https://unsplash.com/photos/6kfQvW5eync
Photo by Wilhelm Gunkel on Unsplash

System Administration

Week 10, Segment 1
Configuration Management, Part I

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

The entropy of an isolated system never decreases.

A static system is a useless system. A useful system is… being used.

- data is processed; files are created, modified, removed
- software is added, upgraded, removed
- systems are created, copied, decommissioned
- instances / containers are even more short-lived, coming into existence and disappearing again as needed

Single Systems are Fragile

Individual systems created and configured by hand are fragile. Our processes need to be repeatable, automated, reliable.

Recall previous lectures:

- OS installation
- package management
- multi-user basics
- recovery / restores

Evolution of Configuration Management

“I set up a server over here to do X. Replicate that setup on all the others.”

```
server1# scp -r /opt/service root@server2:/opt
server1# scp  /etc/service.conf root@server2:/etc/
server1# ssh root@server2 “/etc/rc.d/service start”
```

Evolution of Configuration Management

“I set up a server over here to do X. Replicate that setup on all the others.”

```
server1# rsync -e ssh -avz /opt/service/. root@server2:/opt/service/.
server1# rsync -e ssh -avz /etc/. root@server2:/etc/.
```

“/etc? Why, what about it?”

Variable vs. Static & Shareable vs. Non-Shareable Data

- Variable: data expected to be modified during routine operations
- Static: data not expected to change during runtime

- Shareable: data that remains the same across multiple (instances of) hosts
- Non-shareable: data that is unique to a specific (instance of a) system

	shareable	**non-shareable**
static	/usr /opt	/boot /etc
variable	/var/data /home	/var/run /var/log

Evolution of Configuration Management

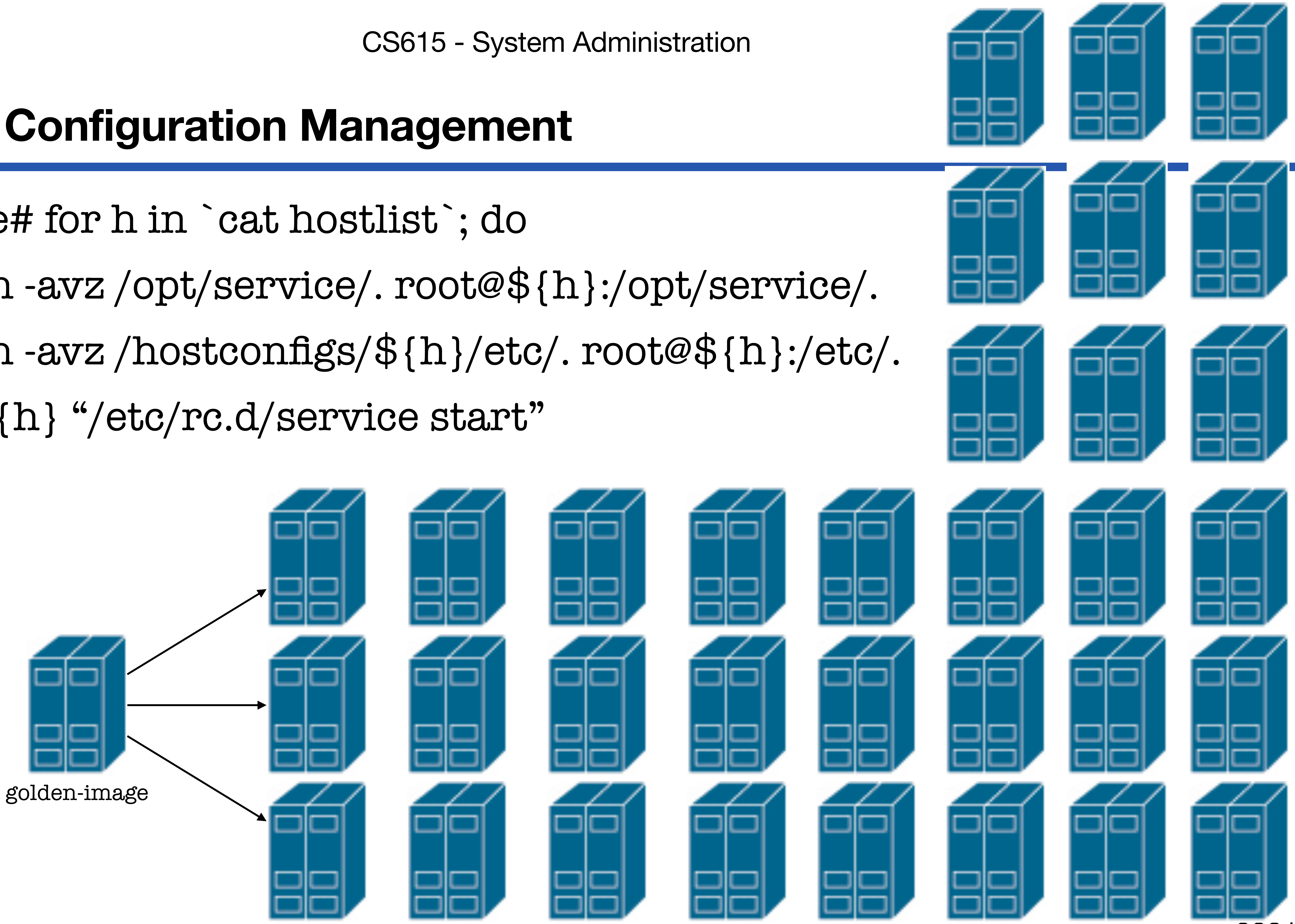
“I set up a server over here to do X. Replicate that setup on all the others.”

```
server1# rsync -e ssh -avz /opt/service/. root@server2:/opt/service/.
server1# rsync -e ssh -avz /etc/. root@server2:/etc/.
```

“/etc? Why, what about it?”

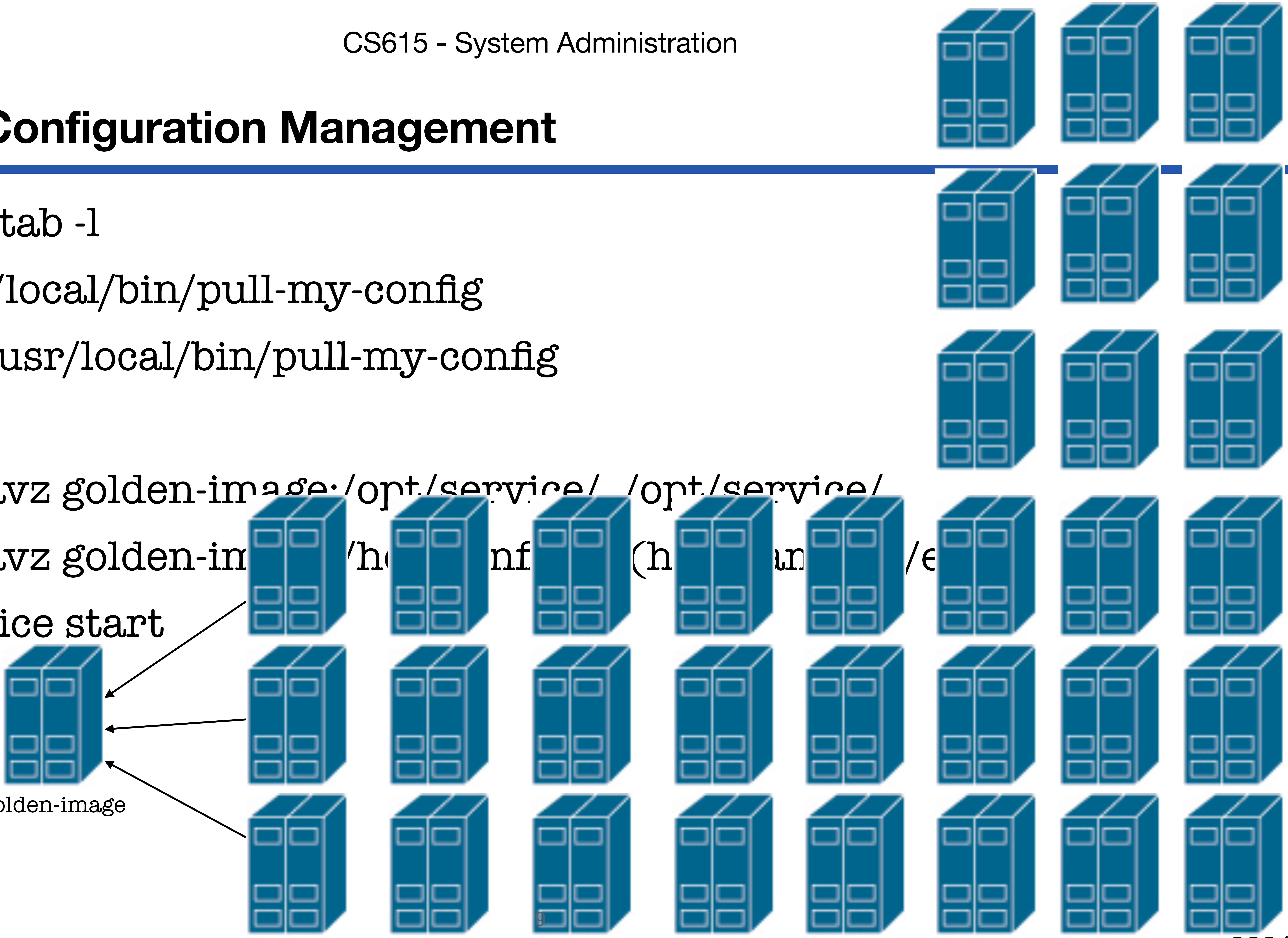
Evolution of Configuration Management

```
golden-image# for h in `cat hostlist`; do
> rsync -e ssh -avz /opt/service/. root@${h}:/opt/service/.
> rsync -e ssh -avz /hostconfigs/${h}/etc/. root@${h}:/etc/.
> ssh root@${h} "/etc/rc.d/service start"
> done
```



Evolution of Configuration Management

```
server1# crontab -l
0 * * * * /usr/local/bin/pull-my-config
server1# cat /usr/local/bin/pull-my-config
#! /bin/sh
rsync -e ssh -avz golden-image:/opt/service/ /opt/service/
rsync -e ssh -avz golden-im    /h     nf     (h     an     /e
/etc/rc.d/service start
server1#
```



Evolution of Configuration Management

```
server1# crontab -l
0 * * * * /usr/local/bin/pull-my-config
server1# cat /usr/local/bin/pull-my-config
#! /bin/sh
sleep $(( ($(date +%s) + $$) % 1800 ))
rsync -e ssh -avz golden-image:/opt/service/. /opt/service/.
rsync -e ssh -avz golden-image:/hostconfigs/$(hostname)/. /etc/.
/etc/rc.d/service start
server1#
```

try to avoid “thundering herd” problem.

Evolution of Configuration Management

```
golden-image# echo “Last updated on: $(date)“ > /hostconfig/server1/etc/motd
golden-image# date +%s > /usr/local/share/htdocs/server1/latest


server1# cat /usr/local/bin/pull-my-config
#! /bin/sh
last=$(cat /etc/last-pull)
latest=$(curl https://golden-image/$(hostname)/latest)
if [ ${latest} -gt ${last} ]; then
        sync-data
fi
date +%s > /etc/last-pull
server1#
```

Evolution of Configuration Management

```
golden-image# sudo rpm -Uvh https://yum.puppet.com/puppet6-release-
el-7.noarch.rpm
```

several hours of reading ~~the docs~~ various StackOverflow answers

```
golden-image# yum install puppetserver
```

several hours of ~~cursing Java~~ chasing dependencies

```
server1# sudo yum install puppet-agent
server1# puppet ssl bootstrap
```

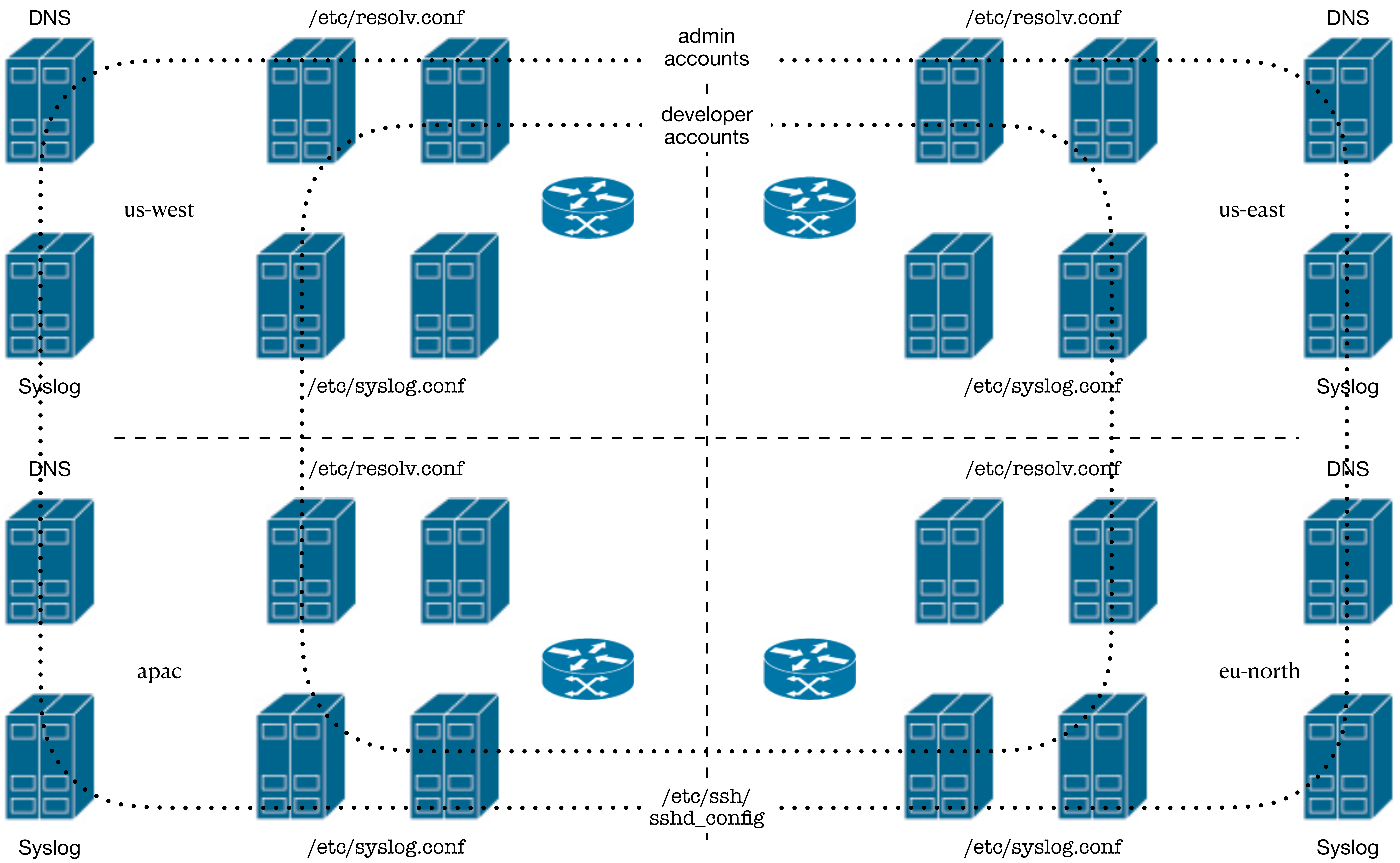
Discover something your CM system can't do and repeat…

Evolution of Configuration Management

"I set up a server over here to do X. Replicate that setup on all the others."

```
server1# rsync -e ssh -avz /opt/service/. root@server2:/opt/service/.
server1# rsync -e ssh -avz /etc/. root@server2:/etc/.
```

"/etc? Why, what about it?"


DNS
/etc/resolv.conf
admin accounts
/etc/resolv.conf
DNS
developer accounts
us-west
us-east
Syslog
/etc/syslog.conf
/etc/syslog.conf
Syslog
DNS
/etc/resolv.conf
/etc/resolv.conf
DNS
apac
eu-north
/etc/ssh/ sshd_config
Syslog
/etc/syslog.conf
/etc/syslog.conf
Syslog

Base configuration vs. service definition

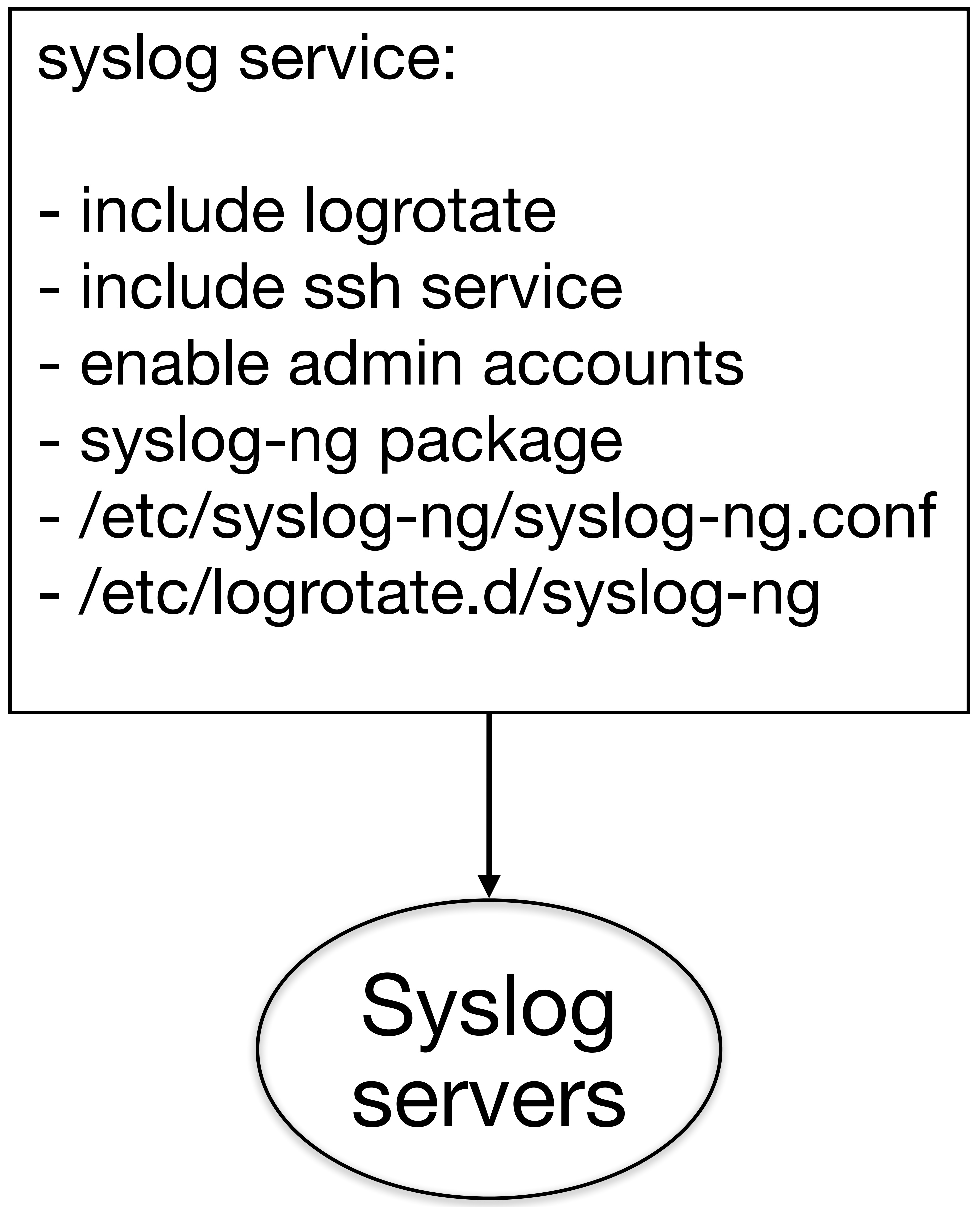
Your servers have unique, yet predictable properties that vary based on workload placement, specific purpose. E.g.,

- network configuration
- critical services such as DNS, NTP, or Syslog
- minimum OS / software version
- user management
- common service configuration (e.g., `sshd(8)`)
- ...

Base configuration vs. service definition

Different sets of servers have shared properties. For example, consider an HTTP server:

- minimum server software
- appropriate TLS specification
- shared TLS certificate and key
- database configuration
- static content (HTML / JS / CSS files)
- …



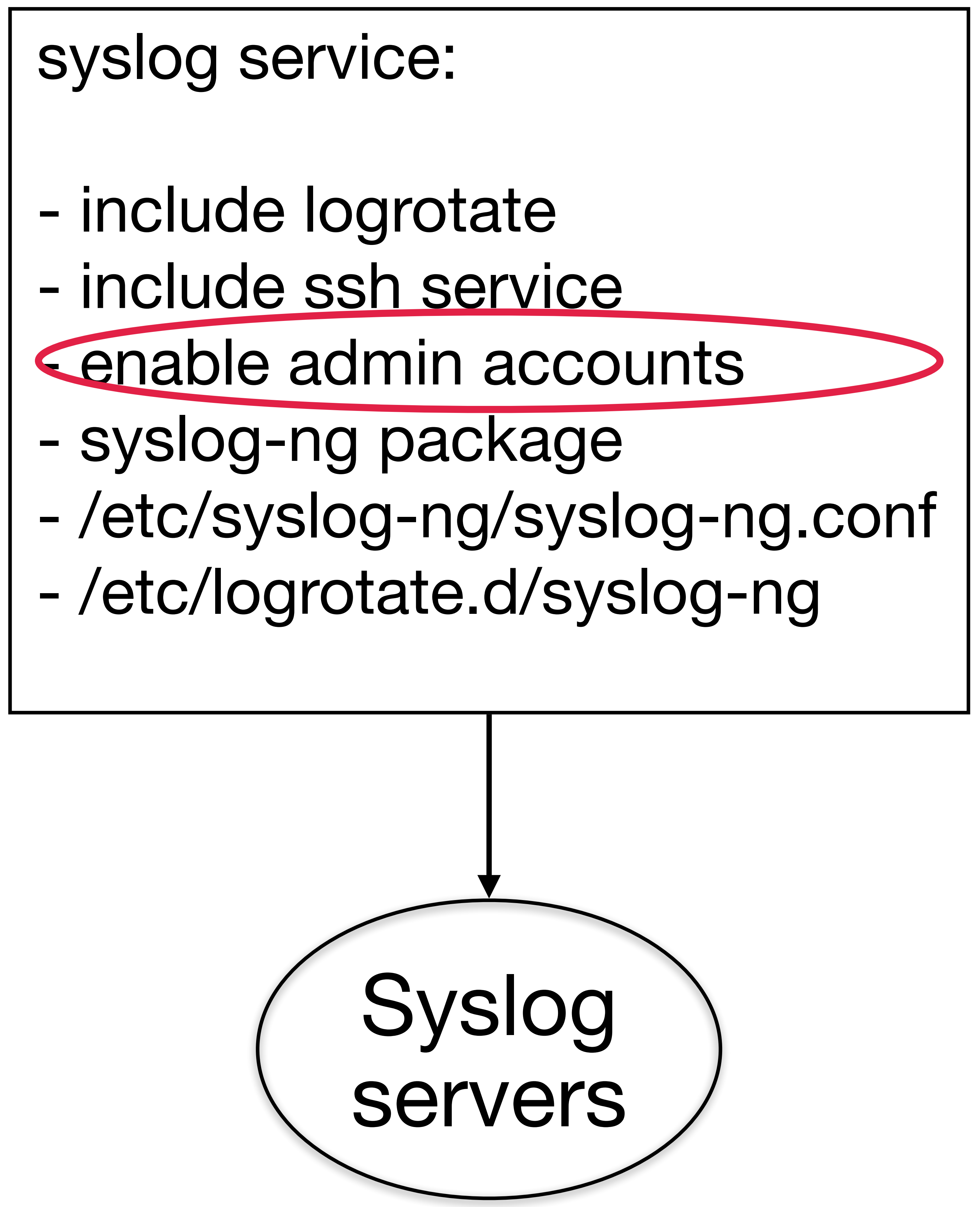
syslog service:

- include logrotate
- include ssh service
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng

```
class syslog {
   include cron
   include logrotate
   package {
       ’syslogng’ :
       ensure => latest,
       require => Service[’syslogng’]; }

   service {
       ’syslogng’ :
       ensure => running,
       enable => true; }

   file {
       ’/etc/syslogng/syslogng.conf’:
           ensure  => file,
           source   => ’puppet:///syslog/syslogng.conf’,
           mode     => ’0644’,
           owner    => ’root’,
           group    => ’root’,
           require  => Package[’syslog-ng’],
           notify    => Service[’syslog-ng’];
       ’/etc/logrotate.d/syslog-ng’:
           ensure  => file,
           source   => ’puppet:///syslog/logrotate-syslogng’,
           mode     => ’0644’,
           owner    => ’root’,
           group    => ’root’,
           require => Package[’logrotate’];
       }
   }
```

puppet

https://puppet.com/



syslog service:

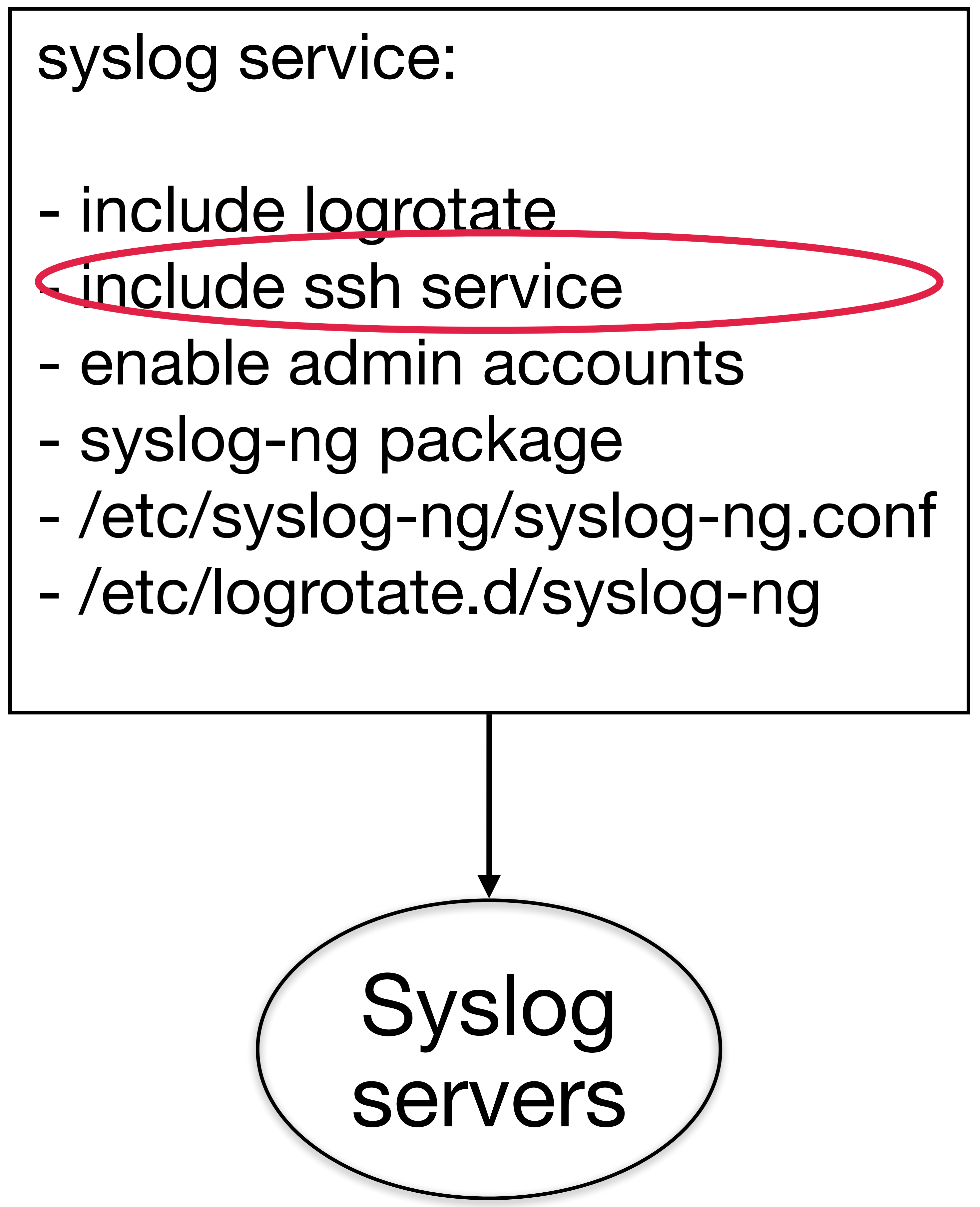
- include logrotate
- include ssh service
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng

```
package "ldap-utils" do
  action :upgrade
end

template "/etc/ldap.conf" do
  source “ldap.conf.erb"
  mode   00644
  owner “root"
  group “root"
end

%w{ account auth password session }.each do |pam|
  cookbook_file "/etc/pam.d/common-#{pam}" do
    source “common-#{pam}”
    mode   00644
    owner “root"
    group “root"
    notifies :restart, resources(:service => "ssh"), :delayed
  end
end
```

https://www.chef.io/

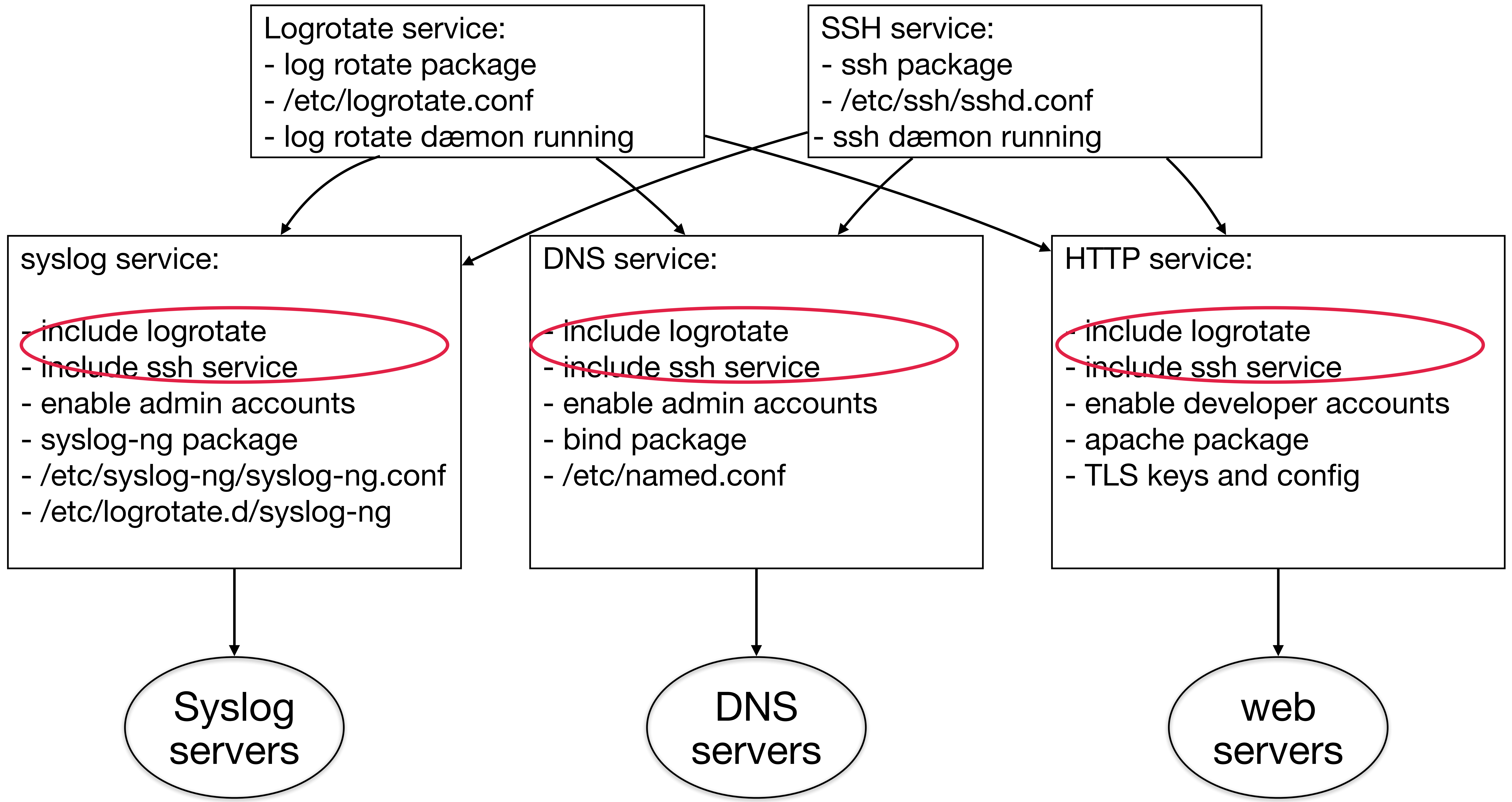
CFEngine

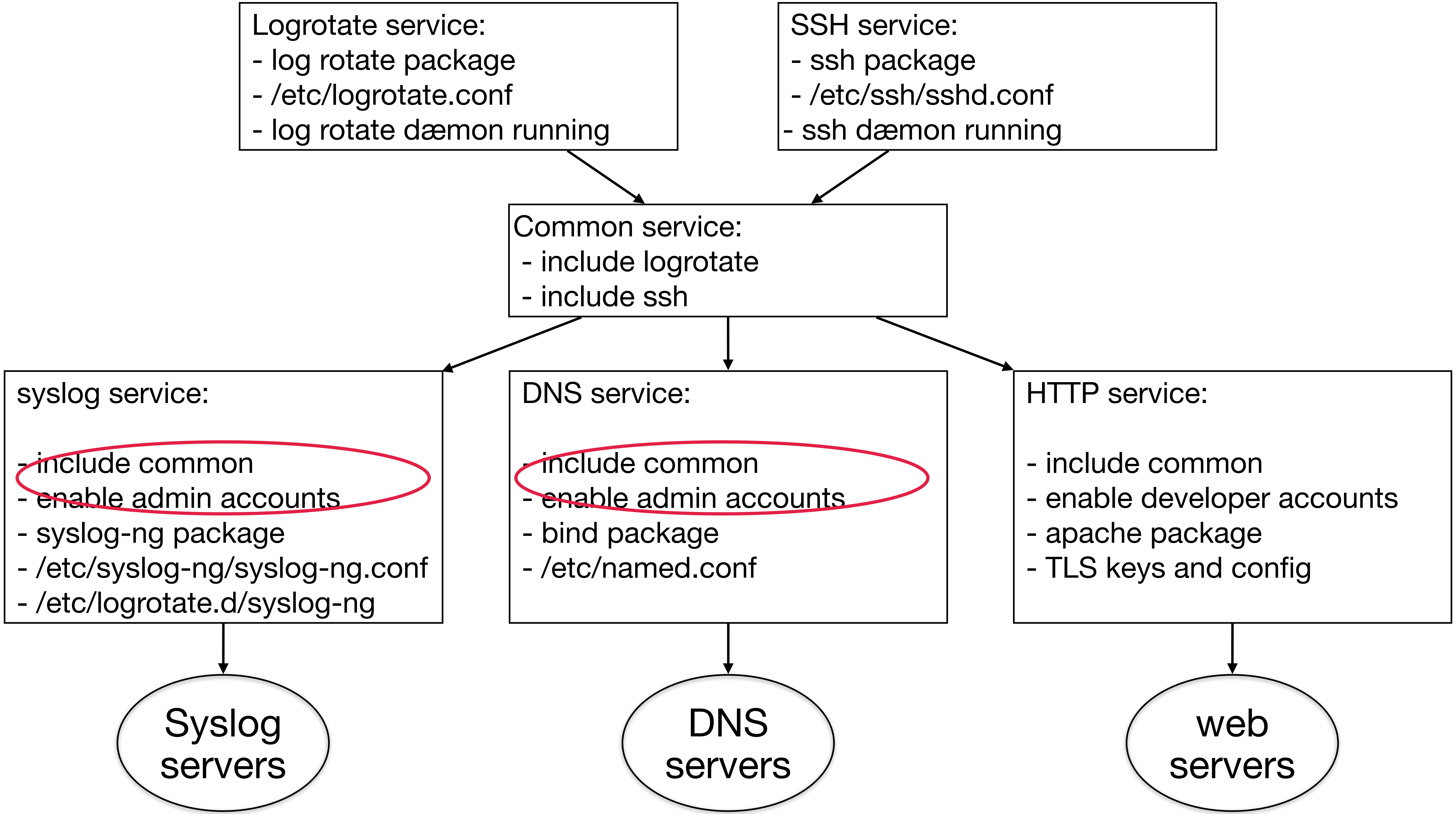
https://cfengine.com/

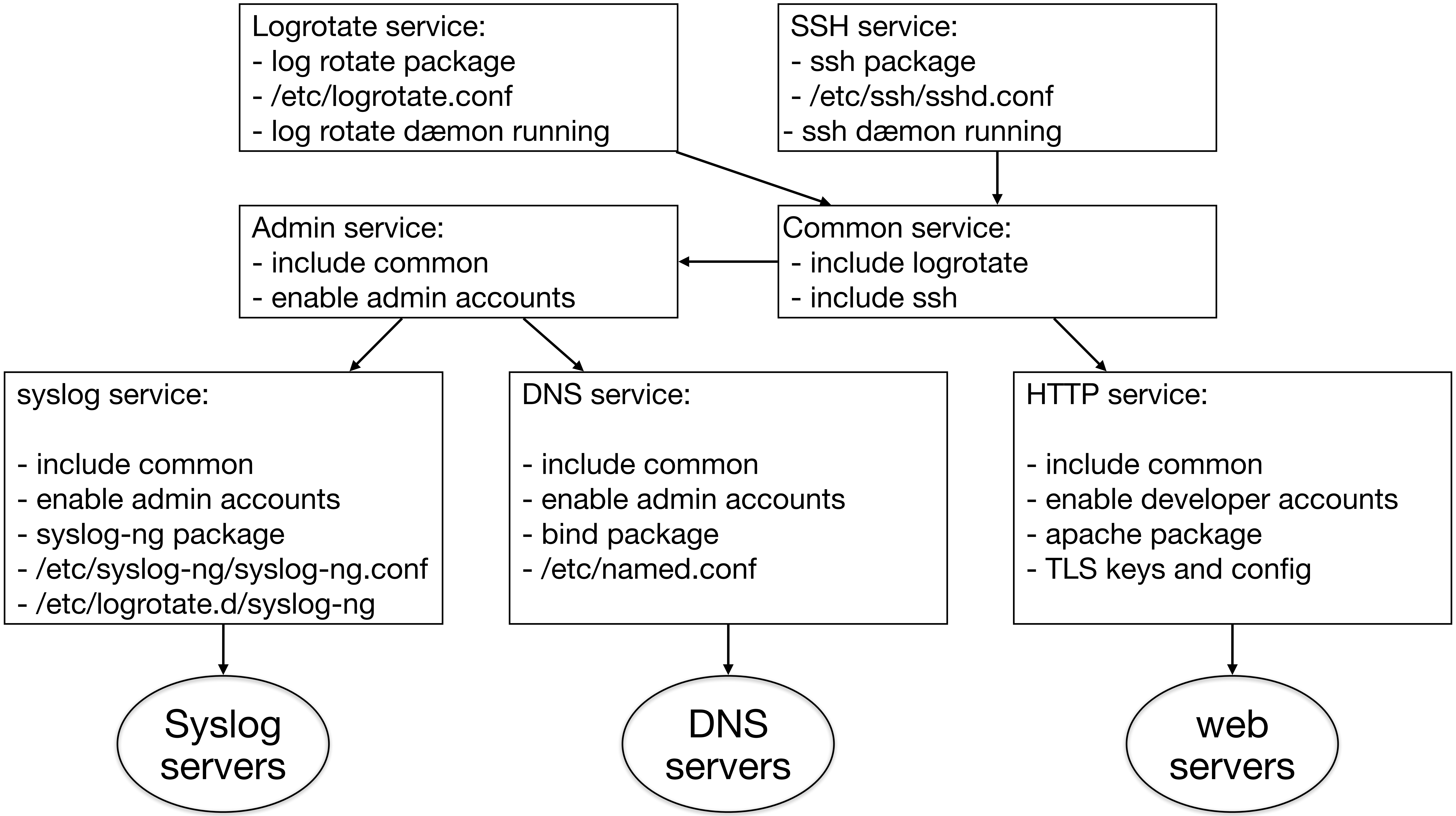

syslog service:

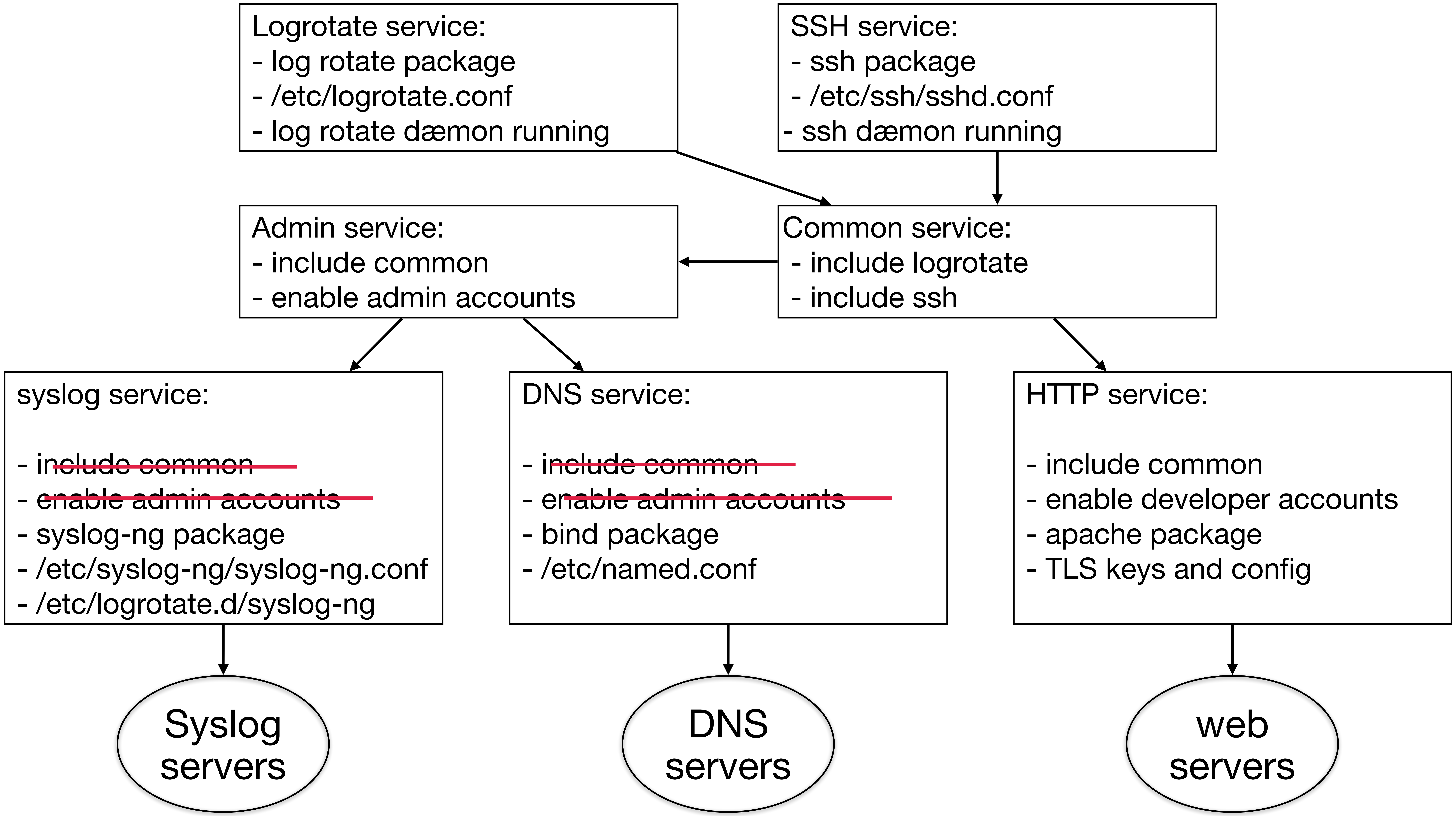
- include logrotate
- include ssh service
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng

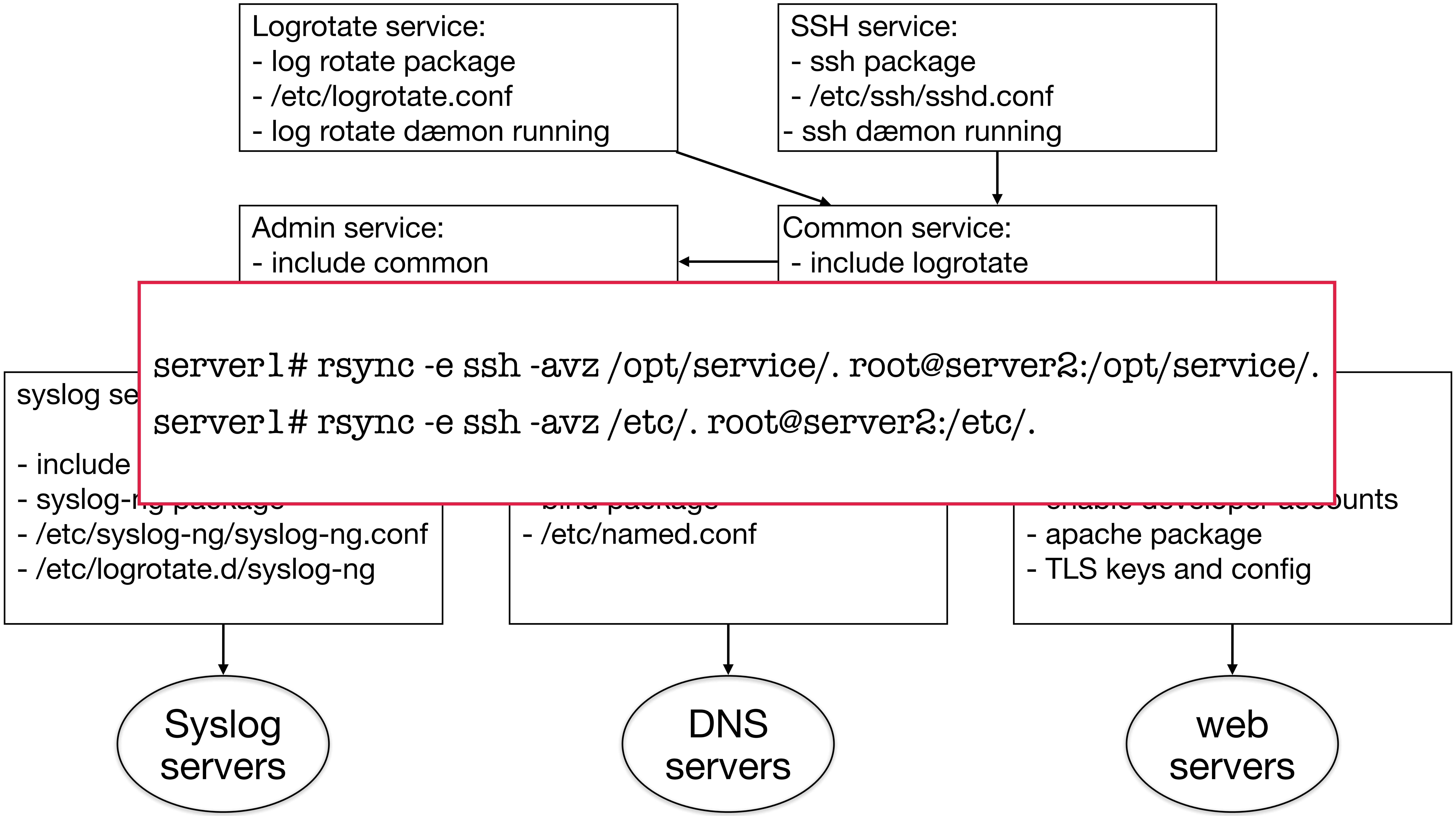
```
bundle agent sshd(parameter) {
  files:
     “/tmp/sshd_config.tmpl"
        perms       => mog(“0600","root","root"),
        copy_from => secure_cp(“/templates/etc/ssh/sshd_config",
                                 “cf-master.example.com");

     “/etc/ssh/sshd_config"
        perms     => mog(“0600","root","root"),
        create     => true,
        edit_line => expand_template(“/tmp/sshd_config.tmpl"),
        classes    => if_repaired(“restart_sshd");

     commands:
        restart_sshd:
           "/etc/rc.d/sshd restart”
  }
```


Logrotate service:
- log rotate package
- /etc/logrotate.conf
- log rotate dæmon running
SSH service:
- ssh package
- /etc/ssh/sshd.conf
- ssh dæmon running
syslog service:
- include logrotate
- include ssh service
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng
DNS service:
- include logrotate
- include ssh service
- enable admin accounts
- bind package
- /etc/named.conf
HTTP service:
- include logrotate
- include ssh service
- enable developer accounts
- apache package
- TLS keys and config
Syslog servers
DNS servers
web servers


Logrotate service:
- log rotate package
- /etc/logrotate.conf
- log rotate dæmon running
SSH service:
- ssh package
- /etc/ssh/sshd.conf
- ssh dæmon running
Common service:
- include logrotate
- include ssh
syslog service:
- include common
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng
DNS service:
- include common
- enable admin accounts
- bind package
- /etc/named.conf
HTTP service:
- include common
- enable developer accounts
- apache package
- TLS keys and config
Syslog servers
DNS servers
web servers


Logrotate service:
- log rotate package
- /etc/logrotate.conf
- log rotate dæmon running
SSH service:
- ssh package
- /etc/ssh/sshd.conf
- ssh dæmon running
Admin service:
- include common
- enable admin accounts
Common service:
- include logrotate
- include ssh
syslog service:
- include common
- enable admin accounts
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng
DNS service:
- include common
- enable admin accounts
- bind package
- /etc/named.conf
HTTP service:
- include common
- enable developer accounts
- apache package
- TLS keys and config
Syslog servers
DNS servers
web servers


Logrotate service:
- log rotate package
- /etc/logrotate.conf
- log rotate dæmon running
SSH service:
- ssh package
- /etc/ssh/sshd.conf
- ssh dæmon running
Admin service:
- include common
- enable admin accounts
Common service:
- include logrotate
- include ssh
syslog service:
- ~~include common~~
- ~~enable admin accounts~~
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng
DNS service:
- ~~include common~~
- ~~enable admin accounts~~
- bind package
- /etc/named.conf
HTTP service:
- include common
- enable developer accounts
- apache package
- TLS keys and config
Syslog servers
DNS servers
web servers



```
server1# rsync -e ssh -avz /opt/service/. root@server2:/opt/service/.
server1# rsync -e ssh -avz /etc/. root@server2:/etc/.
```

Exercises

In our next video: CM system capabilities, state assertion, and the CAP Theorem.

- Review Variable vs. Static & Shareable vs. Non-Shareable Data — classify the common directories you might need to sync across machines accordingly.
- Identify a few common aspects of a service or a system and try to explicitly define its service description.
- Read up on Ansible, CFEngine, Chef, Puppet, and Saltstack. What do they have in common? How do they differ? How would you choose which one to use?
- How does *Configuration Management* relate to *Infrastructure as Code* or *Service Orchestration*?

Links

- https://en.wikipedia.org/wiki/Software_configuration_management
- https://en.wikipedia.org/wiki/Puppet_(software)
- https://en.wikipedia.org/wiki/Chef_(software)
- https://en.wikipedia.org/wiki/CFEngine
- https://en.wikipedia.org/wiki/Ansible_(software)
- https://en.wikipedia.org/wiki/Salt_(software)
- https://en.wikipedia.org/wiki/Infrastructure_as_code