Welcome to

Controlling a High Security Environment with Ansible

Henrik Lund Kramshøj
hlk@pasientsky.no
hlk@patientsky.com

En enklere hverdag med PasientSky

Health data

Doctors appointment

Doctors Journals

Medical data

Prescriptions

...

Obviously this means personal data

Ansible: provisioning, configuration management, security

Pasient Sky

Open Source ☺

Simple playbooks and ad-hoc commands

Well supported on mainstream OSs

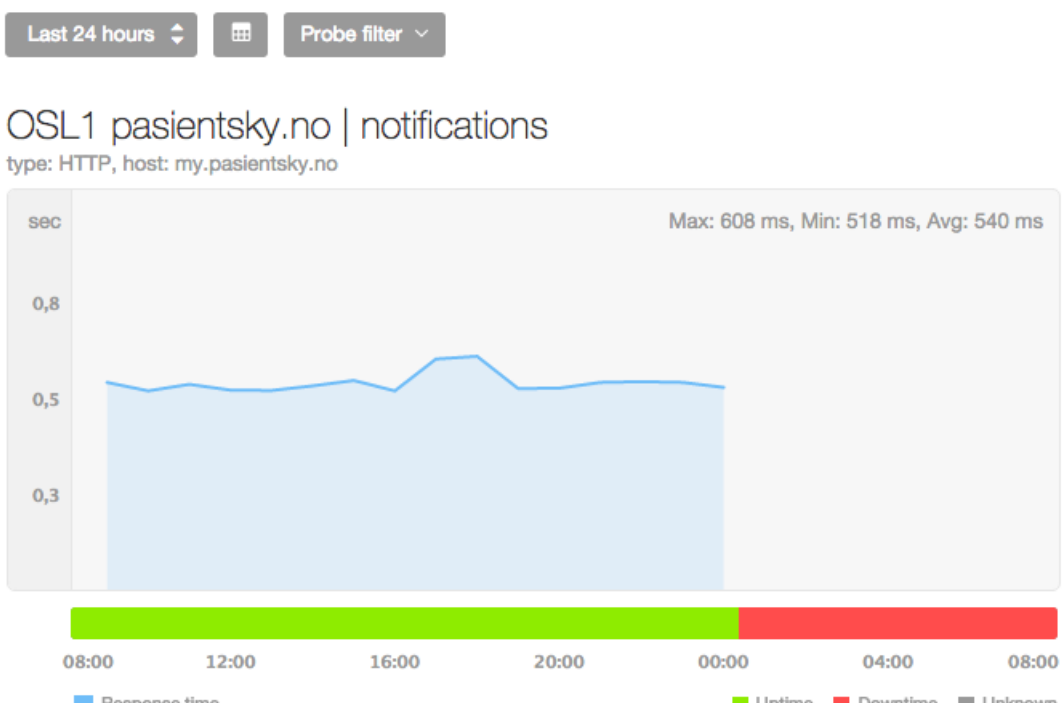
over 200 modules in the core

Supports almost anything which has SSH+Python

Currently 100s of servers

`http://www.ansible.com/`

Note: we dont use Tower



What we learnt about Ansible

.

Easy to get started - YAML playbooks

Easy to configure services

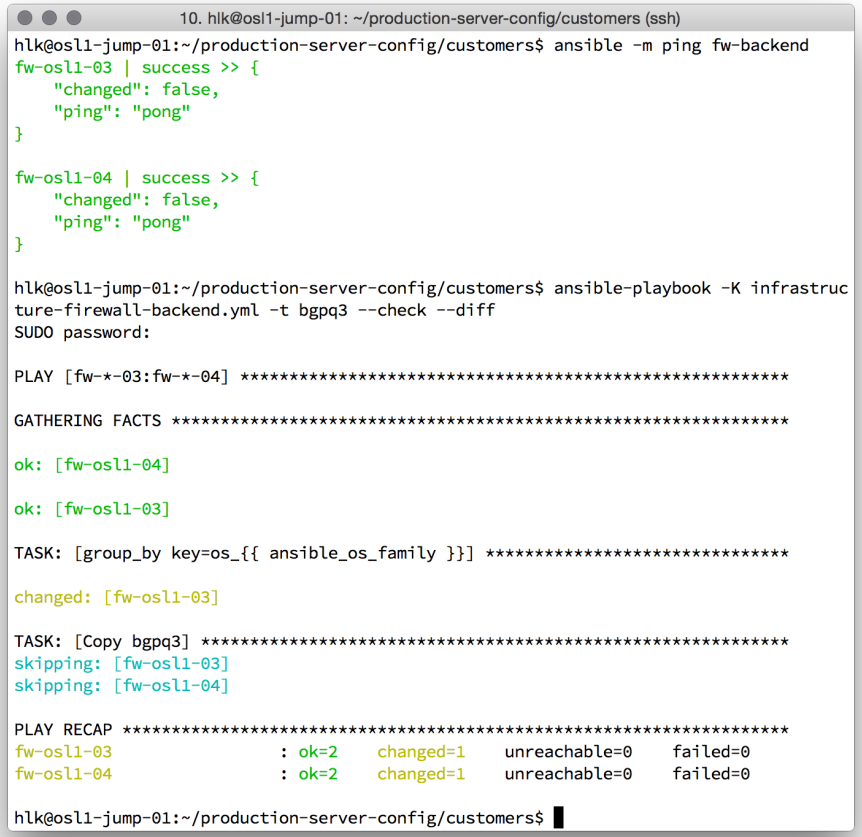
Roles sometimes suck

- too many files in too many directories

Using more flat playbooks nice

Long lists of settings like sysctl

We will continue with Ansible



Why Ansible brings Higher Security

Pasient Sky

```
# VPN tunnels via customer VPN server
pass quick proto { esp, ah } from any to {{ public_ip_prefix }}.59
pass quick proto { esp, ah } from {{ public_ip_prefix }}.59 to any
```

We can rebuild advanced servers in 15 minutes

Example complete Log environment from single playbook:

- Syslog servers, PostgreSQL database, Logstash parser, software and rules, Elasticsearch indexing servers
- with Kibana frontend - in about 150-200 lines of playbook!

From a base Ubuntu install with no manual steps, other than starting Ansible

Settings are saved in playbooks - documented and readable

Config files are templated and

testing, staging and production use **the exact same playbooks/configs**

What Ansible brings in a High Security Environment

Pasient Sky

We can deploy a complete IDS solution in 15 minutes

A complete Suricata IDS environment from a single playbook,

- Suricata IDS
- Rulesets - configuration files the same across environments
- Cron - jobs for updating rules
- Elasticsearch indexing servers
- Kibana front end

Consistency

From a base Ubuntu install with no manual steps, other than starting Ansible

Audit servers? Run Ansible - anything changed manually? `--check --diff`

Plan-Do-Check-Act process - very ISO 27001 compatible

Templates

Pasient Sky

```
jdbc \{
  # Postgres jdbc connection string to our database
  jdbc_driver_library => "/usr/share/java/postgresql-jdbc4-9.2.jar"
  jdbc_driver_class => "org.postgresql.Driver"
  jdbc_connection_string => "jdbc:postgresql://{{ private_ip_prefix }}.22.100:5432/Syslog"
```

We can test the SAME CONFIGS in multiple environments

Using variable group vars, host vars, templates

- Site specific data,
- RFC1918 subnets, IPs, port numbers, DNS, NTP
- Domain names, updates servers,
- environment: development, staging, production
- Passwords, S3 access keys, administrative users
- Service names: ssh (Debian), sshd (OpenBSD)
- ...

No untested changes brought into production

Update security parameters

```
- lineinfile:
    dest=/etc/ssh/sshd_config state=present
    regexp='PasswordAuthentication'
    line='PasswordAuthentication no'
  notify: restart sshd
  tags:
    - sshd
```

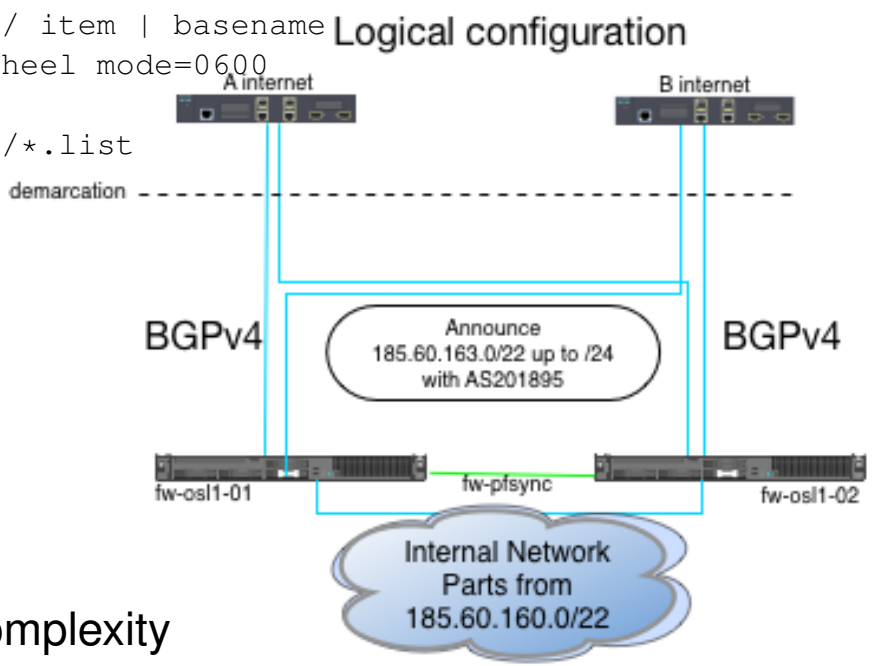
Combined with:

```
- name: restart sshd
    service: name= service_sshd  state=restarted
```

Never forget to restart a service after changing config

Cluster firewalls always consistent

```
- name: copy PF tables
  template:
    src=roles/infrastructure-firewall/files/pf-tables/ item | basename
    dest=/etc/pf/ item | basename  owner=root group=wheel mode=0600
  with_fileglob:
      - roles/infrastructure-firewall/files/pf-tables/*.list
  notify:
    - reload pf
```



Updating firewall tables

Updating rulesets

Updating BGP import filters

Update all parameters and roll-out made easy

Skills required to update production systems, low complexity

Less manual steps => more reliable

Golden rules

Pasient Sky

Always use descriptive name: so people know why/what is being done

Dont use lineinfile, if changing more than a few lines, use a template

Dont use copy, always use a template (if syntax permits)

Manual changes should be banned

Use tags liberally, tags: pf.conf, only update THIS thing

Try to gather a project/feature/setup in single playbook

Example: logging setup with both PostgreSQL and Elasticsearch in same

Use versioning for your playbooks, we use Git

And learn your $EDITOR - search and replace in lots of files ☺

Questions?

Henrik Lund Kramshøj
hlk@pasientsky.no
hlk@patientsky.com

https://pasientsky.no/

You are always welcome to send me questions later via email