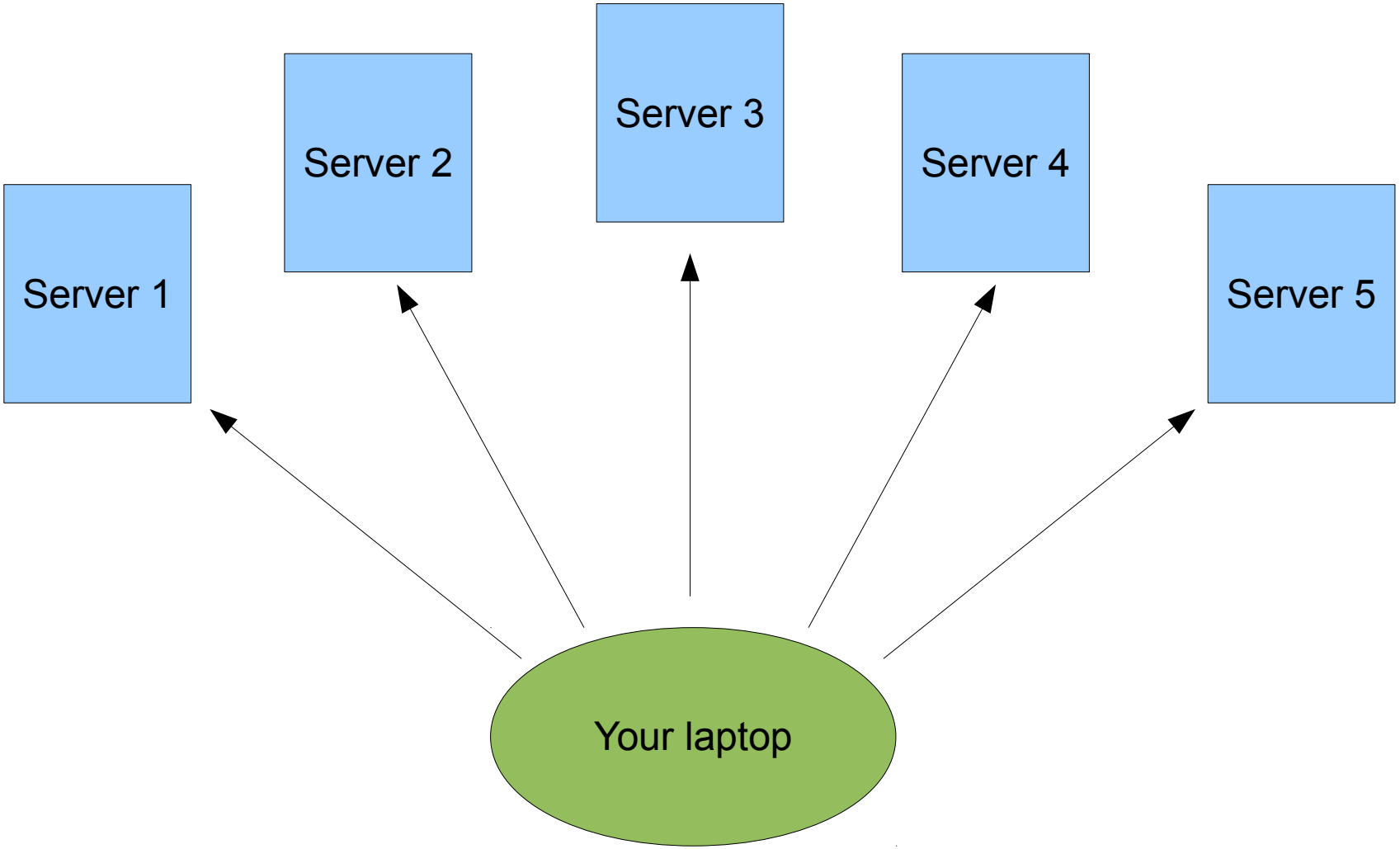
- Install packages
- Users and groups
- Networking
- Services

- Repositories
- Message queues
- Monitoring
- Notification

- Web servers
- Database servers
- Cloud infrastructure

juergen@brendel.com
@BrendelConsult

Ansible architecture

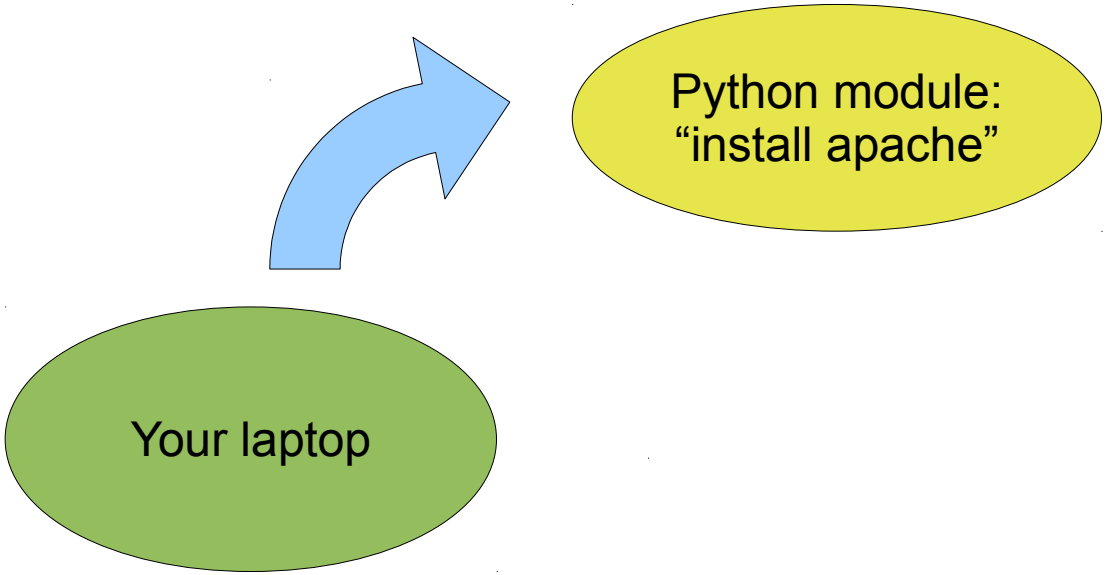


juergen@brendel.com
@BrendelConsult

How does it work?

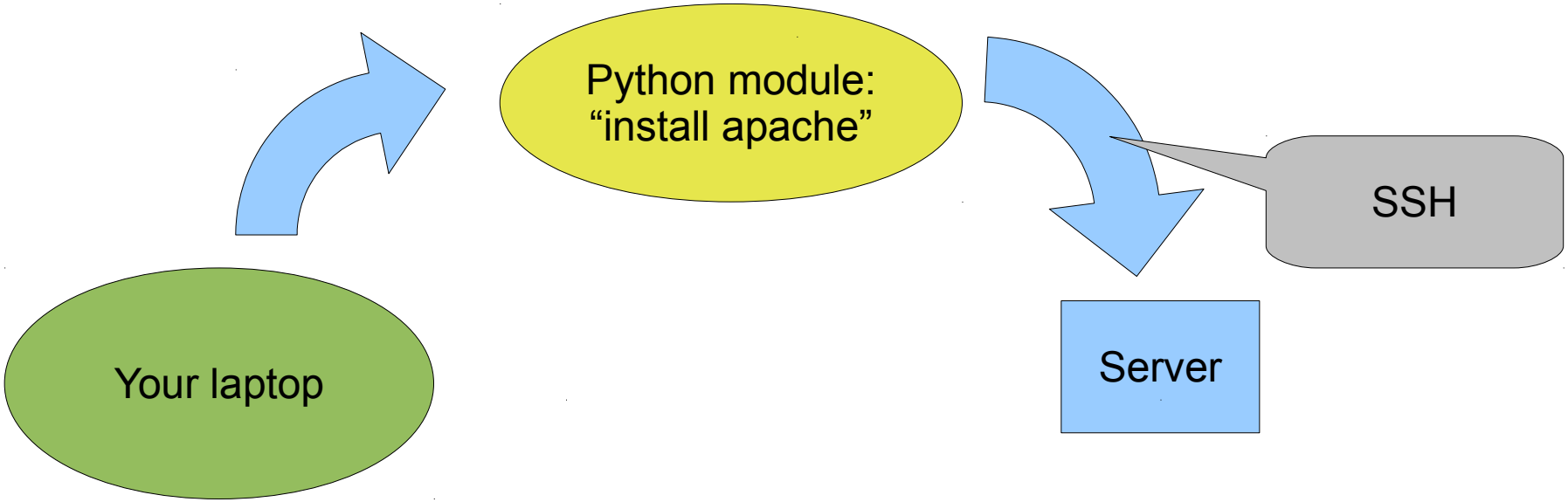
juergen@brendel.com
@BrendelConsult

How does it work?



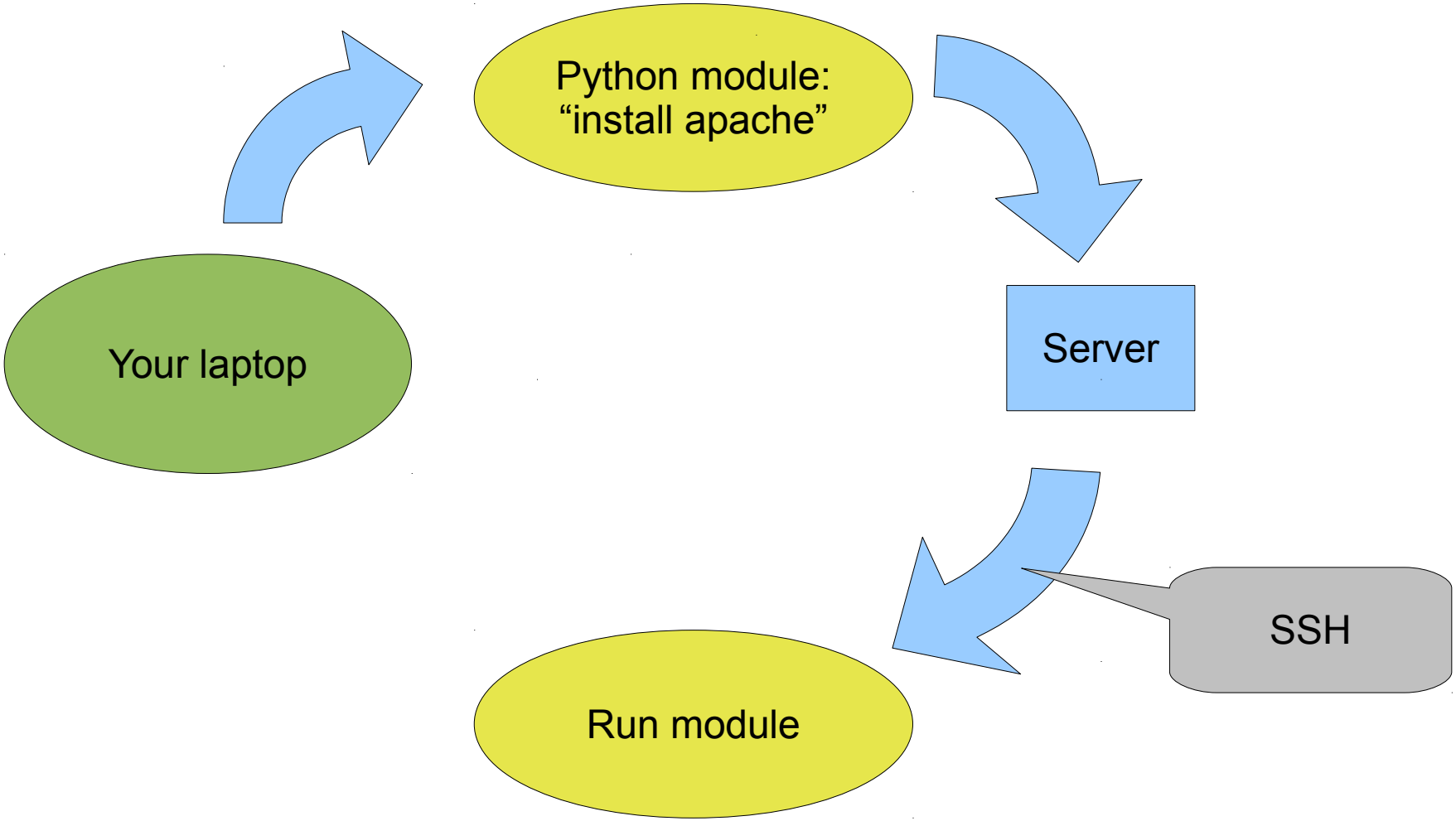
juergen@brendel.com
@BrendelConsult

How does it work?



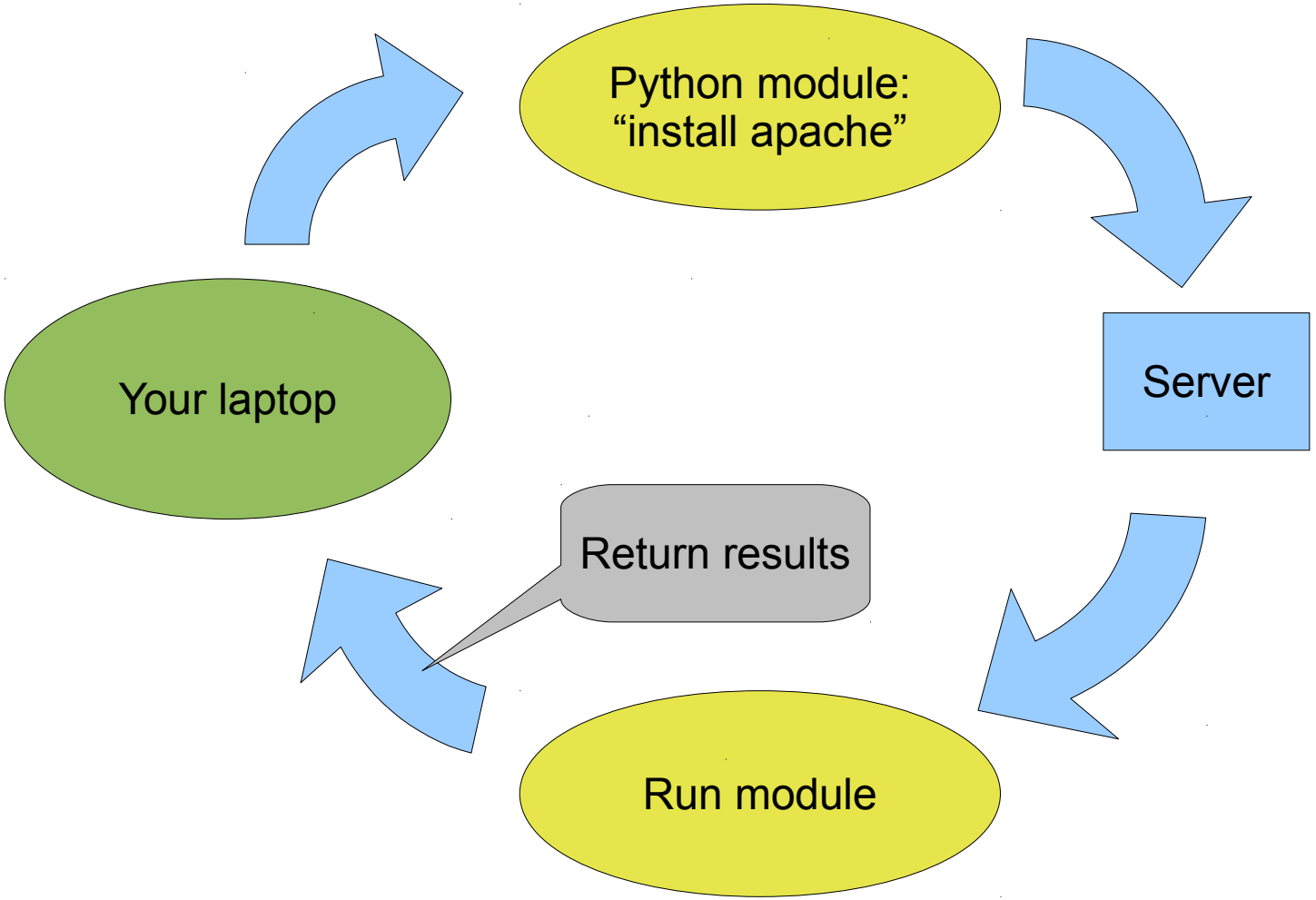
juergen@brendel.com
@BrendelConsult

How does it work?



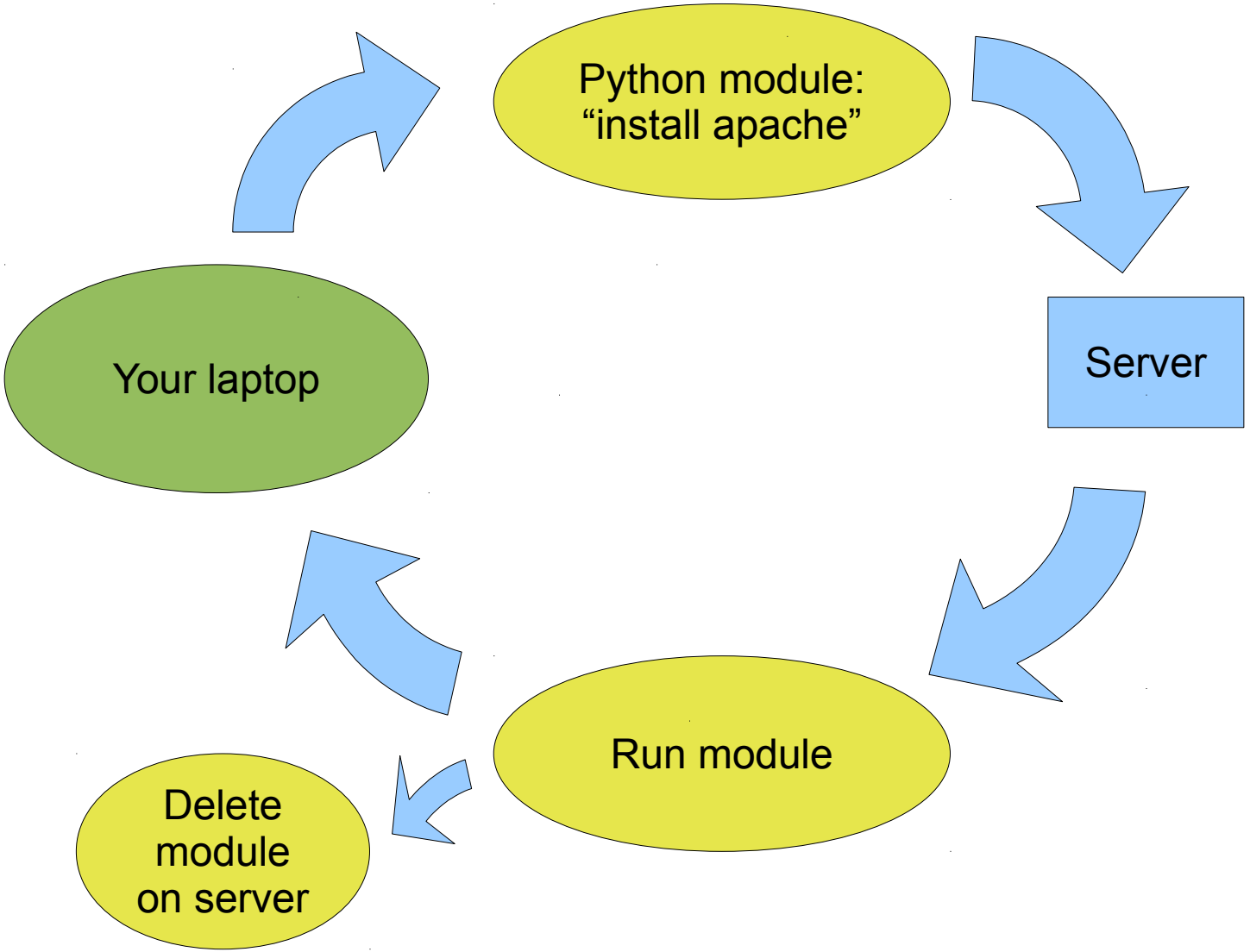
juergen@brendel.com
@BrendelConsult

How does it work?



juergen@brendel.com
@BrendelConsult

How does it work?



juergen@brendel.com
@BrendelConsult

Inventory and groups

Define host, organized in groups

- by function
- by location
- by hosting provider
- ...

juergen@brendel.com
@BrendelConsult

Inventory and groups

Define host, organized in groups

```
[europe]
server1.somehoster.co.uk
server2.otherhoster.de

[north-america]
host-a.serverhost.com
host-b.serverhost.com

[frontend]
server1.somehoster.co.uk
host-b.serverhost.com

[backend]
server2.otherhoster.de
host-a.serverhost.com
```


juergen@brendel.com
@BrendelConsult

Adhoc commands

Single commands, applied to groups

```
$ ansible -i hosts europe -a “uname -a”

$ ansible -i hosts frontend -a “/sbin/reboot” -f 3
```


juergen@brendel.com
@BrendelConsult

Playbooks

```
---
- hosts: all
  sudo: yes

  tasks:
  - name: Update the system
    apt: pkg=nginx state=latest

  - name: Create the user account
    user: name=appuser shell=/bin/bash state=present

  - name: Copy files to remote user's home
    copy: >
        src=files/names.txt dst=/home/appuser
        owner=appuser mode=0644
```


juergen@brendel.com
@BrendelConsult

Variables

```
---
- hosts: all
  sudo: yes
  vars:
    username: appuser

  tasks:
  - name: Create the user account
    user: >
        name={{ username }}
        shell=/bin/bash
        state=present
```


juergen@brendel.com
@BrendelConsult

Project layout 1

```
/
   my_hosts

   group_vars/
      all
      frontend
      backend
      europe
      north-america

   site.yml
```


juergen@brendel.com
@BrendelConsult

Project layout 2

```
/
   ansible.cfg

   deploy_hosts
   staging_hosts

   group_vars/
      all
      frontend
      backend
      europe
      north-america

   host_vars/
      server1.somehoster.co.uk
      host-b.serverhost.com

   site.yml

   roles/
      common/
         tasks/
            main.yml
         handlers/
            main.yml
         templates/
            sshd_config.j2
         files/
            my_script.sh
         vars/
            main.yml
      web/
         ...
      db/
         ...
```


juergen@brendel.com
@BrendelConsult

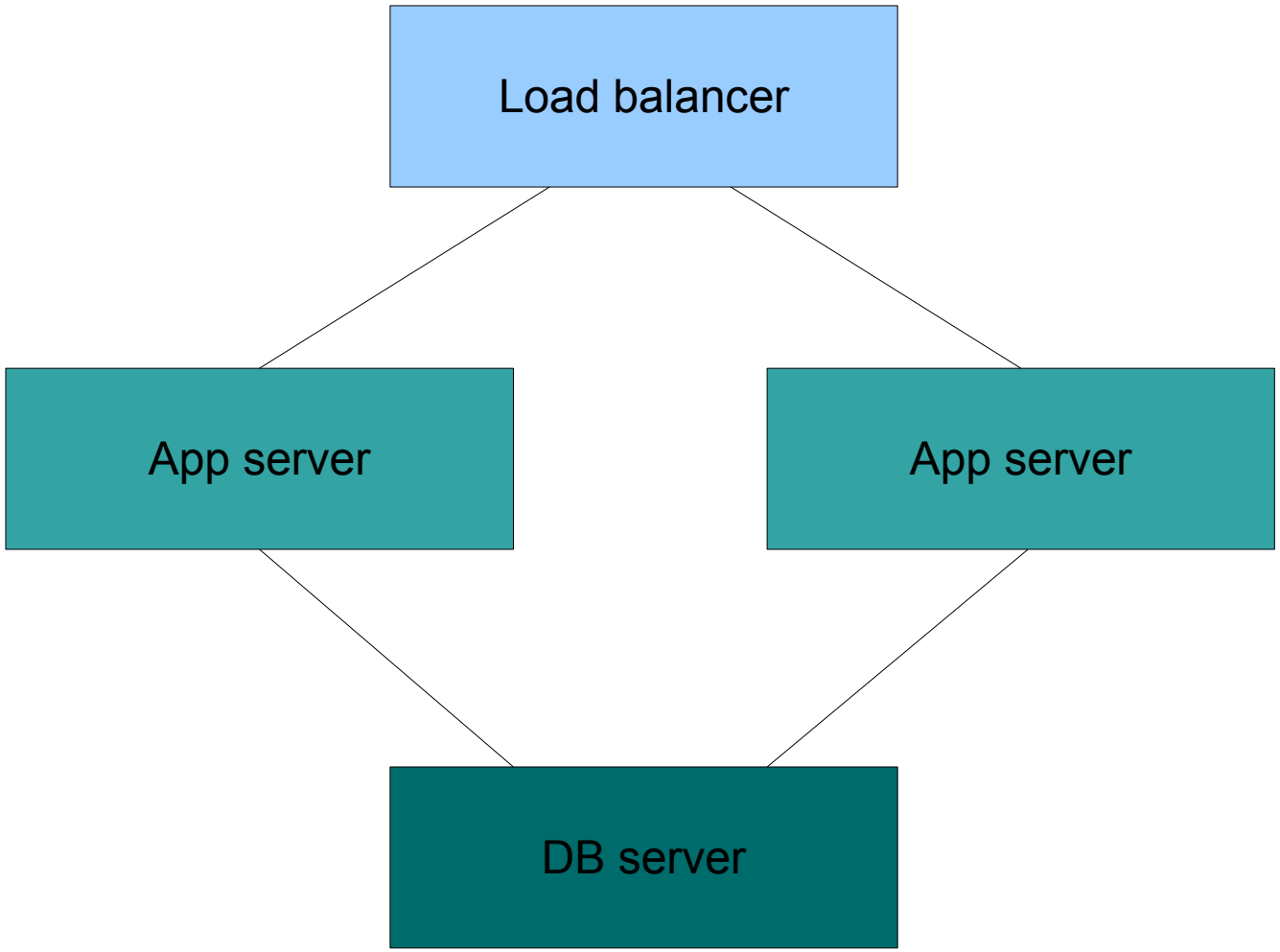
Playbooks with roles

```
---
- hosts: frontend
  sudo: yes
  roles:
    - common
    - web
```


juergen@brendel.com
@BrendelConsult

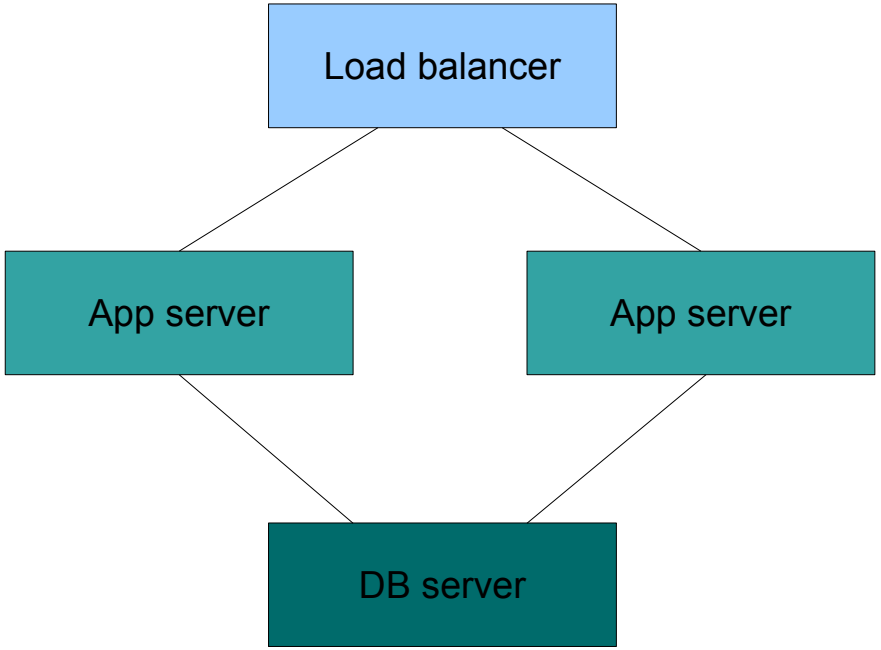
A non-trivial real world example

Example



juergen@brendel.com
@BrendelConsult

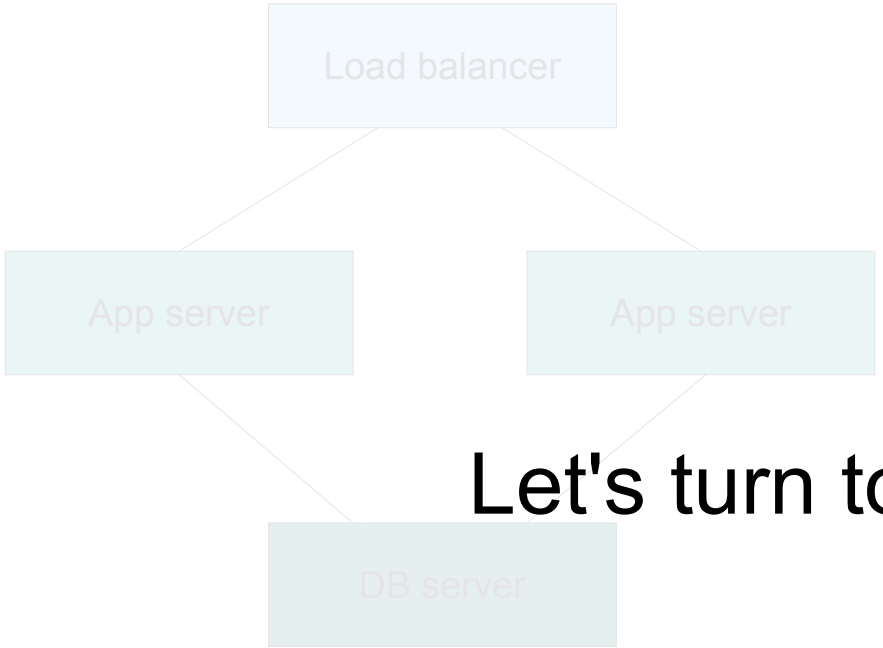
Example



- Ubuntu 13.10
- Amazon EC2
- Nginx
- Django app
- Postgres

juergen@brendel.com
@BrendelConsult

Example



- Ubuntu 13.10
- Amazon EC2
- Nginx
- Django app
- Postgres

Let's turn to the shell...

juergen@brendel.com
@BrendelConsult

Local environments

Dev, Test, Deploy

Common issues

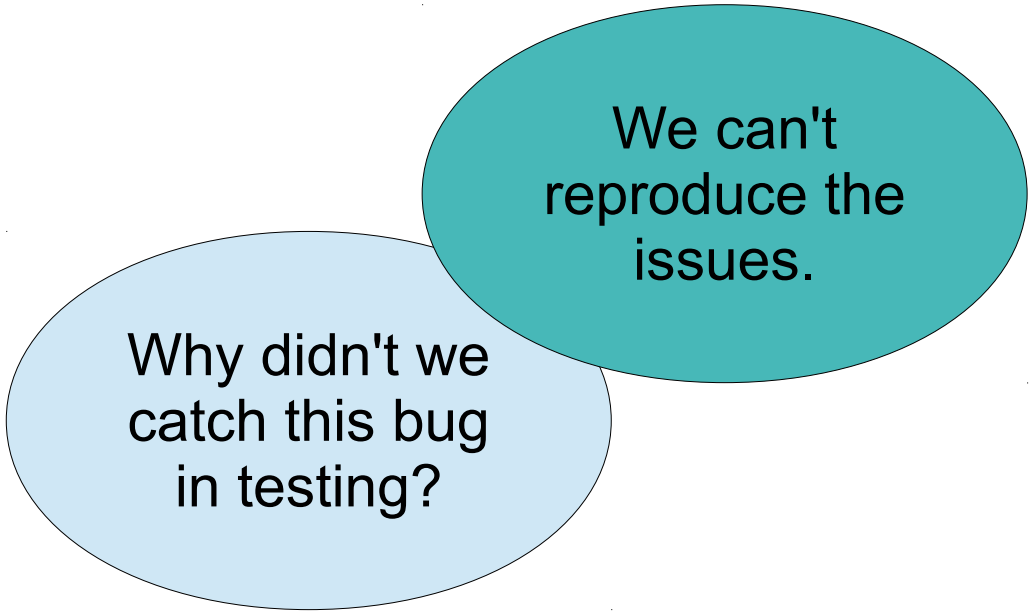
juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy

Why didn't we catch this bug in testing?

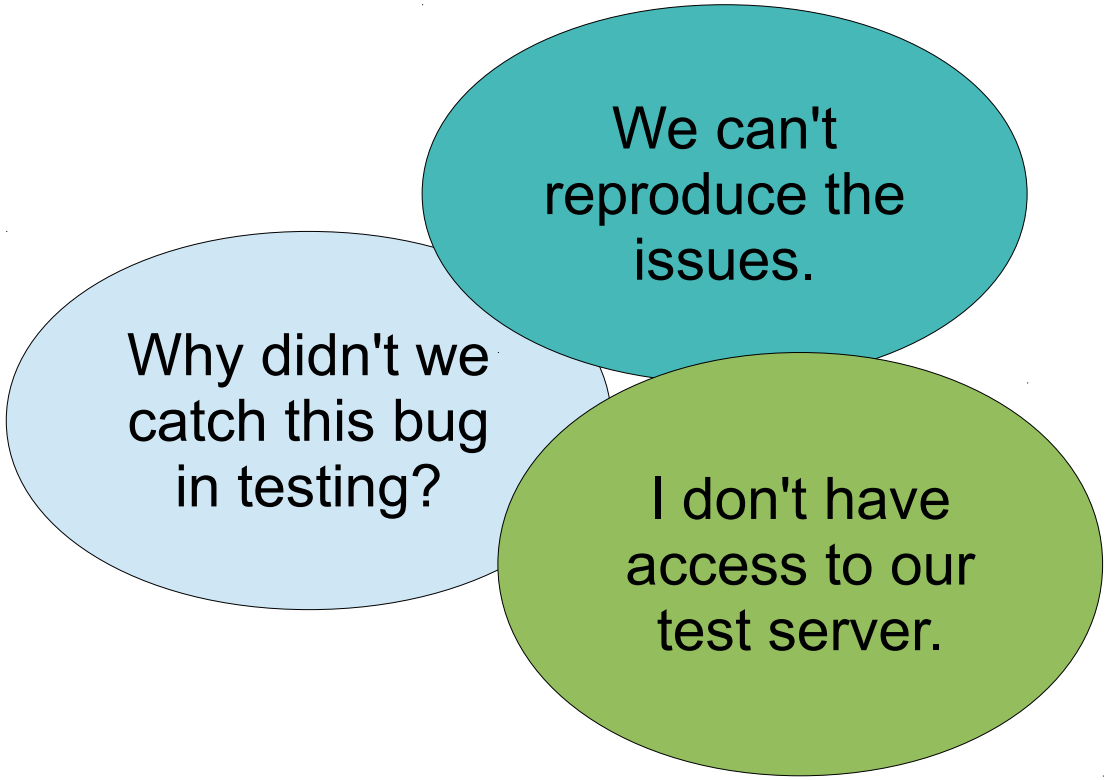
juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy



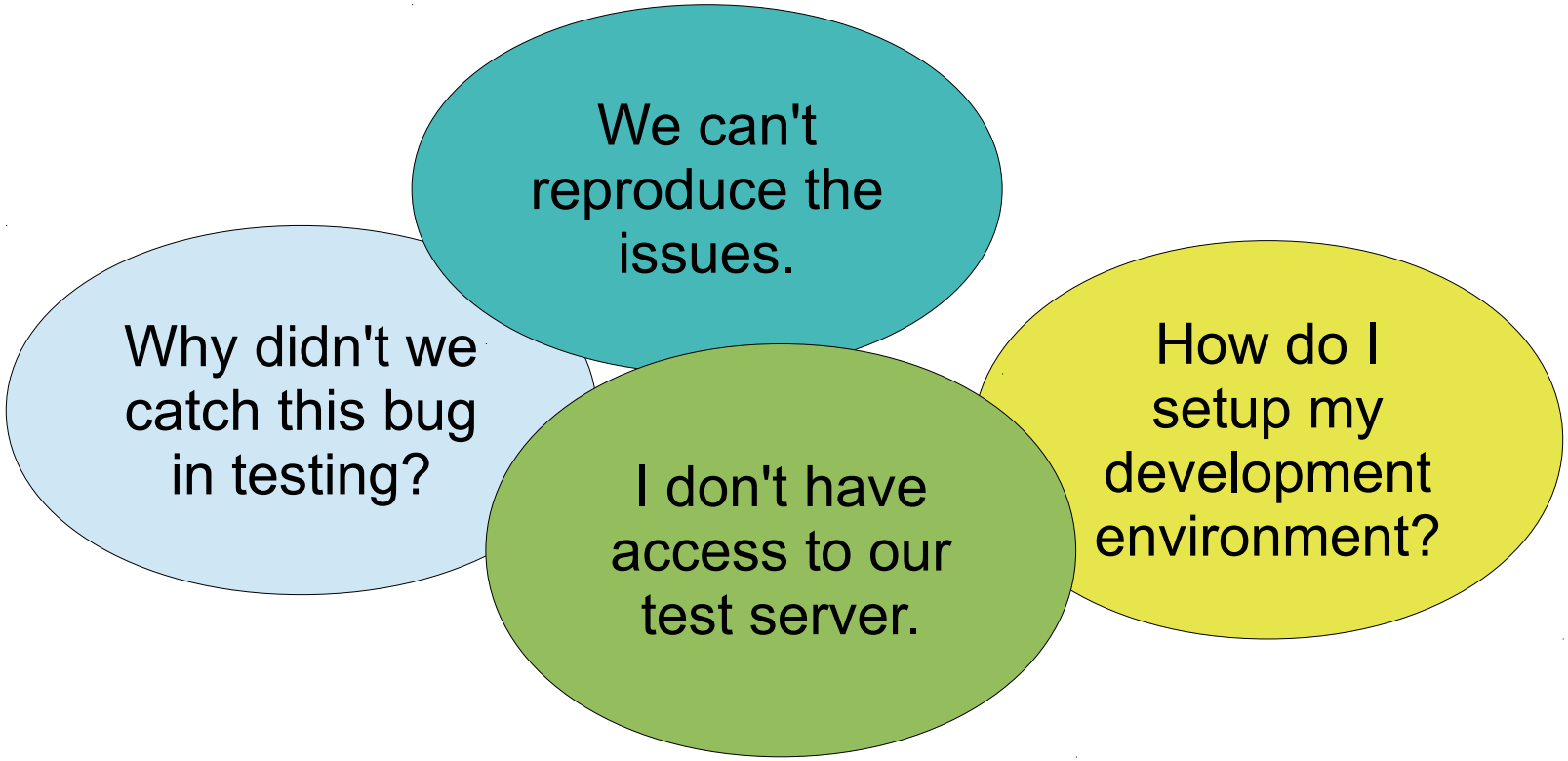
juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy



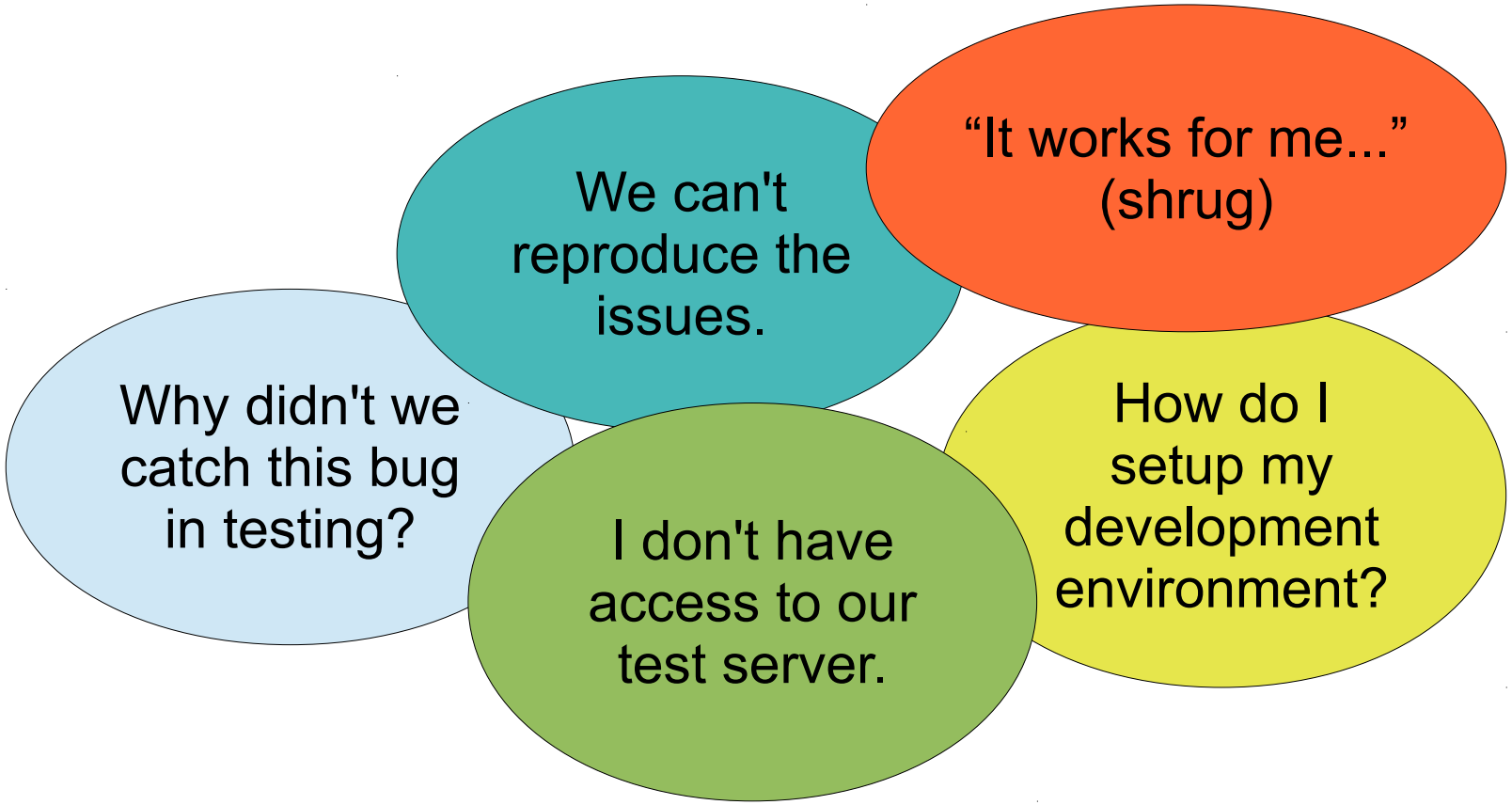
juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy



juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy



juergen@brendel.com
@BrendelConsult

Dev, Test, Deploy

Wouldn't this be nice instead?

Single command: Dev environment created
Single command: Test environment created

juergen@brendel.com
@BrendelConsult

Vagrant

- Use Vagrant to spin-up VMs
  - local (VirtualBox, VMware, etc.)
  - cloud (EC2)
- Use Ansible as 'provisioner'
- Make an inventory file with just your VM
- Point at same playbook as before

juergen@brendel.com
@BrendelConsult

Vagrant config: Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box       = "saucy64"
  config.vm.box_url   = "http://cloud-images.ubuntu.com/vagrant/..."
  config.vm.host_name = "myapp-test"
  config.vm.network "private_network", ip: "192.168.99.99"

  config.vm.provision "ansible" do |ansible|
     ansible.playbook         = "site.yml"
     #ansible.verbose          = "vvvv"
     ansible.inventory_path   = "vagrant_hosts"
     ansible.host_key_checking = false
  end
end
```


juergen@brendel.com
@BrendelConsult

Inventory: vagrant_hosts

```
[vagrant]
vagrant_host  ansible_ssh_host=192.168.99.99

[frontend-hosts]
vagrant_host

[applayer-hosts]
vagrant_host

[backend-hosts]
vagrant_host

[db-access:children]
applayer-hosts
backend-hosts

[appserver-access:children]
frontend-hosts
applayer-hosts
```


juergen@brendel.com
@BrendelConsult

Vars: group_vars/vagrant

```
#
# Variables that only apply to Vagrant instances.
#
---
ansible_ssh_user:                    vagrant
```


juergen@brendel.com
@BrendelConsult

Vagrant

```
$ vagrant up

...

$ vagrant provision
```


juergen@brendel.com
@BrendelConsult

Thank you very much!

Questions: juergen@brendel.com

Ansible docs: http://docs.ansible.com/
Vagrant: http://www.vagrantup.com/