ICA0002: IT Infrastructure Services

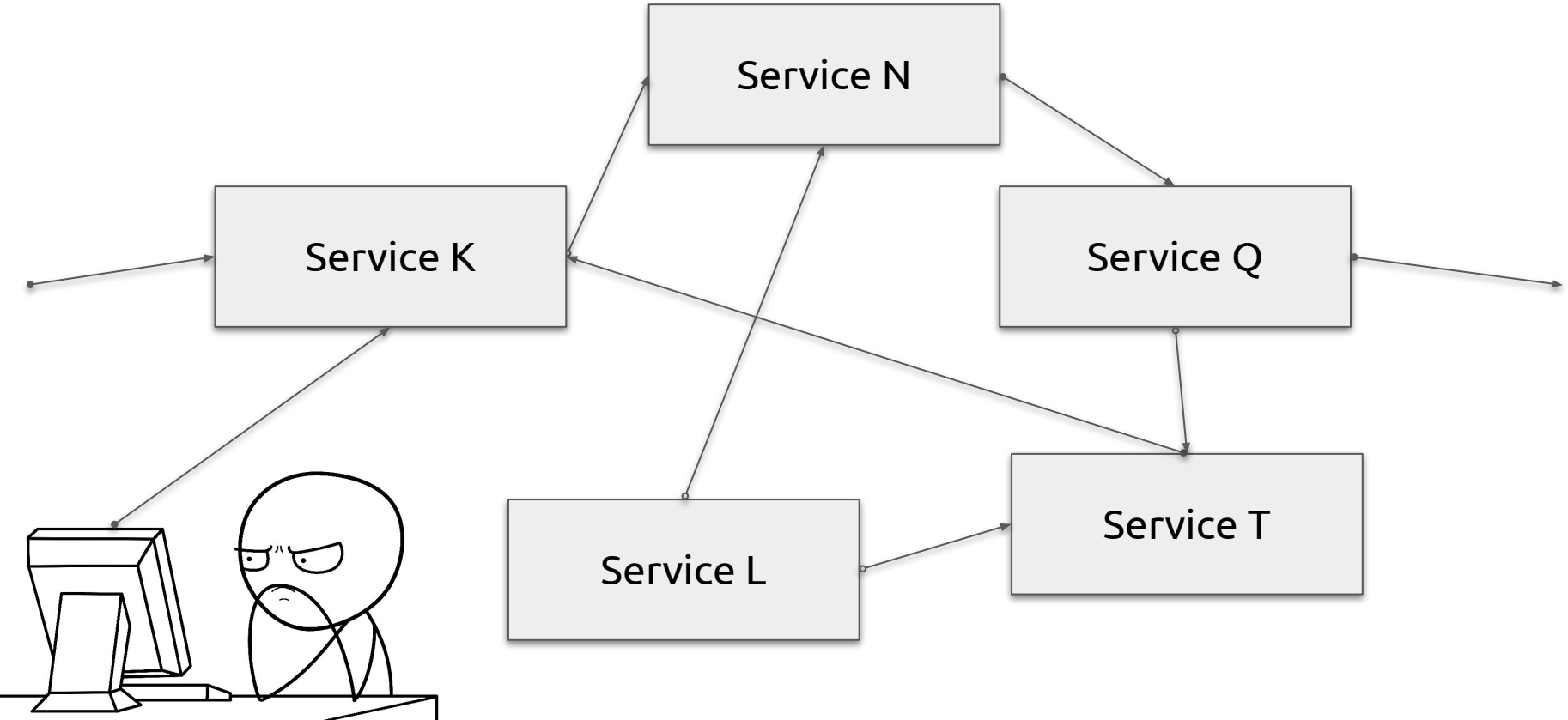
Troubleshooting

Roman Kuchin
Juri Hudolejev
2023

Troubleshooting infrastructure services

1. **Identify the misbehaving component**
2. Identify the problem
3. Find the acceptable solution
4. Fix it!

Identify the misbehaving component



Troubleshooting infrastructure services

1. ~~Identify the misbehaving component~~ ✓
2. **Identify the problem**
3. Find the acceptable solution
4. Fix it!

Identify the problem

"I did everything correctly but it doesn't work!" -- is **not** a problem definition

Most common problems:

- Service is not running
- Service is running but not working correctly
- Service cannot communicate with another service

Problem:

Service is not running

Problem: service is not running

How to detect, option 1:

```
ps ax | grep <my-process-name>
```

- **ps** - lists running OS processes
- **grep** - filter out all but your service process

The simplest and fastest option to detect if the service is running

Problem: service is not running

Example command to check if process is running:

```
ps ax | grep nginx
```

Example output if the process is running:

```
 883 ?        Ss   0:00  nginx: master process /usr/sbin/nginx -g ...
 902 ?        S    0:00  nginx: worker process
1636 pts/0    S+   0:00  grep --color=auto nginx
```

Example output if the process is not running:

```
1638 pts/0    S+   0:00 grep --color=auto nginx
```

Problem: service is not running

How to detect, option 2:

```
systemctl status <my-service-name>
```

Alternative:

```
service <my-service-name> status
```

Both provide more details about the service and also some logs

Note: process, DEB package and Systemd unit names may differ!
Example: **`mysqld`** / **`mysql-server`** / **`mysql`**.

Problem: service is not running

Example command to check service status:

```
systemctl status mysql
```

Example output if the service is running:

```
...
Active: active (running) since Sat 2019-10-26 11:51:03 UTC
...
```

Example output if the service is not running:

```
...
Active: inactive (dead) since Sat 2019-10-26 12:20:08 UTC
...
```

Problem: service is not running

Most common causes:

- Service failed to start because of configuration issues
- Service failed to start because of lack of file permissions
- Service failed to start because the port in binds to is already taken
- Service did not start after the machine reboot
- You forgot to start the service :)

Configuration syntax problems

How to detect:

```
nginx -t
apache2ctl -t  (or apache2ctl configtest)
named-checkconf; named-checkzone
visudo -cf /etc/sudoers.d/my-user
```

Service built-in config checkers

The best way to check the configuration syntax of the existing files

Only a few services have this option :(

Configuration syntax problems

Example command to check the service configuration syntax:

```
nginx -t
```

Example output if the configuration syntax is correct:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Example output if the configuration syntax is not correct:

```
nginx: [emerg] unknown directive "rooot" in /etc/nginx/sites-enabled/default:4
nginx: configuration file /etc/nginx/nginx.conf test failed
```

Configuration syntax problems

How to avoid:

```
ansible.builtin.copy:
  src: default
  dest: /etc/nginx/sites-enabled/default
  validate: nginx -t -c %s
```

Validate the configuration **before** changing the actual file on server

More examples for sudo, SSH etc.:

- https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html
- https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html

Ansible validation: a common mistake

```
# File: tasks/main.yaml
  - ansible.builtin.copy:
      src: default
      dest: /etc/nginx/.../default
      validate: nginx -t -c %s
    notify:
      - Restart Nginx

# File: handlers/main.yaml
  - name: Restart Nginx
    ansible.builtin.service:
      name: nginx
      state: restarted
```

```
# File: tasks/main.yaml
  - ansible.builtin.copy:
      src: default
      dest: /etc/nginx/.../default
    notify:
      - Check Nginx config
      - Restart Nginx

# File: handlers/main.yaml
  - name: Check Nginx config
    ansible.builtin.command: nginx -t

  - name: Restart Nginx
    ansible.builtin.service:
      name: nginx
      state: restarted
```

Starting service automatically

Common practice in the Debian world:

- Start and enable the service automatically during DEB package installation

Do not rely on this behavior! This is just a convention, not a law

Ansible task to ensure that the service is started and enabled on boot:

```
ansible.builtin.service:
  name: nginx
  state: started
  enabled: true
```

Problem:

Service is running but
not working correctly

Problem: service not working correctly

Examples:

- Key-based SSH is configured but it is still asking for a password on login
- Nginx should listen on port 8080 but it is only listening on 80
- User **`root`** can log in to MySQL but user **`elvis`** can not
- Bind should transfer entire zone but it only resolves individual addresses
- Script should backup these three directories but it backs up only this one
- etc.

Problem: service not working correctly

How to detect:

- Make sure that service **is actually running**
  - Rarely a solution, but is a very quick thing to check and exclude: "low hanging fruit"
- Check the configuration
  - If the daemon has started -- it's almost certainly a configuration **logic** problem, syntax is ok
- Check the logs

No silver bullet exists -- every service has its own specifics

Problem: service not working correctly

Logs:

- **/var/log/<service-name>/*.log** or **/var/log/<service-name>.log**
- **/var/log/syslog**

Alternative commands for Systemd journal:

- **journalctl -fu <service-name>**
- **journalctl -xn**

Hints for log monitoring

Follow logs:

```
tail -f /var/log/nginx/error.log /var/log/syslog
```

Only print needed logs (containing 'cron' in this example):

```
grep -i cron /var/log/syslog
```

Filter out some lines (print all but lines containing 'systemd' in this example):

```
grep -v systemd /var/log/syslog
```

Problem: service not working correctly

Verbose (debug) mode:

- `ansible-playbook -v infra.yaml`
- `curl -v http://localhost:8080`
- `wget -d http://localhost:8080`
- `ssh -vvv my-user@my-server`

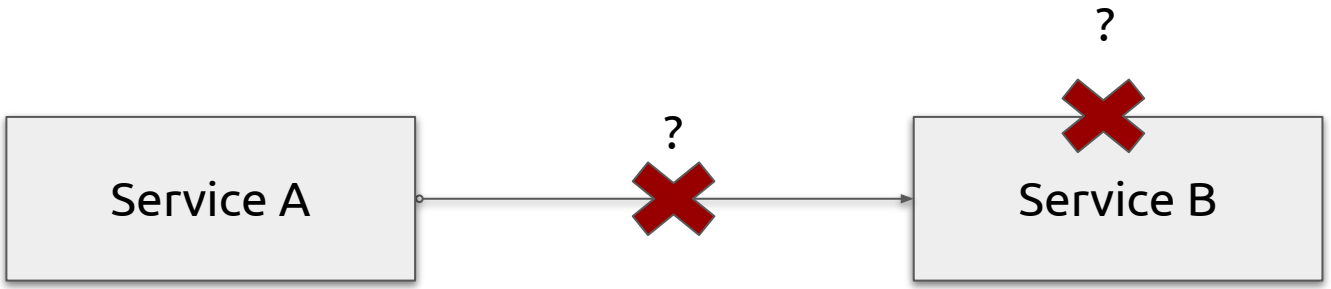
Nginx configuration file: `error_log /var/log/nginx/error.log debug;`

^^^ you'll rarely need this; **do not enable permanently in production!**

Problem:

Service cannot communicate to another service

Service communication issues



Service communication issues

First thing: make sure both services are **running** and **working correctly**

Next thing: detect, identify and fix connectivity issues

Common connectivity issues:

- Machine is not connected to the network
- Machine cannot reach another machine
- Service A cannot connect to another machine's port N

Service communication issues

How to check if machine is connected to the network:

```
ping 1.1.1.1
```

Example output if the machine is connected (**Ctrl+C** to stop pinging):

```
64 bytes from 1.1.1.1: icmp_seq=3 ttl=56 time=9.70 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=56 time=10.3 ms
…
```

Example output if the machine is not connected:

```
connect: Network is unreachable
```

Service communication issues

How to check if machine can reach another machine:

```
ping <another-machine-address>
```

Alternative to ping (provides more info about the packet path):

```
traceroute <another-machine-address>
```

Service communication issues

How to check if one can connect to another machine's port:

```
nc -vz <another-machine-address> <port>
```

Alternative if **nc** is not available:

```
telnet <another-machine-address> <port>
```

Service communication issues

Example command to check if service is listening on the remote port:

```
nc -vz 192.168.42.37 8080
```

Example output if the service is listening on the remote port:

```
Connection to 192.168.42.37 8080 port [tcp/http-alt] succeeded!
```

Example output if the service is not listening on the remote port:

```
nc: connect to 192.168.42.37 port 8080 [tcp/http-alt] failed: Connection refused
```

Network sockets

How to detect if service is listening on the correct port:

```
sudo ss -lptu  # listening, process info, TCP, UDP
```

Needs admin privileges to list process names (**-p** option)

Alternative if **ss** is not available:

```
sudo netstat -lptu
```

Network sockets

Example command to list all services listening on TCP ports:

```
sudo ss -lpt
```

Example output if the service is listening on the **public** interface:

```
State   Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
LISTEN  0       0       0.0.0.0:22          0.0.0.0:*          users:(("ssh",pid=821,fd=7))
LISTEN  0       0       0.0.0.0:80          0.0.0.0:*          users:(("nginx",pid=948,fd=13))
LISTEN  0       0       0.0.0.0:8080        0.0.0.0:*          users:(("nginx",pid=948,fd=13))
```

Example output if the service is listening on the **local** interface:

```
LISTEN  0       0       127.0.0.1:3306      0.0.0.0:*          users:(("mysqld",pid=946,...
```

Troubleshooting infrastructure services

1. ~~Identify the misbehaving component~~ ✓
2. ~~Identify the problem~~ ✓
3. **Find the acceptable solution**
4. Fix it!

Finding the solution

1. Check service documentation:
   - `<service-name> --help` example: `ansible --help`
   - `man <service-name>` example: `man nginx`
   - `info <service-name>` example: `info ssh`
   - `ansible-doc <module>` example: `ansible-doc apt`
2. Ask Google, DuckDuckGo, Bing etc.
   - Localize the problem: it is easier is to find a solution for exact problems
   - "Nginx is not working" vs. "Nginx is not listening on port 8080"
3. Try rubber duck debugging -- not a joke, it really works
4. Ask a colleague for help

^^^ Strictly in **this** order!

Troubleshooting infrastructure services

1. ~~Identify the misbehaving component~~ ✓
2. ~~Identify the problem~~ ✓
3. ~~Find the acceptable solution~~ ✓
4. **Fix it!**

Questions?