Automating Security, Compliance, and Configuration with SaltStack

CODEQT1454LV

Evyn Price

Real Time Operations Systems Administrator

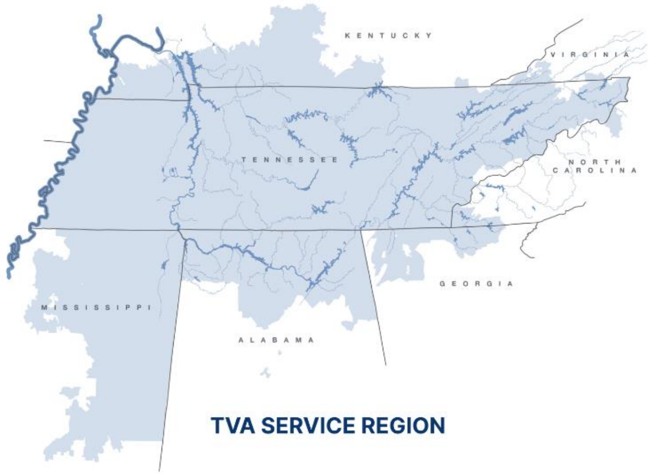
TVA TENNESSEE VALLEY AUTHORITY

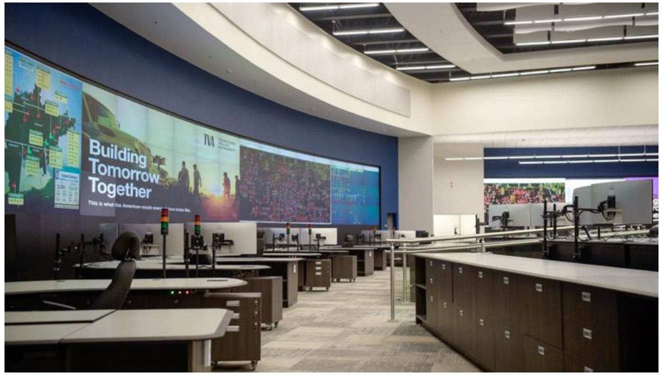
How much time do you spend manually configuring your environment?

TENNESSEE VALLEY AUTHORITY

About TVA

- Nation’s largest public power provider
- Serve over 10 million customers across seven states
- Manage fleet of energy-generating plants and over 16,000 miles of transmission lines
- Partner with 153 local power providers to serve the region





What is this session about?

- Showcase the value of automated configuration management
- Demonstrate real use cases for:
  - Security
  - Operations
  - Compliance

Disclaimer

TVA TENNESSEE VALLEY AUTHORITY

Security - Enforcement of Cybersecurity Hardening

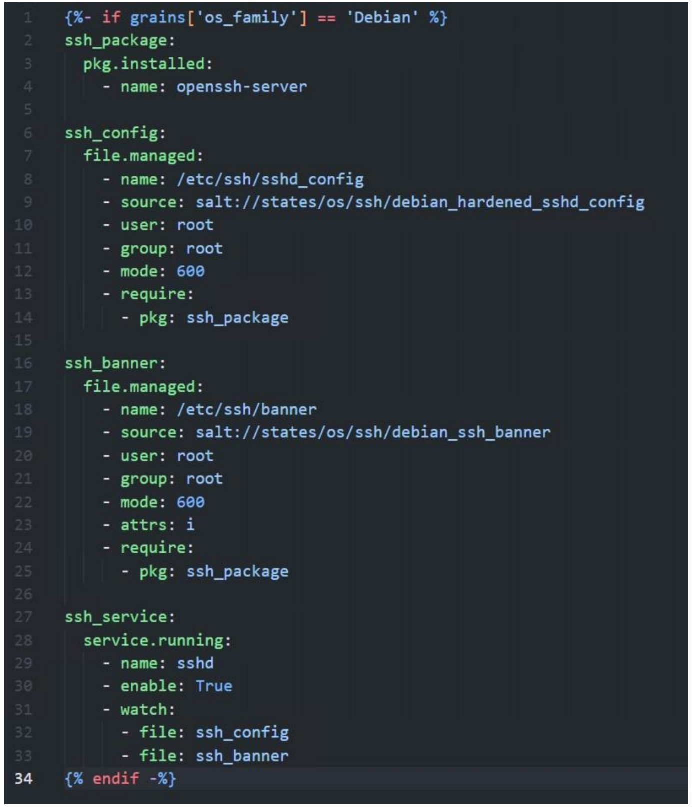
Goal:

- Automatically enforce cybersecurity configurations (DoD STIG, CIS Benchmarks, etc.)

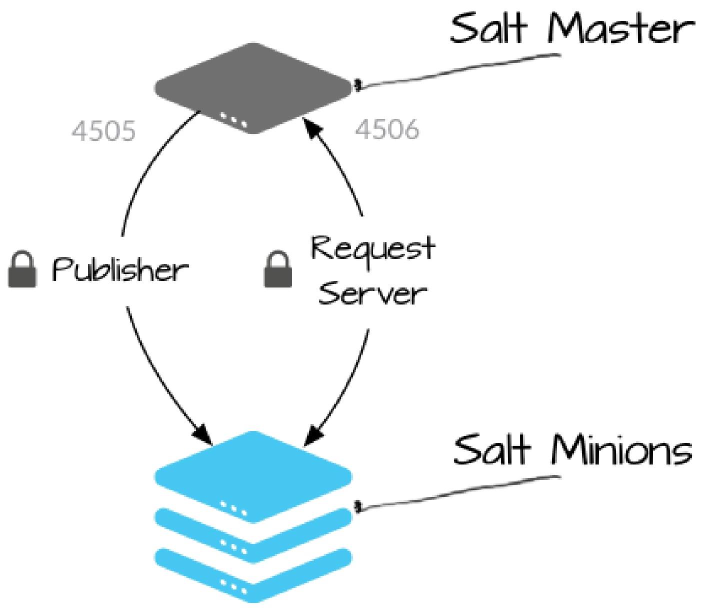
Challenges:

- Continuously identifying and resolving deviations from baseline
- Tracking and implementing changes to standards
- Enforcing standards across entire environment, including multiple operating systems and non-domain attached assets

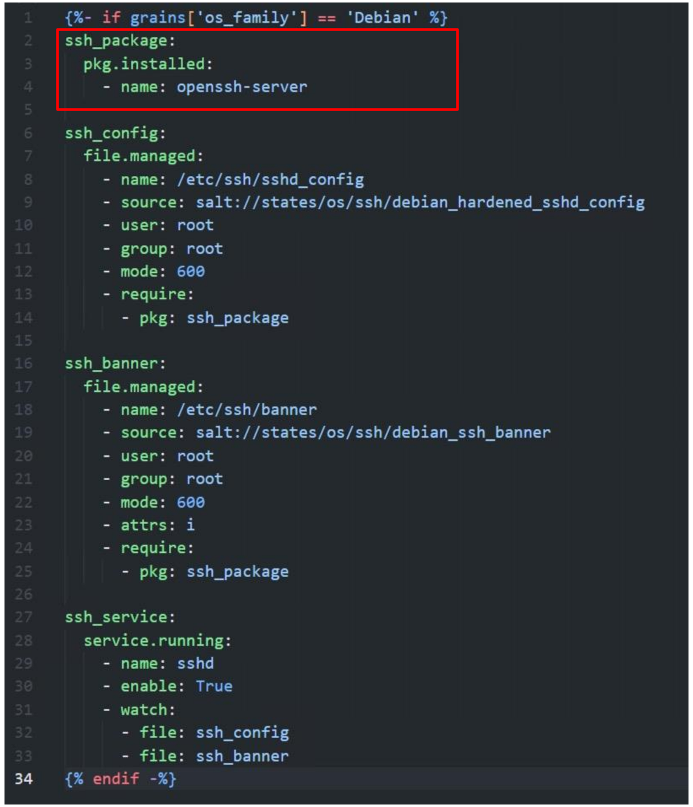
Salt approach to cyber hardening



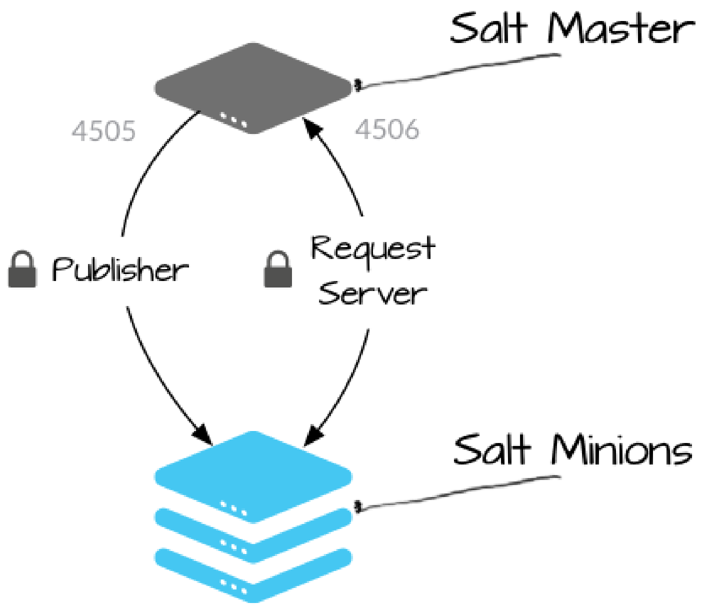
```
{%- if grains['os_family'] == 'Debian' %}
ssh_package:
  pkg.installed:
    - name: openssh-server

ssh_config:
  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://states/os/ssh/debian_hardened_sshd_config
    - user: root
    - group: root
    - mode: 600
    - require:
      - pkg: ssh_package

ssh_banner:
  file.managed:
    - name: /etc/ssh/banner
    - source: salt://states/os/ssh/debian_ssh_banner
    - user: root
    - group: root
    - mode: 600
    - attrs: i
    - require:
      - pkg: ssh_package

ssh_service:
  service.running:
    - name: sshd
    - enable: True
    - watch:
      - file: ssh_config
      - file: ssh_banner
{% endif -%}
```



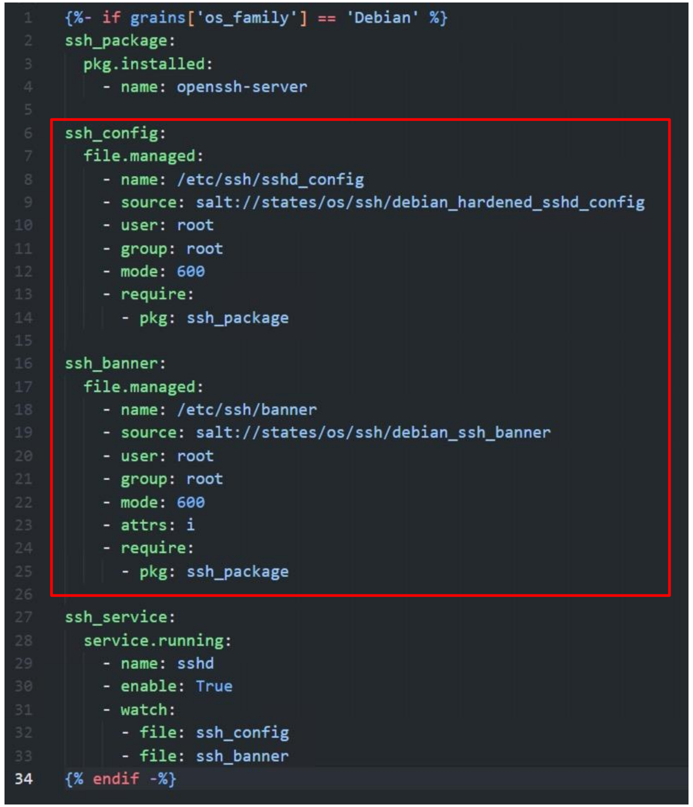
Salt approach to cyber hardening



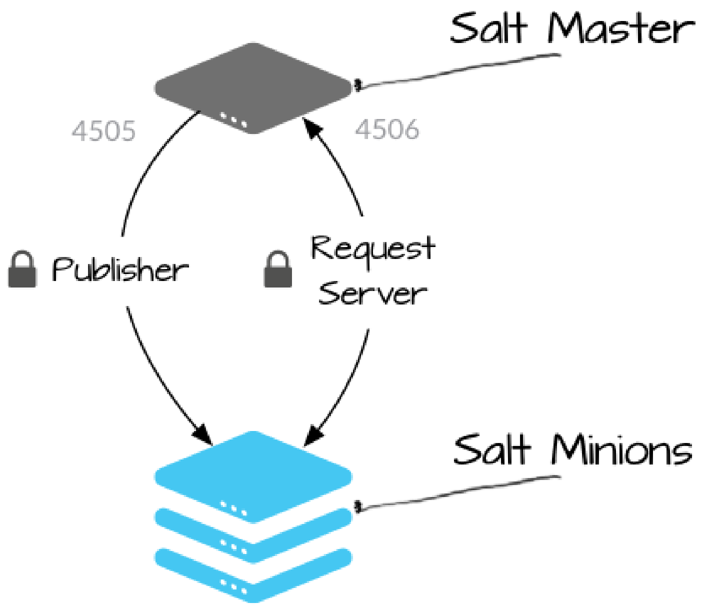
```
{%- if grains['os_family'] == 'Debian' %}
ssh_package:
  pkg.installed:
    - name: openssh-server

ssh_config:
  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://states/os/ssh/debian_hardened_sshd_config
    - user: root
    - group: root
    - mode: 600
    - require:
      - pkg: ssh_package

ssh_banner:
  file.managed:
    - name: /etc/ssh/banner
    - source: salt://states/os/ssh/debian_ssh_banner
    - user: root
    - group: root
    - mode: 600
    - attrs: i
    - require:
      - pkg: ssh_package

ssh_service:
  service.running:
    - name: sshd
    - enable: True
    - watch:
      - file: ssh_config
      - file: ssh_banner
{% endif -%}
```



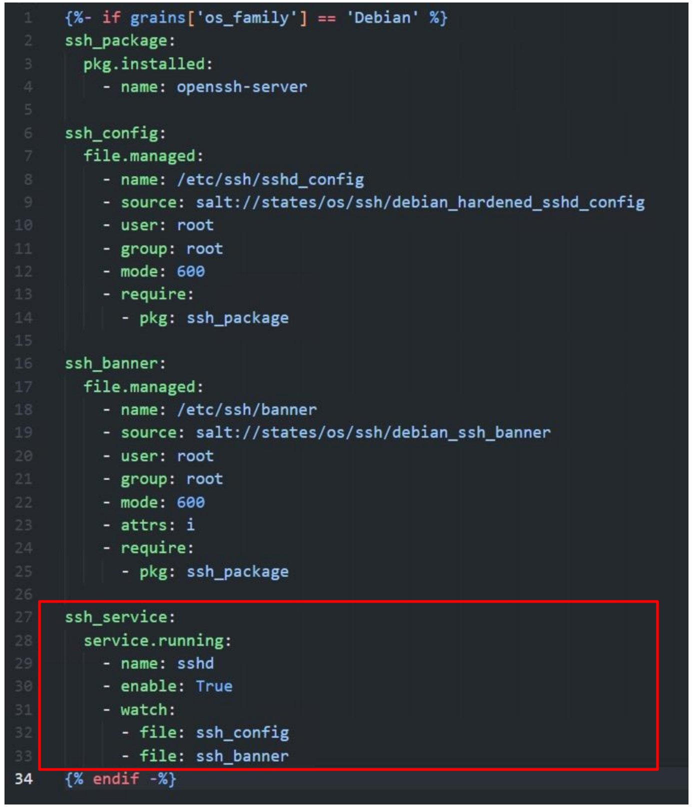
Salt approach to cyber hardening



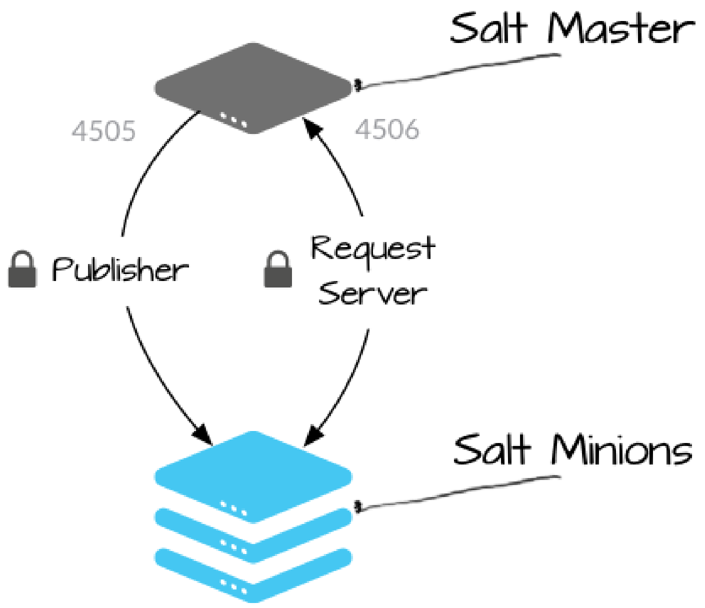
```
{%- if grains['os_family'] == 'Debian' %}
ssh_package:
  pkg.installed:
    - name: openssh-server

ssh_config:
  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://states/os/ssh/debian_hardened_sshd_config
    - user: root
    - group: root
    - mode: 600
    - require:
      - pkg: ssh_package

ssh_banner:
  file.managed:
    - name: /etc/ssh/banner
    - source: salt://states/os/ssh/debian_ssh_banner
    - user: root
    - group: root
    - mode: 600
    - attrs: i
    - require:
      - pkg: ssh_package

ssh_service:
  service.running:
    - name: sshd
    - enable: True
    - watch:
      - file: ssh_config
      - file: ssh_banner
{% endif -%}
```



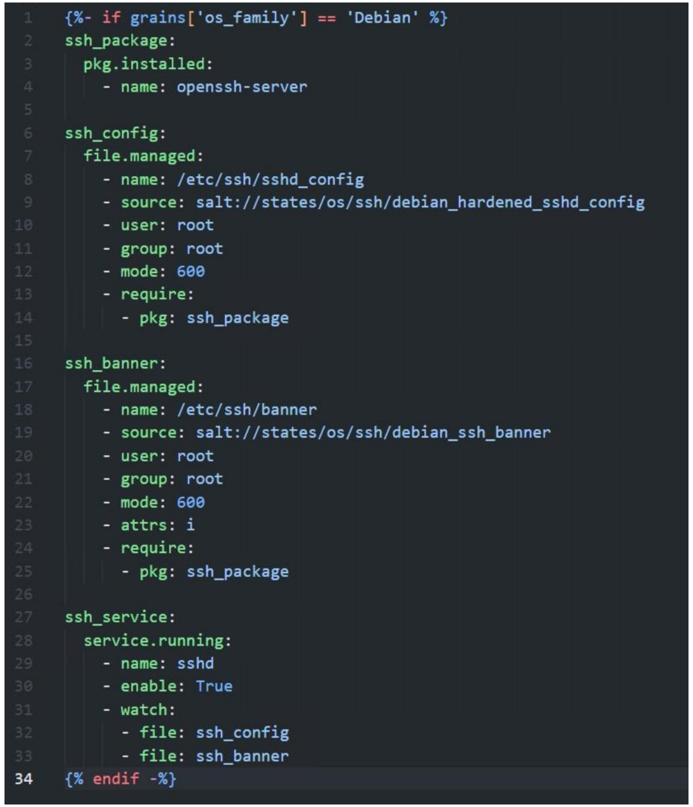
Salt approach to cyber hardening



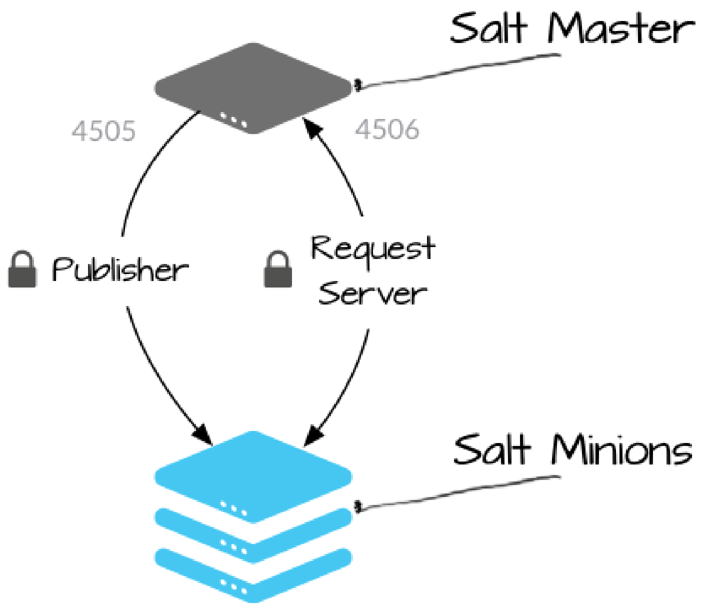
```
{%- if grains['os_family'] == 'Debian' %}
ssh_package:
  pkg.installed:
    - name: openssh-server

ssh_config:
  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://states/os/ssh/debian_hardened_sshd_config
    - user: root
    - group: root
    - mode: 600
    - require:
      - pkg: ssh_package

ssh_banner:
  file.managed:
    - name: /etc/ssh/banner
    - source: salt://states/os/ssh/debian_ssh_banner
    - user: root
    - group: root
    - mode: 600
    - attrs: i
    - require:
      - pkg: ssh_package

ssh_service:
  service.running:
    - name: sshd
    - enable: True
    - watch:
      - file: ssh_config
      - file: ssh_banner
{% endif -%}
```



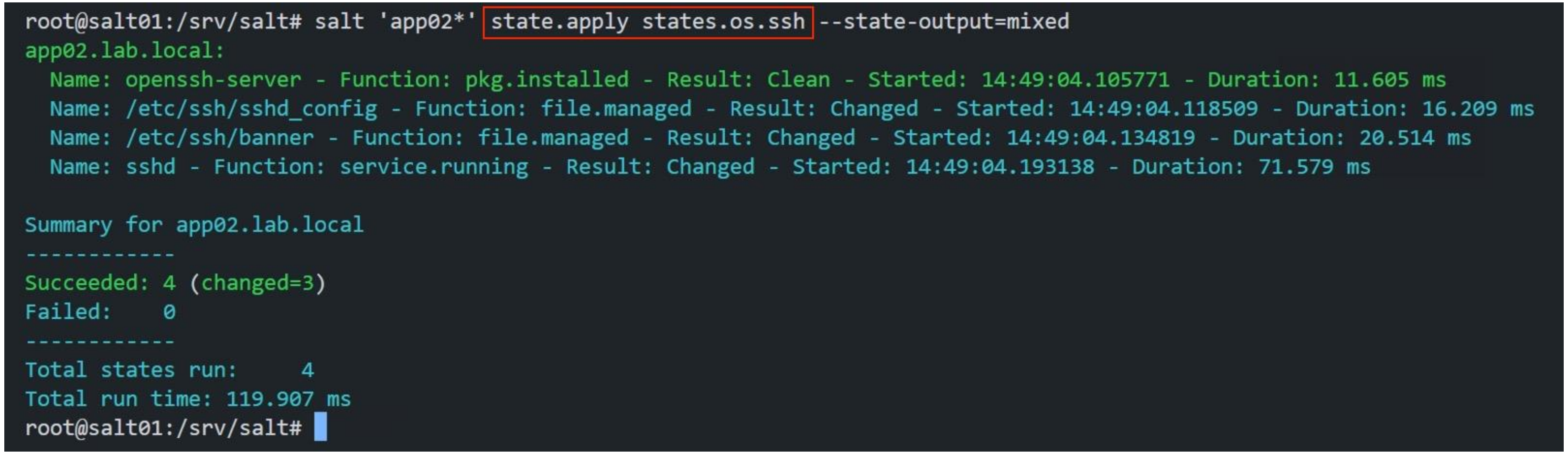
Salt approach to cyber hardening



```
{%- if grains['os_family'] == 'Debian' %}
ssh_package:
  pkg.installed:
    - name: openssh-server

ssh_config:
  file.managed:
    - name: /etc/ssh/sshd_config
    - source: salt://states/os/ssh/debian_hardened_sshd_config
    - user: root
    - group: root
    - mode: 600
    - require:
      - pkg: ssh_package

ssh_banner:
  file.managed:
    - name: /etc/ssh/banner
    - source: salt://states/os/ssh/debian_ssh_banner
    - user: root
    - group: root
    - mode: 600
    - attrs: i
    - require:
      - pkg: ssh_package

ssh_service:
  service.running:
    - name: sshd
    - enable: True
    - watch:
      - file: ssh_config
      - file: ssh_banner
{% endif -%}
```



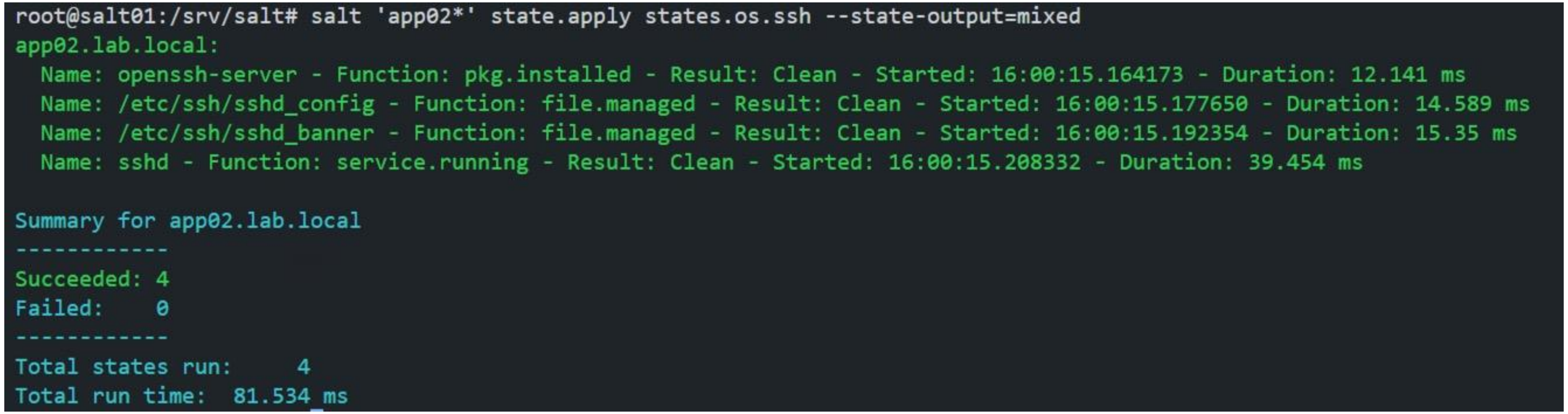
Applying Salt SSH state



```
root@salt01:/srv/salt# salt 'app02*' state.apply states.os.ssh --state-output=mixed
app02.lab.local:
  Name: openssh-server - Function: pkg.installed - Result: Clean - Started: 14:49:04.105771 - Duration: 11.605 ms
  Name: /etc/ssh/sshd_config - Function: file.managed - Result: Changed - Started: 14:49:04.118509 - Duration: 16.209 ms
  Name: /etc/ssh/banner - Function: file.managed - Result: Changed - Started: 14:49:04.134819 - Duration: 20.514 ms
  Name: sshd - Function: service.running - Result: Changed - Started: 14:49:04.193138 - Duration: 71.579 ms

Summary for app02.lab.local
------------
Succeeded: 4 (changed=3)
Failed:    0
------------
Total states run:     4
Total run time: 119.907 ms
root@salt01:/srv/salt#
```

** "--state-output=mixed" flag is not required but used for presentation purposes*

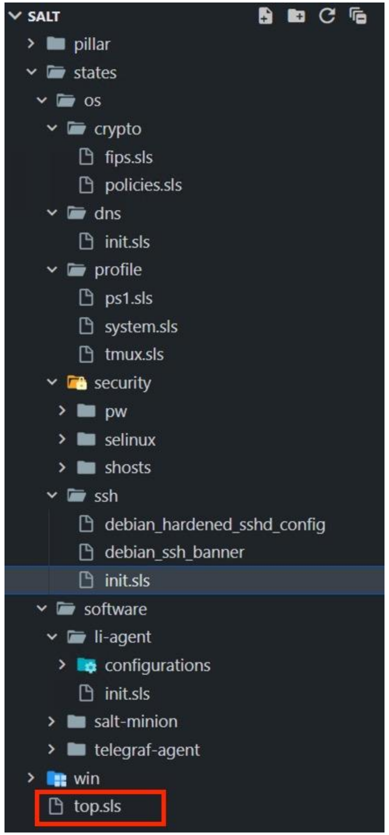
Applying Salt SSH state

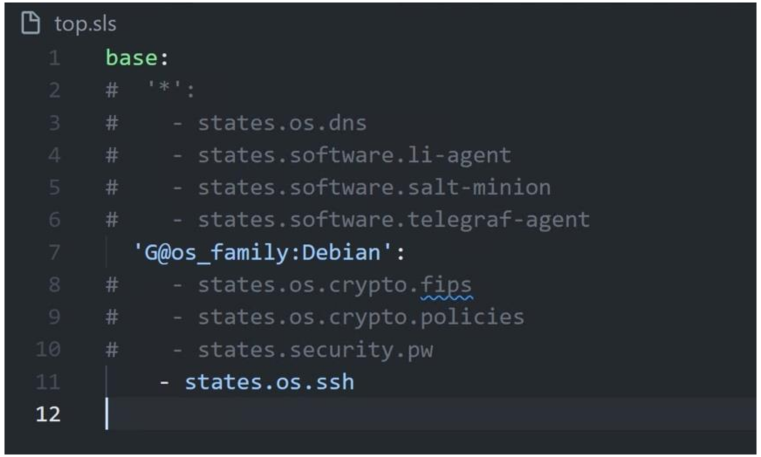


```
root@salt01:/srv/salt# salt 'app02*' state.apply states.os.ssh --state-output=mixed
app02.lab.local:
  Name: openssh-server - Function: pkg.installed - Result: Clean - Started: 16:00:15.164173 - Duration: 12.141 ms
  Name: /etc/ssh/sshd_config - Function: file.managed - Result: Clean - Started: 16:00:15.177650 - Duration: 14.589 ms
  Name: /etc/ssh/sshd_banner - Function: file.managed - Result: Clean - Started: 16:00:15.192354 - Duration: 15.35 ms
  Name: sshd - Function: service.running - Result: Clean - Started: 16:00:15.208332 - Duration: 39.454 ms

Summary for app02.lab.local
------------
Succeeded: 4
Failed:    0
------------
Total states run:     4
Total run time:  81.534 ms
```

** "--state-output=mixed" flag is not required but used for presentation purposes*

Salt State Tree

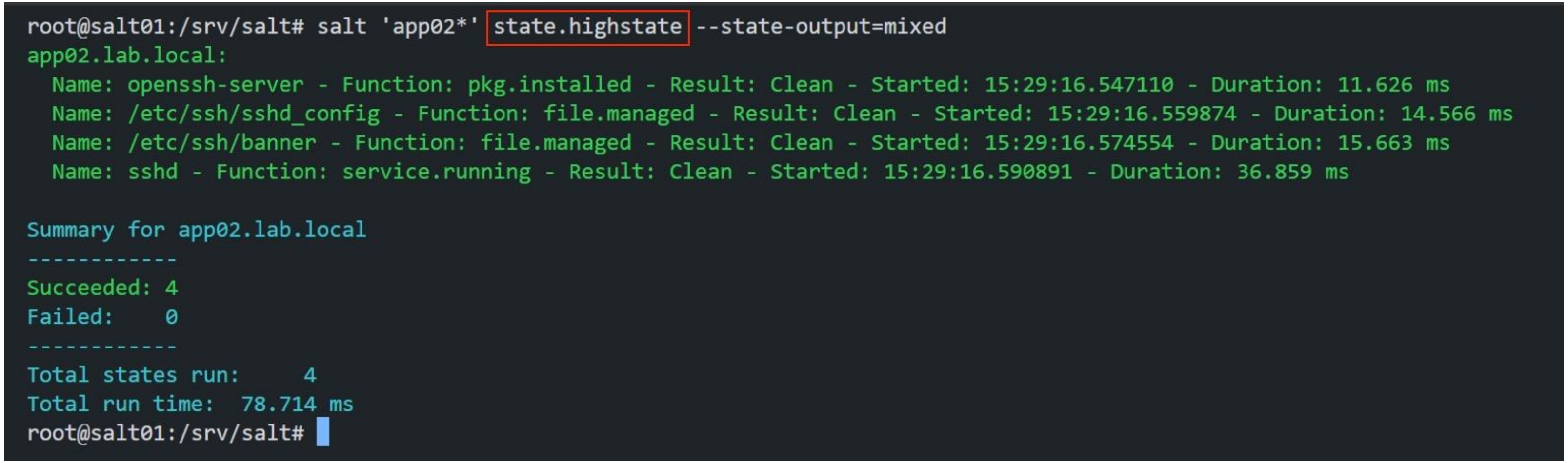




```
top.sls
base:
#  '*':
#     - states.os.dns
#     - states.software.li-agent
#     - states.software.salt-minion
#     - states.software.telegraf-agent
  'G@os_family:Debian':
#     - states.os.crypto.fips
#     - states.os.crypto.policies
#     - states.security.pw
    - states.os.ssh

```

** Example top.sls file, placed in the root directory of Salt file server*

Applying Salt Highstate

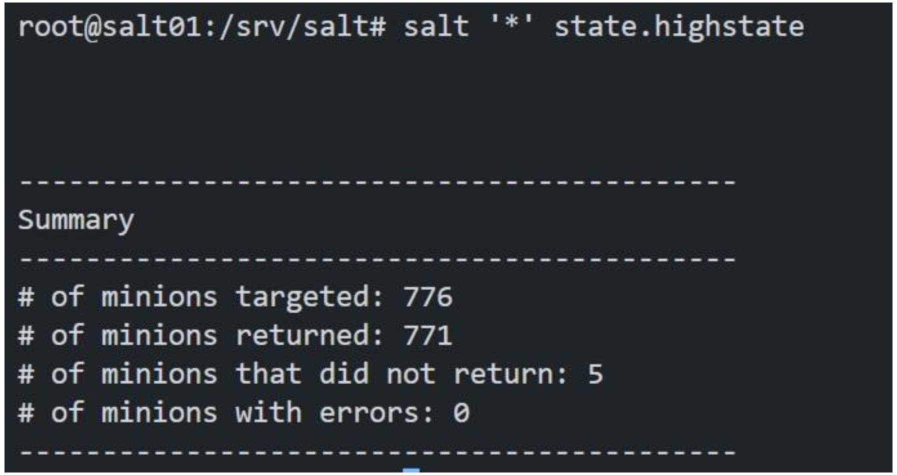


```
root@salt01:/srv/salt# salt 'app02*' state.highstate --state-output=mixed
app02.lab.local:
  Name: openssh-server - Function: pkg.installed - Result: Clean - Started: 15:29:16.547110 - Duration: 11.626 ms
  Name: /etc/ssh/sshd_config - Function: file.managed - Result: Clean - Started: 15:29:16.559874 - Duration: 14.566 ms
  Name: /etc/ssh/banner - Function: file.managed - Result: Clean - Started: 15:29:16.574554 - Duration: 15.663 ms
  Name: sshd - Function: service.running - Result: Clean - Started: 15:29:16.590891 - Duration: 36.859 ms

Summary for app02.lab.local
------------
Succeeded: 4
Failed:    0
------------
Total states run:     4
Total run time:  78.714 ms
root@salt01:/srv/salt#
```

** "--state-output=mixed" flag is not required but used for presentation purposes*

** "salt <minion id> state.apply" without specifying a State file achieves the same result*

Applying Salt Highstate



```
root@salt01:/srv/salt# salt '*' state.highstate



-------------------------------------------
Summary
-------------------------------------------
# of minions targeted: 776
# of minions returned: 771
# of minions that did not return: 5
# of minions with errors: 0
-------------------------------------------
```

* “salt <minion id> state.apply” without specifying a State file achieves the same result

Operations - Automating agent deployments

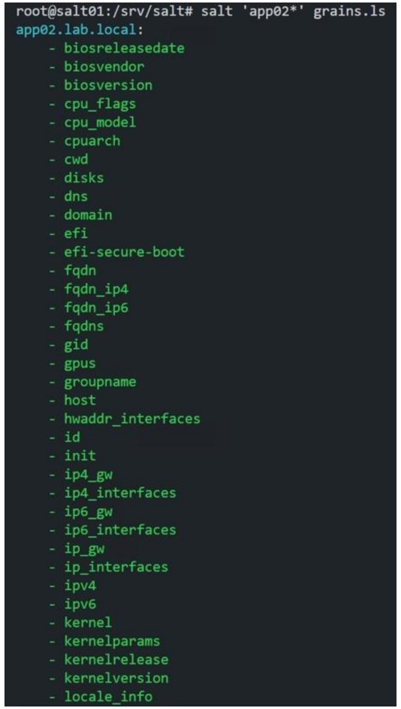
Goal:

- Automatically install and configure software agents (log collectors, EDR/XDR tools, etc.)

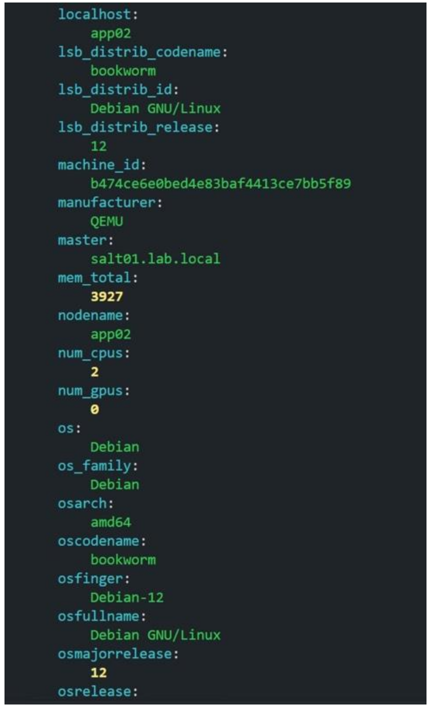
Challenges:

- Quickly installing software across fleet
- Keeping software installs up-to-date
- Setting system or environment-specific configuration files

Salt Grains



```
root@salt01:/srv/salt# salt 'app02*' grains.ls
app02.lab.local:
    - biosreleasedate
    - biosvendor
    - biosversion
    - cpu_flags
    - cpu_model
    - cpuarch
    - cwd
    - disks
    - dns
    - domain
    - efi
    - efi-secure-boot
    - fqdn
    - fqdn_ip4
    - fqdn_ip6
    - fqdns
    - gid
    - gpus
    - groupname
    - host
    - hwaddr_interfaces
    - id
    - init
    - ip4_gw
    - ip4_interfaces
    - ip6_gw
    - ip6_interfaces
    - ip_gw
    - ip_interfaces
    - ipv4
    - ipv6
    - kernel
    - kernelparams
    - kernelrelease
    - kernelversion
    - locale_info
```

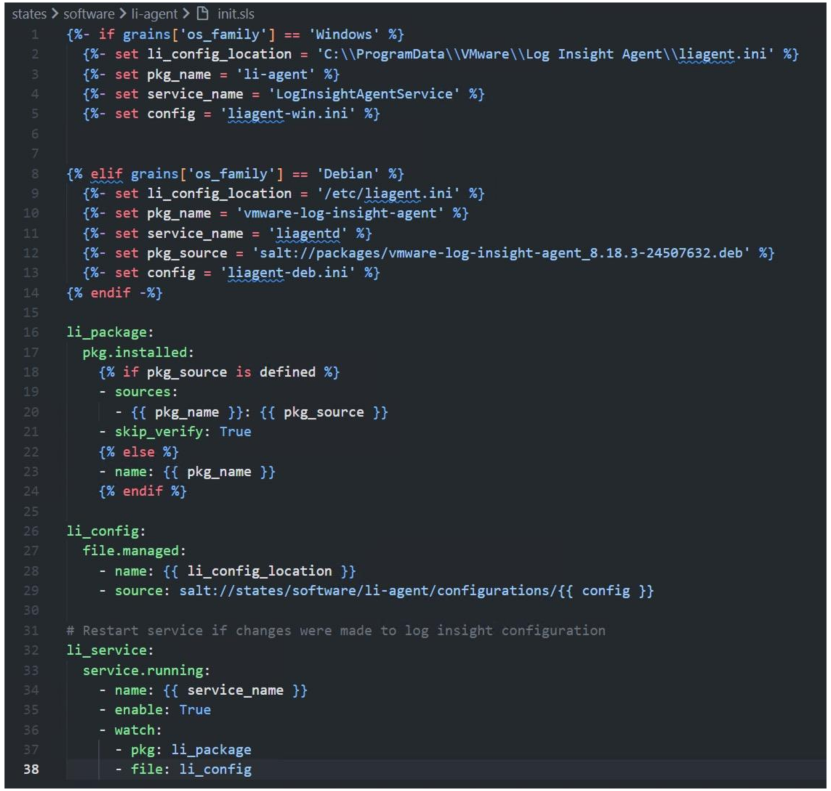


```
    localhost:
        app02
    lsb_distrib_codename:
        bookworm
    lsb_distrib_id:
        Debian GNU/Linux
    lsb_distrib_release:
        12
    machine_id:
        b474ce6e0bed4e83baf4413ce7bb5f89
    manufacturer:
        QEMU
    master:
        salt01.lab.local
    mem_total:
        3927
    nodename:
        app02
    num_cpus:
        2
    num_gpus:
        0
    os:
        Debian
    os_family:
        Debian
    osarch:
        amd64
    oscodename:
        bookworm
    osfinger:
        Debian-12
    osfullname:
        Debian GNU/Linux
    osmajorrelease:
        12
    osrelease:
```

- The grains interface allows you to derive information about a minion
- Relatively static information
- Can be used as variables in Salt States

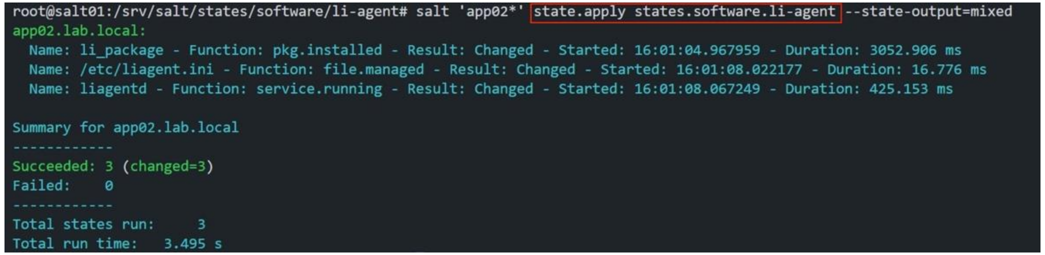
**(right snippet) subset of “salt <minion id> grains.items”*

VMware Log Insight Agent State



states > software > li-agent > init.sls

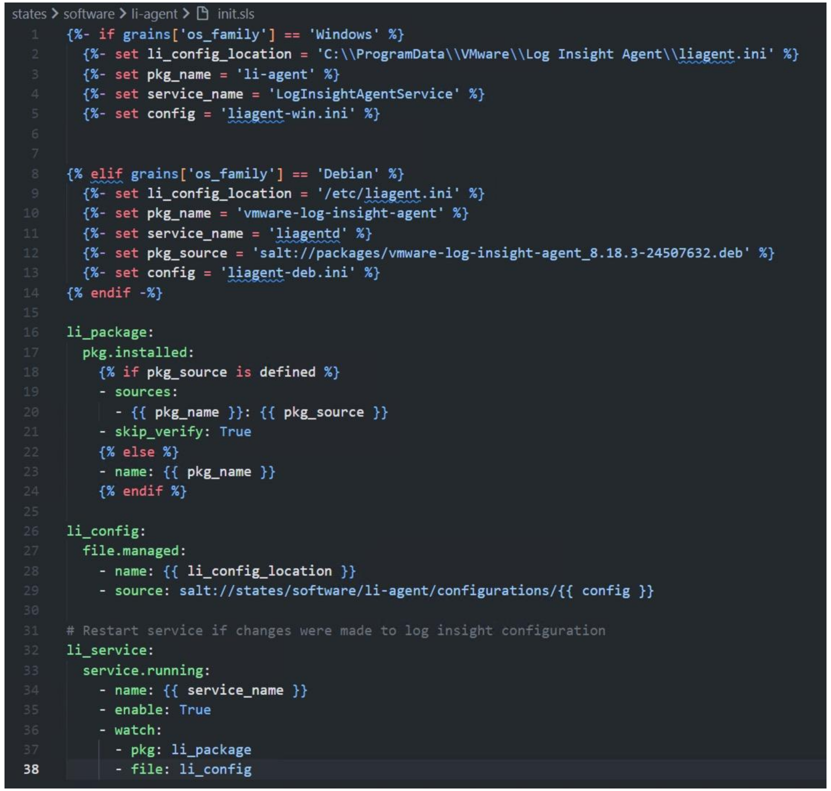
```
{%- if grains['os_family'] == 'Windows' %}
  {%- set li_config_location = 'C:\\ProgramData\\VMware\\Log Insight Agent\\liagent.ini' %}
  {%- set pkg_name = 'li-agent' %}
  {%- set service_name = 'LogInsightAgentService' %}
  {%- set config = 'liagent-win.ini' %}


{% elif grains['os_family'] == 'Debian' %}
  {%- set li_config_location = '/etc/liagent.ini' %}
  {%- set pkg_name = 'vmware-log-insight-agent' %}
  {%- set service_name = 'liagentd' %}
  {%- set pkg_source = 'salt://packages/vmware-log-insight-agent_8.18.3-24507632.deb' %}
  {%- set config = 'liagent-deb.ini' %}
{% endif -%}

li_package:
  pkg.installed:
    {% if pkg_source is defined %}
    - sources:
      - {{ pkg_name }}: {{ pkg_source }}
    - skip_verify: True
    {% else %}
    - name: {{ pkg_name }}
    {% endif %}

li_config:
  file.managed:
    - name: {{ li_config_location }}
    - source: salt://states/software/li-agent/configurations/{{ config }}

# Restart service if changes were made to log insight configuration
li_service:
  service.running:
    - name: {{ service_name }}
    - enable: True
    - watch:
      - pkg: li_package
      - file: li_config
```



```
root@salt01:/srv/salt/states/software/li-agent# salt 'app02*' state.apply states.software.li-agent --state-output=mixed
app02.lab.local:
  Name: li_package - Function: pkg.installed - Result: Changed - Started: 16:01:04.967959 - Duration: 3052.906 ms
  Name: /etc/liagent.ini - Function: file.managed - Result: Changed - Started: 16:01:08.022177 - Duration: 16.776 ms
  Name: liagentd - Function: service.running - Result: Changed - Started: 16:01:08.067249 - Duration: 425.153 ms

Summary for app02.lab.local
------------
Succeeded: 3 (changed=3)
Failed:    0
------------
Total states run:     3
Total run time:   3.495 s
```

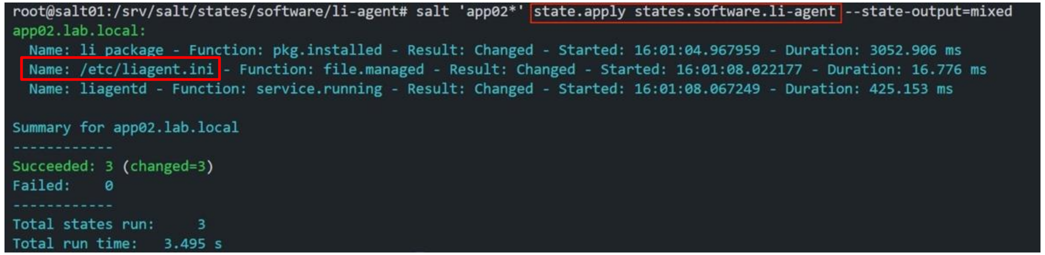
** "--state-output=mixed" flag is not required but used for presentation purposes*

VMware Log Insight Agent State



states > software > li-agent > init.sls

```
{%- if grains['os_family'] == 'Windows' %}
  {%- set li_config_location = 'C:\\ProgramData\\VMware\\Log Insight Agent\\liagent.ini' %}
  {%- set pkg_name = 'li-agent' %}
  {%- set service_name = 'LogInsightAgentService' %}
  {%- set config = 'liagent-win.ini' %}


{% elif grains['os_family'] == 'Debian' %}
  {%- set li_config_location = '/etc/liagent.ini' %}
  {%- set pkg_name = 'vmware-log-insight-agent' %}
  {%- set service_name = 'liagentd' %}
  {%- set pkg_source = 'salt://packages/vmware-log-insight-agent_8.18.3-24507632.deb' %}
  {%- set config = 'liagent-deb.ini' %}
{% endif -%}

li_package:
  pkg.installed:
    {% if pkg_source is defined %}
    - sources:
      - {{ pkg_name }}: {{ pkg_source }}
    - skip_verify: True
    {% else %}
    - name: {{ pkg_name }}
    {% endif %}

li_config:
  file.managed:
    - name: {{ li_config_location }}
    - source: salt://states/software/li-agent/configurations/{{ config }}

# Restart service if changes were made to log insight configuration
li_service:
  service.running:
    - name: {{ service_name }}
    - enable: True
    - watch:
      - pkg: li_package
      - file: li_config
```



```
root@salt01:/srv/salt/states/software/li-agent# salt 'app02*' state.apply states.software.li-agent --state-output=mixed
app02.lab.local:
  Name: li package - Function: pkg.installed - Result: Changed - Started: 16:01:04.967959 - Duration: 3052.906 ms
  Name: /etc/liagent.ini - Function: file.managed - Result: Changed - Started: 16:01:08.022177 - Duration: 16.776 ms
  Name: liagentd - Function: service.running - Result: Changed - Started: 16:01:08.067249 - Duration: 425.153 ms

Summary for app02.lab.local
------------
Succeeded: 3 (changed=3)
Failed:    0
------------
Total states run:     3
Total run time:   3.495 s
```

** “--state-output=mixed” flag is not required but used for presentation purposes*

Automating compliance data collection

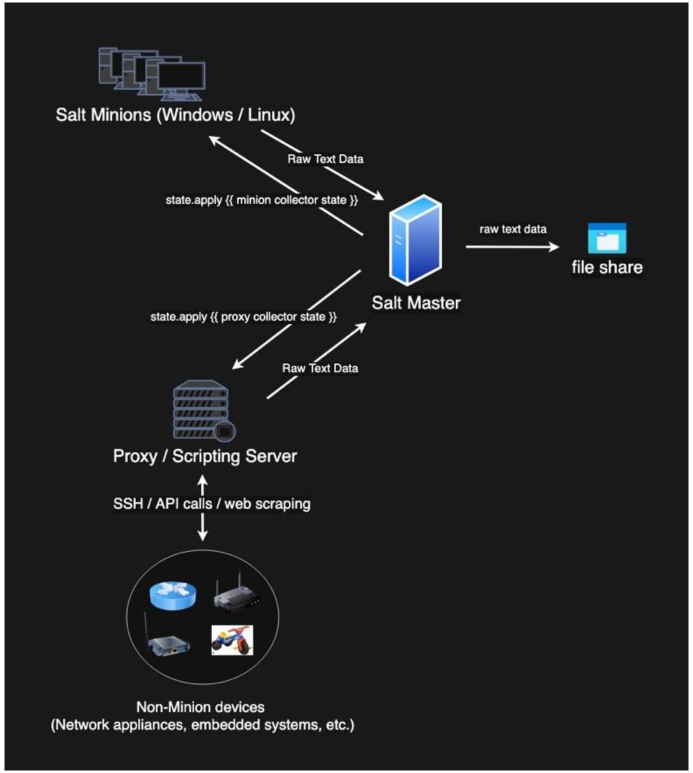
Goal:

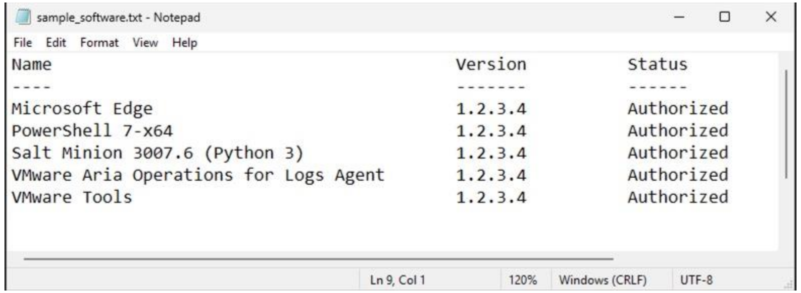
- Automatically collect and store system configuration compliance data:
    - Network Configurations
    - Installed Software
    - Authorized Users and Groups
    - Password Policy

Challenges:

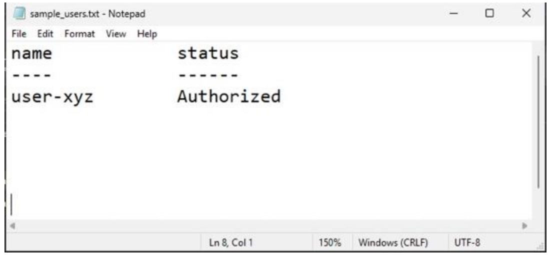
- Reliable mechanism to collect data on a regular cadence
- Output in convenient data structure for simple archival
- Support collection from hardened and non-traditional assets (appliances, OT devices)

Compliance data collection overview


Salt Minions (Windows / Linux)
Raw Text Data
state.apply {{ minion collector state }}
raw text data
file share
Salt Master
state.apply {{ proxy collector state }}
Raw Text Data
Proxy / Scripting Server
SSH / API calls / web scraping
Non-Minion devices
(Network appliances, embedded systems, etc.)



```
Name                                    Version         Status
----                                    -------         ------
Microsoft Edge                          1.2.3.4         Authorized
PowerShell 7-x64                        1.2.3.4         Authorized
Salt Minion 3007.6 (Python 3)           1.2.3.4         Authorized
VMware Aria Operations for Logs Agent   1.2.3.4         Authorized
VMware Tools                            1.2.3.4         Authorized
```



```
name                status
----                ------
user-xyz            Authorized
```

**Only sample data is shown for presentation purposes*

Summary

Use Cases:

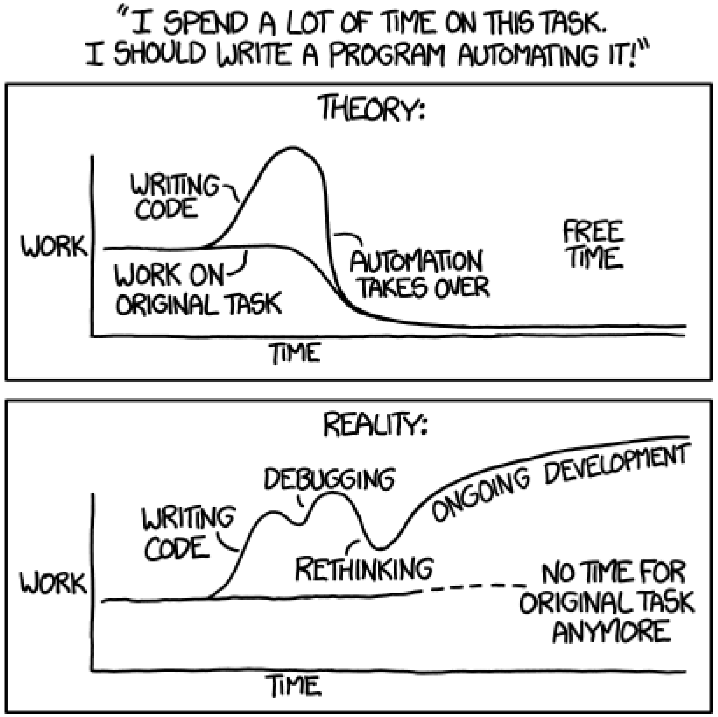
- Security (System Hardening)
- Operations (Agent Deployment)
- Compliance (Data Collection)

Where to next?

- Talk to your teams!
- Think ”outside the box” for automation opportunities

Resources:

- Additional resources available at https://github.com/evynprice/explore2025-salt



TVA

TENNESSEE
VALLEY
AUTHORITY