inuits

OPEN SOURCE INNOVATORS

Pavel Grochal

darkless@inuits.eu

Configuration Management Automation

2022-03-08

For lecture:
PV077: UNIX – programování a správa systémů II

Patříte mezi

5% NEJÚSPĚŠNĚJŠÍCH lidí

inuits
OPEN SOURCE INNOVATORS

Patříte mezi

...% NEJ........................ lidí

inuits
OPEN SOURCE INNOVATORS

Museli jste
něco překonat,
abyste zde
dnes seděli.

inuits
OPEN SOURCE INNOVATORS

Linux Administrator

Introduction

name: **Pavel Grochal**

nickname: **darkless**

email: **darkless@inuits.eu** ; **pavel.grochal@inuits.eu**

keyboard layouts: **English(US)**, **Czech (QWERTY)**

favourite color: **blue**

favourite language: **Python**

favourite shell: **fish**

home igloo: **Brno**

Tech history

2004: **HTML, CSS, PHP, SQL, JS** – Websites (Firefox 1.0, IE 6, Opera 7)

2006: **dualboot Linux** - Ubuntu 6.10 (Edgy Eft)

2008: Bc. study **FI MUNI, JAVA** ecosystem, **MVC PHP** (CodeIgniter - thesis)

2011: Mgr. study **FI MUNI**, teaching - **PB162 Programování Java**

2012: **PYTHON, DJANGO**, IT Specialist @ **Academy of Sciences**

2013: **Networking & Firewalls** (CCNA certs)

2014: **Virtualization** (KVM, XEN, OpenShift, PXE), **CFGMGMT** (ANSIBLE, Chef)

2015+ OpenSource Consultant @ **INUITS.eu**

Overview

(1) **Brainstorming**

(2) **Real-life story** – don’t try this at home!

(3) Where does [**CFG MGMT**] fit in?

(4) **Concepts** of Configuration Management

01

Brainstorming

inuits
OPEN SOURCE INNOVATORS

Own VPS

/w Linux ?

inuits
OPEN SOURCE INNOVATORS

Example of provided services

- Web server
- Game server
- Mail server
- VPN
- DBs
- Custom Application server

Provided Service

Related services

- How to setup?
- How to configure?
- How to deploy and run?
- How to log?
- How to monitor?
- How to alert?
- How to secure access server?

Setup

Configuration

Deployment

Logging

inuits
OPEN SOURCE INNOVATORS

Monitoring

Alerting

Server access hardening

02

Real-life story – don’t try this at home!

inuits
OPEN SOURCE INNOVATORS

Real-life story introduction

Task: Install Samba(SMB) server on Ubuntu
(used as network storage for employees)

Actions:

- Prepare (Physical) Server, DHCP, add DNS records, (setup other services…)
- boot Linux, install Linux – Ubuntu (**USB / PXE**)
- Setup RAID storage (**mdadm**)
- Setup Monitoring (**Icinga**)
- Setup Backups (**Tape DRIVE + Bacula**)
- Setup FW (**iptables**)
- Setup SMB server ← **Focus part!**

How to setup SMB server?

1) **Google** “how to install SMB server”

   (https://www.google.com/search?q=how+to+install+SMB+server)

2) Click on **first link**

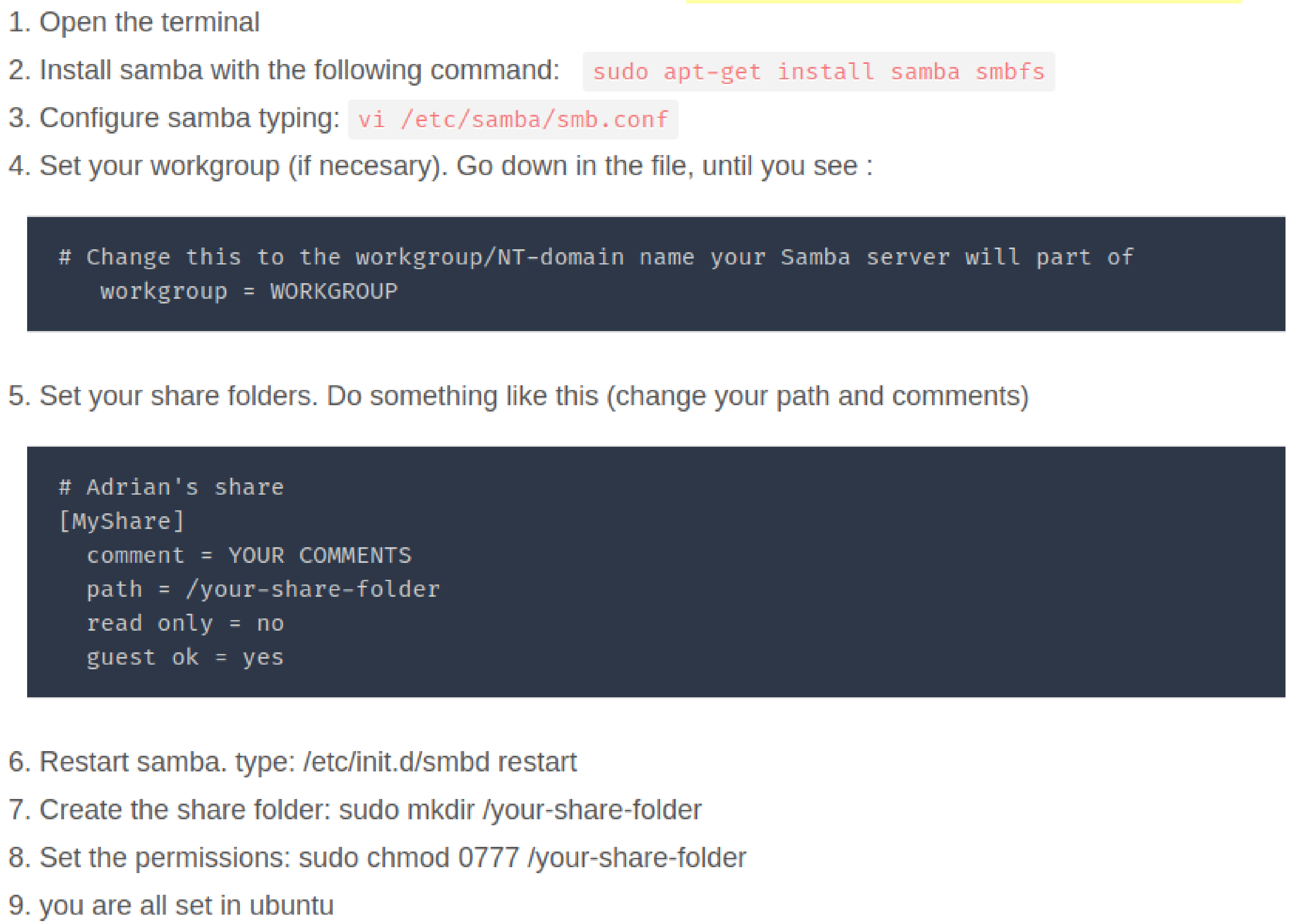
   (https://adrianmejia.com/how-to-set-up-samba-in-ubuntu-linux-and-access-it-in-mac-os-and-windows/)

3) **Copy&paste** commands to **terminal**

4) **Profit** !!!

Setting up the Samba File Server on Ubuntu/Linux:

FROM: https://adrianmejia.com/how-to-set-up-samba-in-ubuntu-linux-and-access-it-in-mac-os-and-windows/



1. Open the terminal
2. Install samba with the following command: `sudo apt-get install samba smbfs`
3. Configure samba typing: `vi /etc/samba/smb.conf`
4. Set your workgroup (if necesary). Go down in the file, until you see :

```
# Change this to the workgroup/NT-domain name your Samba server will part of
   workgroup = WORKGROUP
```

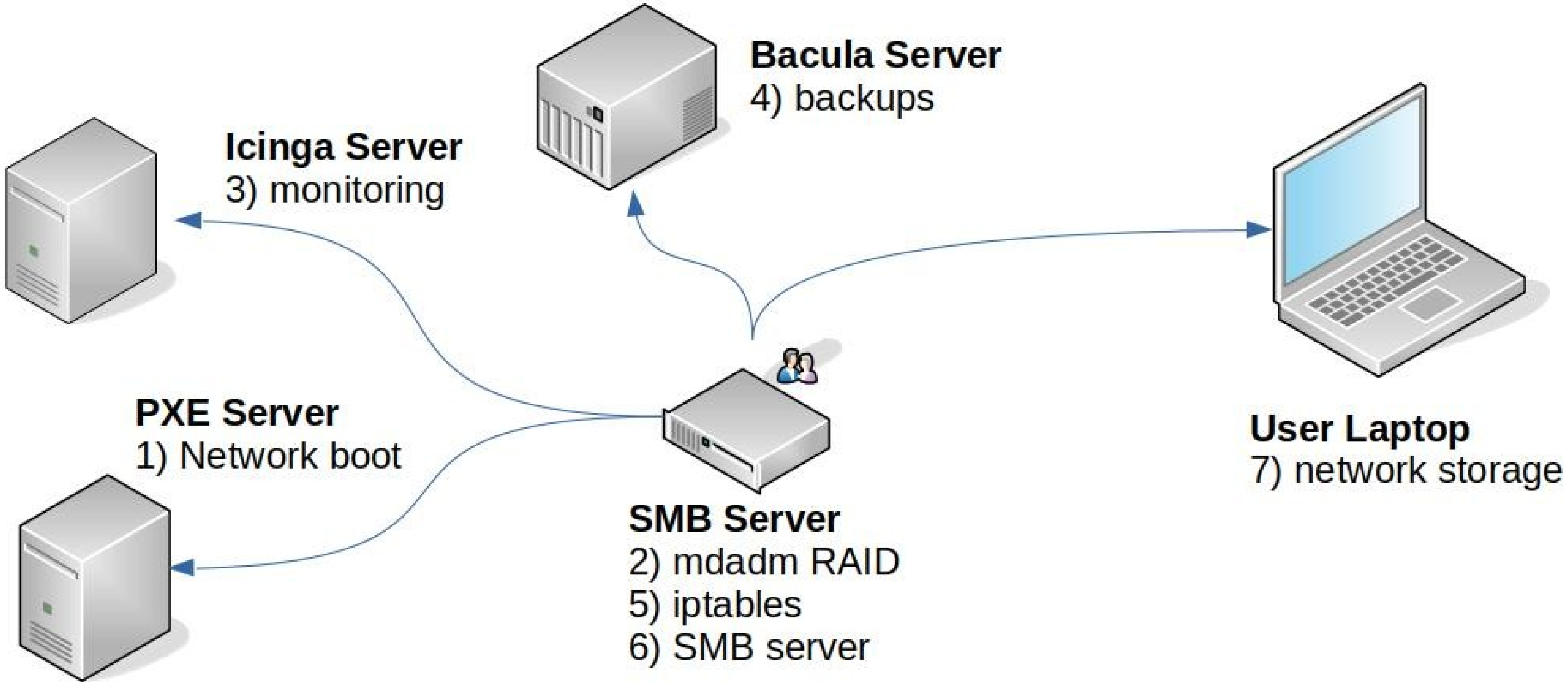
5. Set your share folders. Do something like this (change your path and comments)

```
# Adrian's share
[MyShare]
  comment = YOUR COMMENTS
  path = /your-share-folder
  read only = no
  guest ok = yes
```

6. Restart samba. type: /etc/init.d/smbd restart
7. Create the share folder: sudo mkdir /your-share-folder
8. Set the permissions: sudo chmod 0777 /your-share-folder
9. you are all set in ubuntu

inuits OPEN SOURCE INNOVATORS

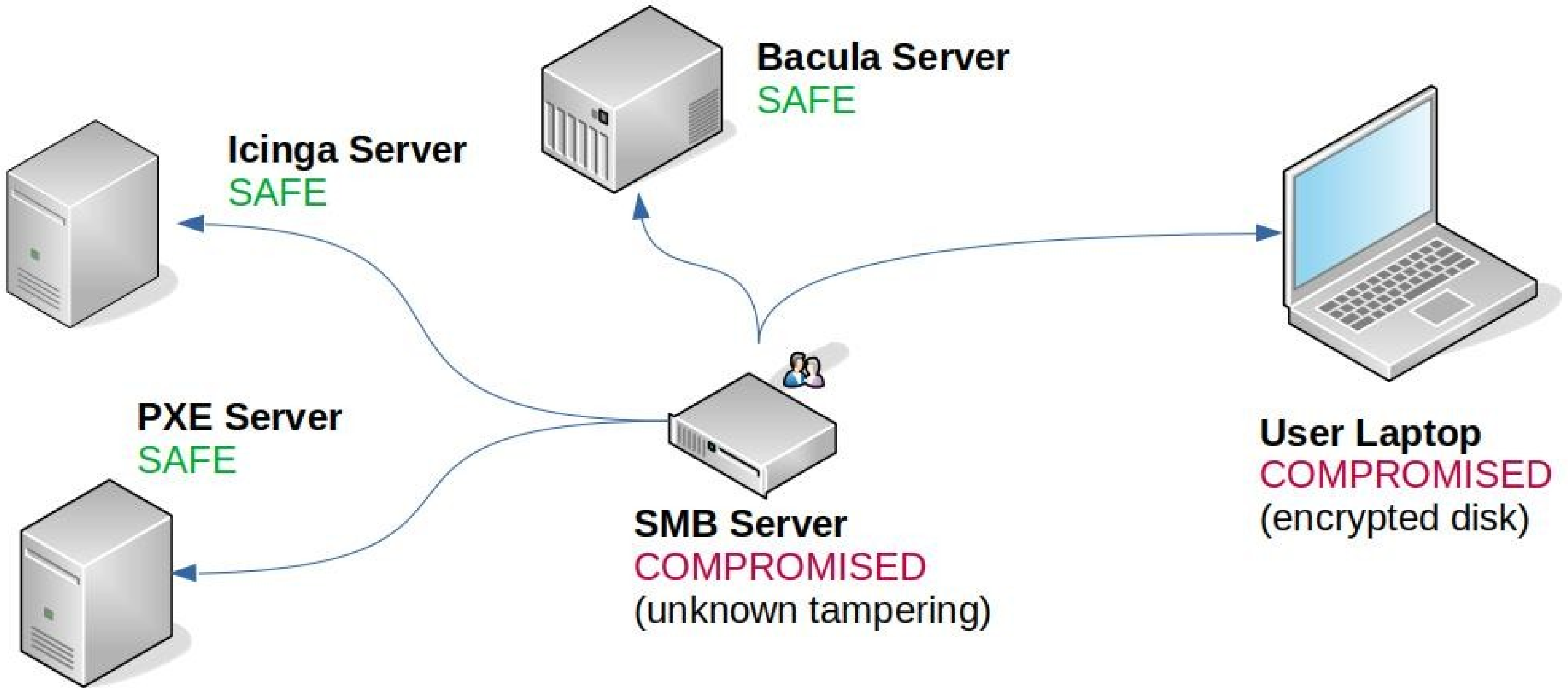
Production Overview



Ransomware Attack!

- One of the employees computer was **compromised** (Windows 7 Laptop)
- Shared network disk was **encrypted** demanding ransom
- Gained **access to server** due to security issue in SMB service

Attack Overview



First DAY mitigation

- Remove server from Network
- Inspect server via HW console in Data center
- Backup whole OS disk (dd) - for the future analysis
  - (User’s data are backed on TAPE DRIVE and restorable)
- Stop server

Difficulties

- Server was running for **1 year** periodically maintained.
- Nobody knew what was **ACTUALLY** installed on server. Only what SHOULD be installed.
- Custom-tweaked configurations for different users.

Rinse & Repeat?

- Reinstall compromised Linux server – Ubuntu (**USB / PXE**)
- Setup RAID storage (**mdadm**)
- Setup Monitoring (**Icinga**)
- Setup Backups (**Tape DRIVE + Bacula**)
- Setup FW (**iptables**)
- Setup SMB server (**again!**)

There has to be another way!

Configuration Management

03

Where does [CFG MGMT] fit in?

inuits OPEN SOURCE INNOVATORS

inuits
OPEN SOURCE INNOVATORS

What is Automation?

Google: “What is Automation?”

“Automation is the use of technology to perform tasks with reduced human assistance.”

* https://www.redhat.com/en/topics/automation

IT Automation maybe?

“**IT automation**, sometimes referred to as **infrastructure automation**, is the use of software to create **repeatable instructions and processes** to replace or reduce human interaction with IT systems. Automation software works within the confines of those instructions, tools, and frameworks to carry out the tasks with little to no human intervention.”

* https://www.redhat.com/en/topics/automation/whats-it-automation

Related services recap.

- How to setup?
- How to configure?
- How to deploy and run?
- How to log?
- How to monitor?
- How to alert?
- How to secure access server?

IT Automation topics

- Provisioning
- Configuration Management
- (Container) Orchestration
- IT migration
- Application deployment (CI/CD)
- Infrastructure as Code (IaC)

Provisioning

“Provisioning is the process of **setting up IT infrastructure**. It can also refer to the steps required to manage access to data and resources, and make them available to users and systems.

Provisioning is not the same thing as configuration, but they are both steps in the deployment process. Once something has been provisioned, the next step is configuration.

When the term "provisioning" is used, it can mean many **different types of provisioning**, such as **server** provisioning, **network** provisioning, **user** provisioning, **service** provisioning, and more.”

Configuration Management

“Configuration management is a process for **maintaining computer systems**, servers, and software in a **desired, consistent state**. It’s a way to make sure that a system performs as it’s expected to as changes are made over time. ”

* https://www.redhat.com/en/topics/automation/what-is-configuration-management

Orchestration

“Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services. Orchestration helps IT to more easily manage complex tasks and workflows.

Automation and orchestration are different, but related concepts.

In general, **automation refers to automating a single task**. This is different from **orchestration**, which is how you can **automate a process or workflow** that **involves many steps** across **multiple** disparate **systems**.”

Container Orchestration

“Container orchestration **automates** the **deployment**, **management**, **scaling**, and **networking** of **containers**. Enterprises that need to deploy and manage hundreds or thousands of Linux® containers and hosts can benefit from container orchestration.”

IT Migration

“An IT migration is the **shifting of data or software from one system to another**. Depending on the project, an IT migration could involve one or more kinds of movement: **Data** migration, **application** migration, **operating system** migration, and **cloud** migration.”

* https://www.redhat.com/en/topics/automation/what-is-it-migration

Application deployment (CI/CD)

“Continuous integration (CI) is the practice of **merging** all developers' **working copies** to a **shared mainline** several times a day.”

“Continuous delivery (CD) is a software engineering approach in which teams **produce software in short cycles**, ensuring that the **software can be reliably released at any time** and, when releasing the software, without doing so manually.”

“Continuous deployment (CD) is a software engineering approach in which **software functionalities** are **delivered frequently through automated deployments**.”

* https://en.wikipedia.org/wiki/Continuous_integration
* https://en.wikipedia.org/wiki/Continuous_delivery
* https://en.wikipedia.org/wiki/Continuous_deployment

Commonly Used Open Source Services for Ops

- **Provisioning**: Packer, Terraform, Pulumi, Kickstart
- **Configuration Management**: Puppet, Ansible, Chef, SaltStack
- **Container Orchestration**: Kubernetes, Nomad, Docker Swarm
- **Continous Integration/Delivery**: Jenkins, GitLab CI
- **Web servers**: Apache, Nginx, Caddy
- **Load Balancers** (including TLS termination): Nginx, HAProxy
- **Java application deployment**: JBoss, Wildfly, Tomcat
- **Databases**: MySQL, Postgres, CouchDB, MongoDB
- **Backup**: rsync, Bacula, BackupPC
- **Central log aggregation**: ELK, Fluentd, Graylog, Loki
- **Metrics and monitoring**: Zabbix, Icinga, Prometheus + Grafana
- **DNS server**: Bind, PowerDNS
- **Virtualization**: qemu/kvm, VirtManager, OpenNebula, Proxmox

Infrastructure as Code (IaC)

“Infrastructure as Code (IaC) is the **managing** and **provisioning** of **infrastructure** through **code** instead of through manual processes.

With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations. It also **ensures that you provision (and configure) the same environment every time**.”

General IaC Requirements

3 types of tools needed

Provisioning:

- Create me an instance of asset X
  - Container instance
  - VM instance
  - Service X configuration via API

Configuration Management (Desired state, Continuous Configuration Automation, …):

- Ensure that this file present / service is always running
- Set X with these permissions
- Ensure User Removed

Orchestration:

- Non frequent (complex to setup)
- Trigger action X on resource Y based on characteristics A,B and or C
- First do X here then do Y there
- One off actions
- Fully automated

Brief intro into Provisioning Tools

Packer: Automates the creation of any type of machine image.

Terraform: Codifies cloud APIs into declarative configuration files, i.e.: Infrastructure as Code.

Pulumi: similar to Terraform, in that you create, deploy, and manage Infrastructure as Code on any cloud.

Packer

Creates machine images from code:

1) **launch** virtual machine
2) **install** operating system
3) perform base **configuration** (Debian / Ubuntu: preseed or user-data, CentOS / RHEL: kickstart)
    - partitioning
    - base packages
    - users
    - networking
4) derives **machine image** from created virtual machine.

The resulting machine image can be used to launch new virtual machines from.

Terraform

- Describes infrastructure as code and manages described infrastructure
  - **init** Terraform and providers used (e.g. Qemu, AWS, …)
  - **validate** if the configuration is correct
  - **plan** the required changes to achieve the described infrastructure state.
  - **apply** the required changes.
  - **destroy** previously declared infrastructure.
- Functionality is not limited to virtual machines.
- Hashicorp Configuration Language (HCL).
- State Management stored in files
- **plan** & **apply** steps could be joined and part of CI/CD

Pulumi

- Similar to Terraform (but imperative)
- Supports and uses general purpose languages (Python, JavaScript, Go, C#, ...)
- Excellent code testing
- Mid-sized community
- State Management stored online

* https://www.pulumi.com/docs/intro/vs/terraform/#pulumi-vs-terraform

04

Concepts of Configuration Management

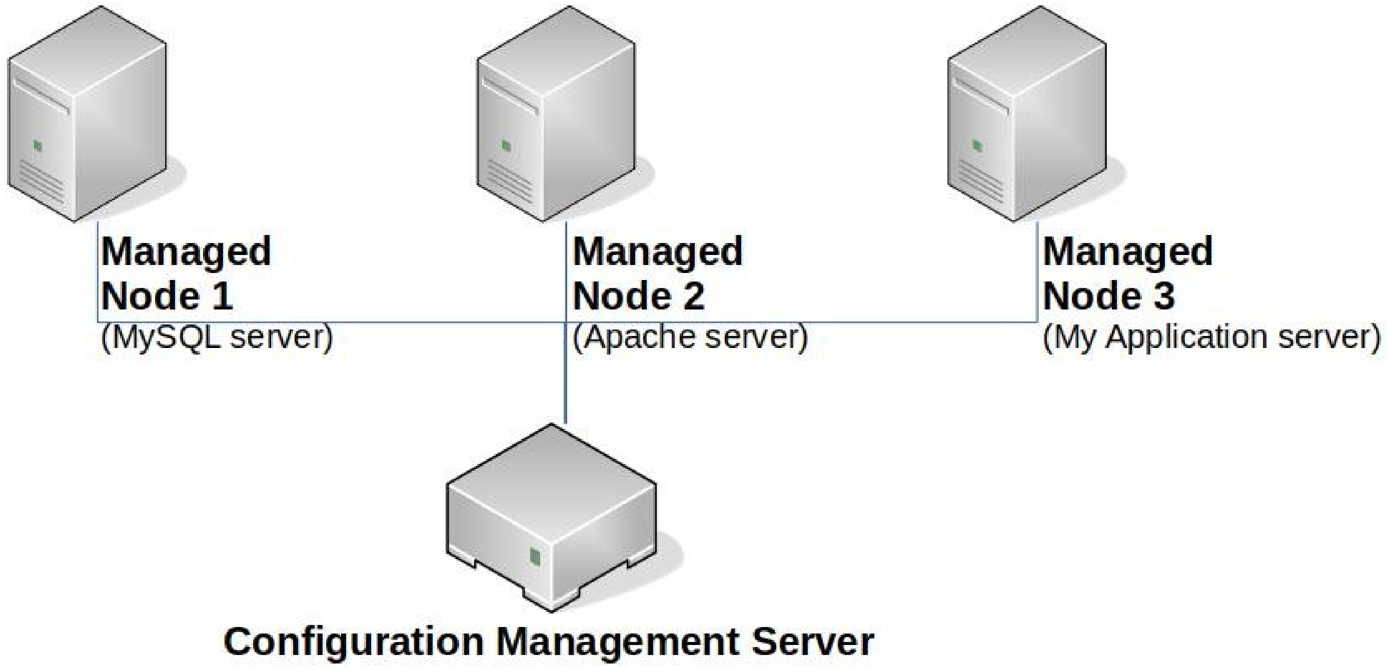
inuits

OPEN SOURCE INNOVATORS

Concepts of CFG MGMT

- Server-client / push-pull models
- Imperative / Declarative models
- Desired State
- Idempotence
- Comparison of Configuration Management tools
  - Puppet
  - Ansible
  - Chef
  - SaltStack

Topology Visualization



Push vs Pull model

Push model

Configuration Management Server **pushes** the configurations on the nodes.

1) **Server obtain** the **current state** of the node
2) **Compare** the **state** of the node with the stored **configuration**
3) **Perform actions** on the node to match the configuration

Pull model

Nodes (regularly) **pull** changes from the Configuration Management Server.

1) **Node obtain** the **stored configuration** from server - requires **dedicated client**
2) **Compare** the **configuration** with the current **state** of the node
3) **Perform actions** on the node to match the configuration

Imperative vs Declarative

Imperative model

- Concept similar to Imperative Programming languages (Python, Java, PHP, ...)
- States **HOW** things should be done.
- Mostly includes Implementation details.
- **Example**: Recipe for baking a cake.

Declarative model

- Concept similar to Declarative Programming languages (Proglog, DSL, ...)
- States **WHAT** the end result should be.
- Should exclude Implementation details.
- **Example**: Photo of how the cake should look like.

Desired State

- Corresponds with Declarative model.
- Setup Managed Nodes into an expected state.
- Enforces consistency, reproducibility and automation.

Idempotence

“Idempotence (UK: /ˌɪdɛmˈpoʊtəns/, US: /ˌaɪdəm-/) is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.”

Idempotence Example - wrong

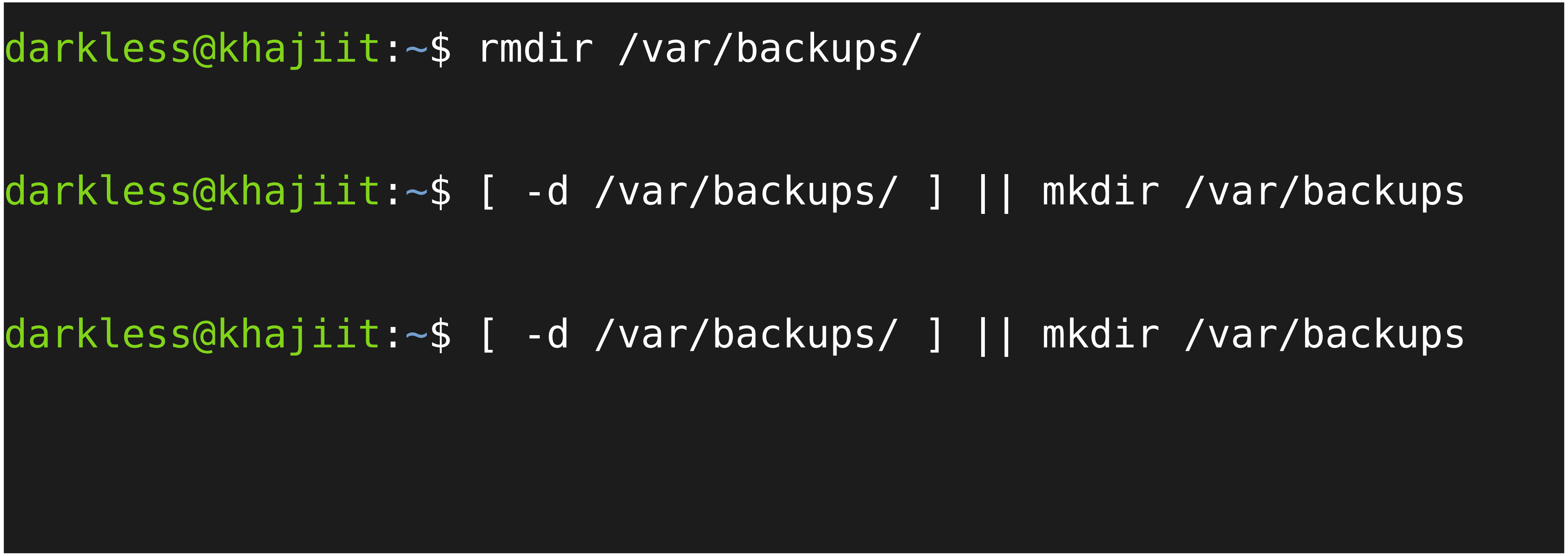
Create folder `/var/backups/`

```
darkless@khajiit:~$ mkdir /var/backups/

darkless@khajiit:~$ mkdir /var/backups/

mkdir: cannot create directory ‘/var/backups/’: File exists
```

Idempotence Example - better



```
darkless@khajiit:~$ rmdir /var/backups/

darkless@khajiit:~$ [ -d /var/backups/ ] || mkdir /var/backups

darkless@khajiit:~$ [ -d /var/backups/ ] || mkdir /var/backups
```

* https://stackoverflow.com/a/793867/2047157

Puppet vs Ansible vs Chef vs SaltStack

ANSIBLE

- YAML (Python)
- Push model
  - No client (SSH)
- Pull model
  - Client (Ansible Pull)

puppet

- DSL (Puppet DSL)
- Pull model
  - Client required

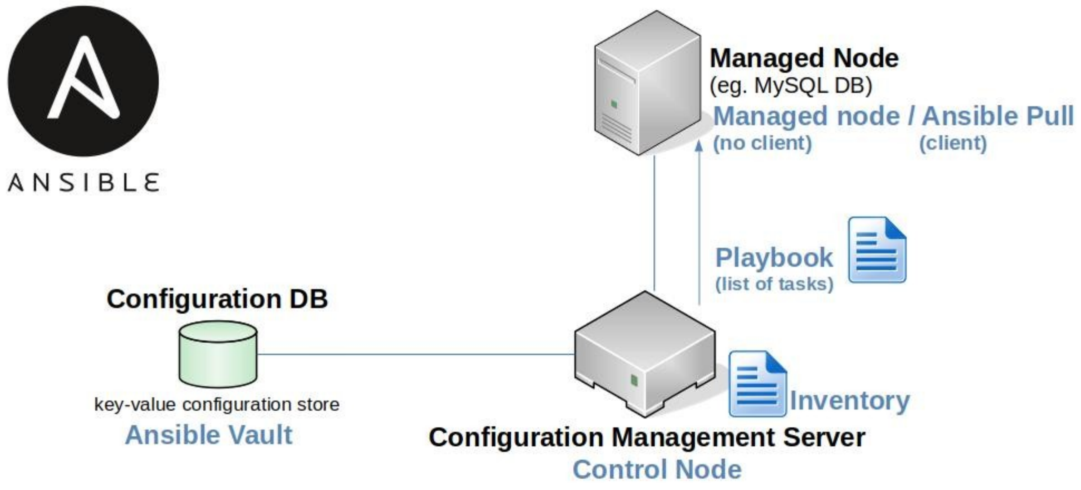
CHEF

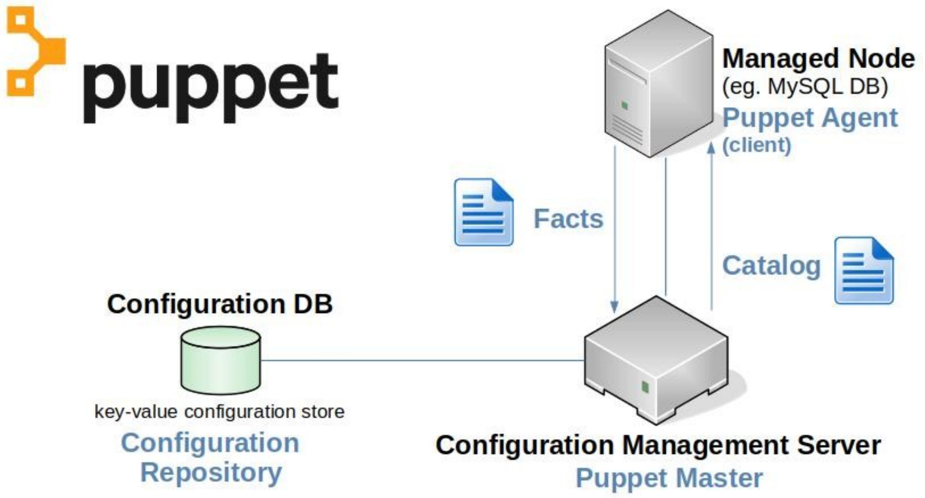
- DSL(Ruby)
- Pull model
  - Client required

SALTSTACK

- YAML(Python)
- Pull model
  - Client required
- Push model
  - No client (Salt SSH)

Ansible vs Puppet - topology





* This is **SIMPLIFIED** overview. See detailed information in respective guide.

Ansible vs Puppet – create folder

ANSIBLE

```
- ansible.builtin.file:
    path: /var/backups
    state: directory
```

puppet

```
class directories {
  file { '/var/backups':
    ensure => 'directory',
  }
}
```

Ansible vs Puppet – more options

ANSIBLE

```
- name: Create a /var/backups/ directory
  ansible.builtin.file:
    path: /var/backups
    state: directory
    owner: darkless
    group: games
    mode: '0755'
```

puppet

```
class directories {
  file { '/var/backups':
    ensure => 'directory',
    owner  => 'darkless',
    group  => 'games',
    mode   => '0750',
  }
}
```

inuits
OPEN SOURCE INNOVATORS

inuits

OPEN SOURCE INNOVATORS

Pavel Grochal
pavel.grochal@inuits.eu

INUITS bvba

Essensteenweg 31
2930 Brasschaat
Belgium
BE 0891.514.231

Contact:
+32.380.821.05
info@inuits.eu

inuits.eu

INUITS s.r.o.

Brno Igloo
Pekařská 962/72
602 00 Brno

Prague Igloo
ImpactHub
Drtinova 557/10
150 00 Prague