IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

Infrastructure parts

- Network
- Hardware (optional)
- Virtualization platform
- Container platform
- Supporting services
  - Load balancers
  - DNS
  - DHCP
  - Proxies
  - Databases(mysql, postgresql, mongodb, elastic, influxdb, couchdb)
- Services
  - Web
  - Applications

Managing infrastructure

- Manually
  - rtfm -> console -> seems working
- Infrastructure as a code
  - rtfm -> code -> staging/testing -> works

IaaC

- Takes more time at start
- Takes way more actions to start
- Slow, very slow
- No normal errors
- New languages, new surprises
- More bugs

IaaC?

- Version control
- Staging
- Code review
- Auto testing
- Scalability
- Components reuse

Management tools

- Ansible
- Chef
- Puppet
- SaltStack
- Terraform

Ansible

- Easy to use (when compare to Chef, Puppet, Salt)
- Uses SSH, no client needed
  - Can be used in networking as well
- Support Jinja2 templates
- Lots of built-in modules
- IaaD

Ansible parts

- Control node (Where Ansible runs)
- Managed node
- Inventory
- Playbook
- Play
- Task
- Module

Docs: https://docs.ansible.com/ansible/latest/user_guide/index.html

Inventory

```
[switches]
1.1.1.3
1.1.1.4


[routers]
1.1.1.1
gw1.mydomain.net


[network-devices:children]
switches
routers
```

Ansible config

1. ANSIBLE_CONFIG (env)
2. ansible.cfg (current dir)
3. .ansible.cfg (~)
4. /etc/ansible/ansible.cfg

Modules

- ansible.builtin.apt
- ansible.builtin.user
- ansible.builtin.hostname
- ansible.builtin.ping
- cisco.ios.ios_config
- community.mysql.mysql_db

Docs: https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Ad-hoc vs playbooks

```
ansible network-devices -m “raw” -a “show ip int br” -u cisco --ask-pass

ansible-playbook get_interfaces.yml
```

Useful links

Course materials: https://github.com/romankuchin/ica0002-2021

Course rules: https://github.com/romankuchin/ica0002-2021/blob/main/rules.md

VM resourses: http://193.40.156.86/

GitHub bot: https://github.com/ica0002-bot

ICA0002: IT Infrastructure Services

Ansible Basics

Roman Kuchin
Juri Hudolejev
2021

Files from previous lab

```
roles/
    test_connection/
            tasks/
                main.yaml
ansible.cfg
hosts
infra.yaml
```

Files from previous lab

`roles/`

Directory where all our infrastructure service roles are stored

Role defines a service type: web server, database server, mail server etc.

Each role is defined in its own subdirectory and
should contain a file **`tasks/main.yaml`**

`roles/test_connection/`

Custom role we have created to test Ansible setup

Files from previous lab

`roles/test_connection/tasks/main.yaml`

YAML file, list of tasks for **`test_connection`** role

For every server that has this role assigned,
every task from this role is executed

Task

Action to ensure that certain element (file, service etc.) is in desired state

On task execution Ansible will change the element state -- if needed

Ansible task example

```
name: Ansible ping module
ansible.builtin.ping:
```

Where

`name` is free text task identifier for better readability

`ansible.builtin.ping` is Ansible module call

Note the colons! '**`ping:`**' means 'module **`ping`**, no parameters'.

Another Ansible task example

Ensures that Linux user with login 'elvis' exists; creates one if not:

```
name: Linux user
ansible.builtin.user:   ← Ansible module to manage OS users
  name: elvis           ← Module parameter named 'name'
                          with value 'elvis'; note the indent!
```

All Ansible modules:
https://docs.ansible.com/ansible/latest/collections/index_module.html

Yet another Ansible task example

Ensures both APT packages **`curl`** and **`wget`** are installed; installs if not:

```
name: HTTP client packages
ansible.builtin.apt:
  name:
    - curl
    - wget
```

And another Ansible task example

Ensures local file is copied to a managed host; copies if not:

```
name: Really important file (no)
ansible.builtin.copy:
  src: foo.txt
  dest: /srv/foo.txt
```

Note: file **`files/foo.txt`** must exist in the role directory for this to work:

```
roles/my_role/files/foo.txt
roles/my_role/tasks/main.yaml
```

Files from previous lab

`infra.yaml`

Playbook, YAML file, list of plays

Ansible execution entry point

Play

Maps a group of roles to a group of servers (hosts)

Ansible plays and playbooks:
https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

Ansible play example

Assigns role **`test_connection`** to **`foo-1`** host:

```
name: Ansible connection test
hosts: foo-1
roles:
  - test_connection
```

Another Ansible play example

Assigns multiple roles to every defined host named 'web*' -- **web-1**, **web-2** etc.:

```
name: Public web servers
hosts: web*
roles:
  - firewall
  - nginx
```

Ansible playbook example

```
- name: Database servers
  hosts: db_servers
  roles:
    - mysql

- name: Web servers
  hosts: web_servers
  roles:
    - nginx
```

Playbook is just a list of several plays -- may contain 0, 1 or more plays

YAML

Human readable data serialization language with minimal syntax

```
key: value              # map keys are followed by ': ' and optional value
foo:                    # maps can include other maps
  bar: 0.42             # numbers are written as is
list:                   # comments start with '#'
  - one element         # list elements start with '-'
  - another element     # indents are important!
```

More info: https://yaml.org

Files from previous lab

`hosts`

Inventory file, high-level overview of your entire infrastructure:

- List of servers
- Server groups
- Sometimes server variables: connection parameters etc.

Format -- flavor of INI (other formats also supported):

```
[my_group]
server-name   param=value   another_param=another_value   ...
```

Ansible inventory file example

```
[db_servers]
db-1   ansible_host=10.10.10.37

[web_servers]
web-1  ansible_host=10.10.10.13  ansible_port=9522
web-2  ansible_host=10.10.10.14  ansible_port=9622

[all:vars]
ansible_ssh_user=ansible
```

Ansible inventory file location

In this course: file named **`hosts`** in your repository

Default location: **`/etc/ansible/hosts`** ← do not use

More about Ansible inventory:
https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Files from previous lab

`ansible.cfg`

Ansible configuration file -- changes some aspects of Ansible behavior

So far (lab 1..2) our Ansible configuration file only instructs Ansible to fetch the inventory from the same directory:

```
[defaults]
inventory = ./hosts
```

We will improve Ansible configuration file later during this course

Ansible configuration file format

Flavor of INI format:

```
[section_name]
key_name = value
another_key_name = another_value

[another_section_name]
...
```

More about Ansible configuration:
https://docs.ansible.com/ansible/latest/reference_appendices/config.html

Ansible configuration file locations

`ansible.cfg` (root of your repository)

Project specific configuration

`~/.ansible.cfg` (your home directory)

User specific configuration for all projects on this machine

`/etc/ansible/ansible.cfg`

Global configuration for all users on this machine

Ansible configuration file locations

`ansible.cfg` ← go for it

Project specific configuration

`~/.ansible.cfg` ← do not use

User specific configuration for all projects on this machine

`/etc/ansible/ansible.cfg` ← do not use

Global configuration for all users on this machine

Questions?

ICA0002: IT Infrastructure Services

SSH Basics

Roman Kuchin
Juri Hudolejev
2021

SSH: Secure Shell

Remote shell operated securely over insecure network

Replaced **`telnet`**, **`rsh`**, **`rlogin`** and **`rexec`**

De-facto standard tool to operate remote machines

Default connection protocol in Ansible

More info: https://www.ssh.com/academy/ssh

SSH: Secure Shell

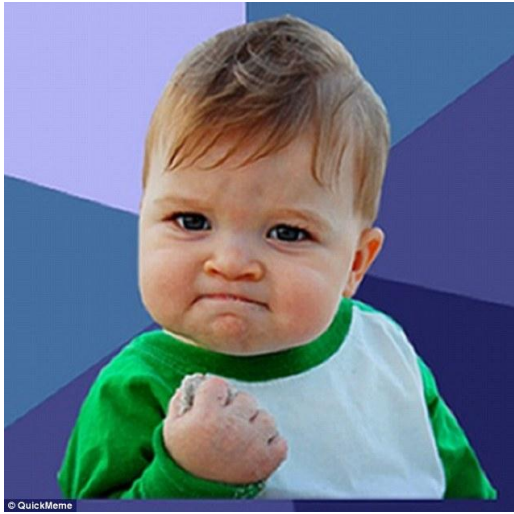
Remote shell operated securely over insecure network

Replaced **`telnet`**, **`rsh`**, **`rlogin`** and **`rexec`**

De-facto standard tool to operate remote machines

Default connection protocol in Ansible

More info: https://www.ssh.com/academy/ssh



Encryption

Symmetric encryption: DES, AES etc.

- Same key for encryption and decryption -- shared secret

Asymmetric (public key) encryption: DSA, RSA etc.

- Public key (openly distributed) + private key (kept secret)
- Message encrypted with one key from the pair
  can only be decrypted with the other key from the same pair

Encryption

Symmetric encryption: DES, AES etc.

Requires secure channel to exchange the shared secret :(

Asymmetric (public key) encryption: DSA, RSA etc.

Is unacceptably inefficient on large data :(

Let's use a **public key** to create a secure channel for a shared key exchange, and then use that **shared key** to encrypt the data...

That would work!

SSH session initialization

Client sends the connection request to the server

Then client and server both:

- agree algorithms for key exchange, symmetric and public key encryption
- generate shared session key using Diffie-Hellman method (SSH v2)

Then server performs the client authentication

If all good, connection is established

SSH client authentication

Authentication options:

- User password based
- User key based: RSA, DSA, ECDSA etc.
- Host key based
- Interactive -- for one time passwords
- GSSAPI -- for external authentication services such as Kerberos

SSH client authentication

Authentication options:

- User password based ← avoid at all costs
- User key based: RSA, DSA, ECDSA etc. ← we only use this on this course
- Host key based
- Interactive -- for one time passwords
- GSSAPI -- for external authentication services such as Kerberos

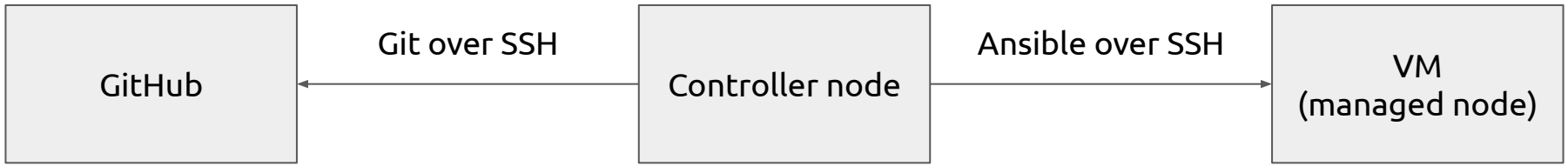
SSH public key based client authentication

In very simple terms:

- Client encrypts (signs) certain data with its **private** key
- **Public** key and signature are sent to SSH server
- SSH server checks if the public key is acceptable (authorized)
- SSH server verifies signature
- If all checks passed -- client is authenticated

More detailed info: https://tools.ietf.org/html/rfc4252

SSH in this course



Your SSH keys in this course

Your public key:

- **`~/.ssh/id_rsa.pub`** file on Controller node (connect from)
- **`~/.ssh/authorized_keys`** file on your VMs (connect to)
- In your GitHub account: https://github.com/<username>.keys (lab 1)

Your private key:

- **`~/.ssh/id_rsa`** file on Controller node -- should never leave your machine!

Public key may also be extracted from the private key file, but not vice versa!

Important!

If your private key is lost or compromised,

1. Delete the corresponding public key from your GitHub account immediately!
2. Generate a new key pair (see lab 1)
3. Contact the teachers to reset the keys on your VMs

Questions?

ICA0002: IT Infrastructure Services

Web Servers

Roman Kuchin
Juri Hudolejev
2021

Basic terms

WWW: World Wide Web, the Web

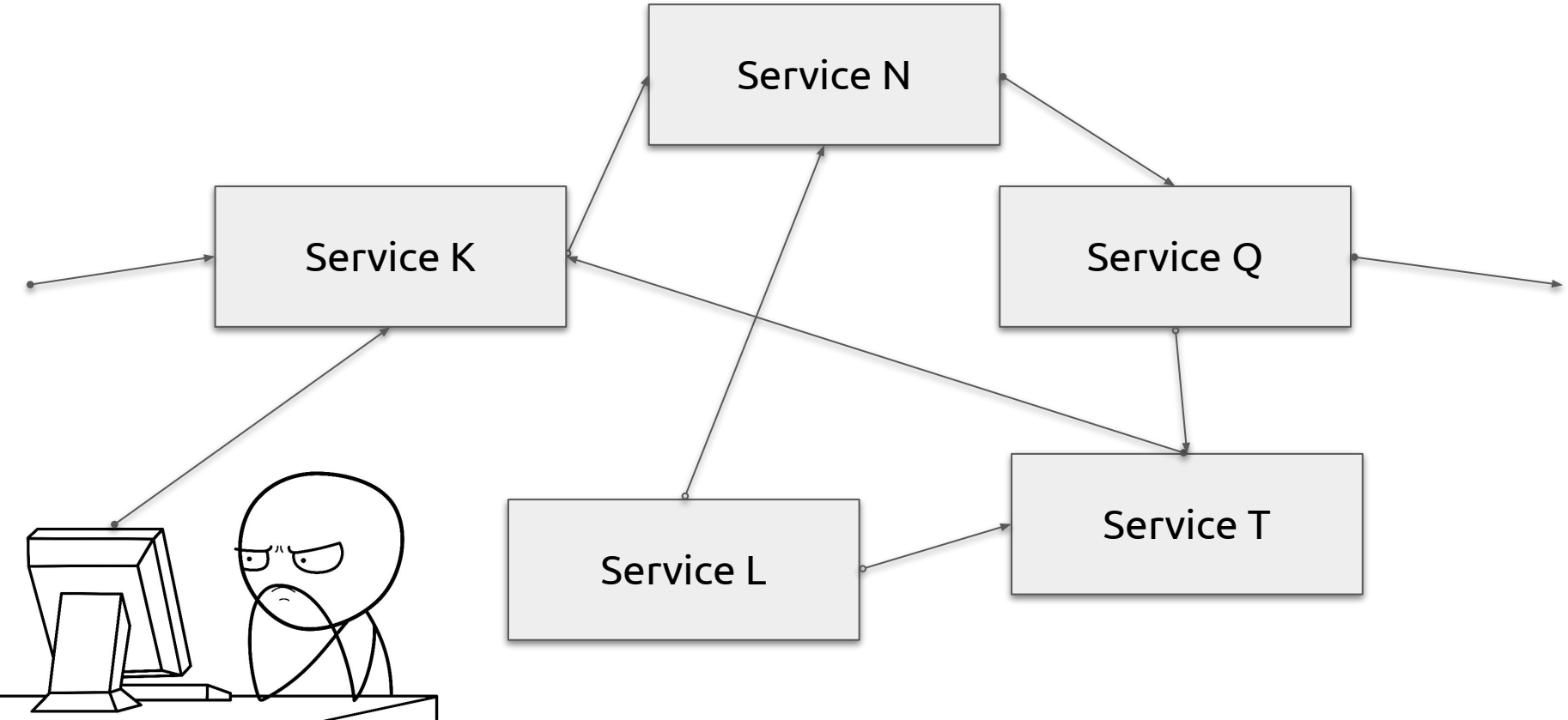
URL: Uniform Resource Locator

Web server

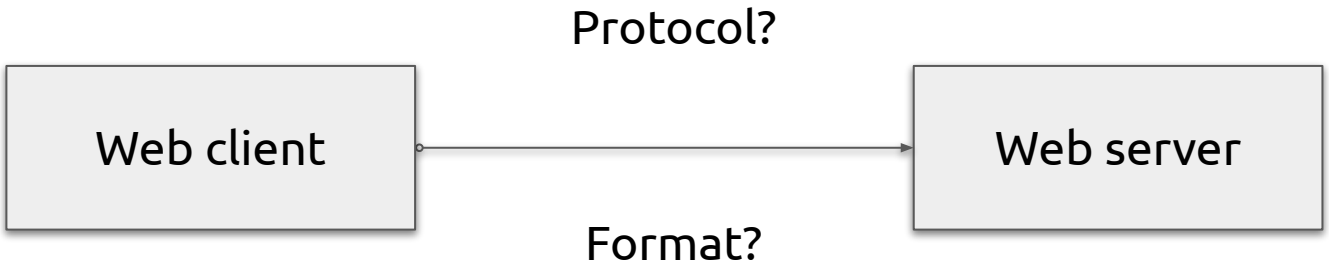
Web client (web browser, user agent)

(very) Generic IT service

Service communication



Web client and Web server



Web client and Web server

Web client and Web server

Demo time!

Behind the Web server

Behind the Web server

Static documents:

- web server sends files from local filesystem as is

Dynamic documents:

- web server calls scripts to generate the resource on the fly (dynamically) and sends that generated resource to the client

Proxy mode:

- web server forwards request to other services

Other topics

Not discussed in detail in this course -- but some are covered in labs as needed:

- HTTPS, SSL/TLS
- WebSockets, HTTP/2, HTTP/3
- CGI, WSGI
- Caching
- Proxying
- High availability

Web server market share

Apache HTTPd

The oldest of the existing mainstream web servers, and still widely used

Free and open-source, maintained by Apache Software Foundation

First release in 1995, current version: 2.4

Modules for TLS, server-side scripts, authentication, proxying, etc.

- List of modules: https://en.wikipedia.org/wiki/List_of_Apache_modules

Web site: https://httpd.apache.org

Nginx

Rather 'new' web server: first public release in 2004, current version: 1.18

- Nginx: free and open-source (BSD license)
- Nginx Plus: proprietary

One of the most widely used -- second after Apache HTTPd (released in 1995)

Web server, HTTP proxy, load balancer

- List of modules: https://www.nginx.com/nginx-wiki/build/dirhtml/modules

Web site: https://nginx.org

Questions?

ICA0002: IT Infrastructure Services

Ansible Handlers

Roman Kuchin
Juri Hudolejev
2021

Types of applications

Controlled by user

- Interaction via GUI actions or CLI commands

Controlled by system

- Run in background
- Interaction via configuration files (input) and log files (output)
- Called 'daemons' in UNIX ('services' in Windows -- idea is the same)

Daemons

- Apache HTTPd -- web server daemon ('d' stands for 'daemon')
- **`syslogd`** -- system logging daemon
- **`sshd`** (not **`ssh`**) -- SSH server daemon
- **`mysqld`** (not **`mysql`**) -- MySQL server daemon

Common things:

- Name ends with 'd' to indicate that the application is daemon
- Daemon reads configuration file only once -- while starting
- **If the configuration is changed, daemon needs to be restarted***

* Some daemons can reload the configuration on system signal

Example task: configure Nginx

```
- name: Nginx configuration
  ansible.builtin.copy:
    src: default.conf
    dest: /etc/nginx/sites-enabled/default
```

Example task: configure Nginx

```
- name: Nginx configuration
  ansible.builtin.copy:
    src: default.conf
    dest: /etc/nginx/sites-enabled/default
```

Problem: service needs to be restarted to apply the changed configuration

Reload would also work here but the idea is the same

Example task: restart Nginx

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

Example task: restart Nginx

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

Problem: Nginx will be restarted on every Ansible run!

Can the tasks share the state with each other?

Ansible handlers

'Special' tasks that are **not** run by default

Handlers are called (notified) **by name** by other tasks

- If notified multiple times -- still run once
- Run in the order they are **defined** -- not the order they are notified
- Run after all tasks **in all roles** are completed

More:
https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html

Ansible handlers have the same syntax as tasks

```
- name: Nginx configuration
  ansible.builtin.copy:
    src: default.conf
    dest: /etc/nginx/sites-enabled/default
  notify: Restart Nginx

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

← 'Regular' task

← Handler

Ansible handlers are defined in their own file

```
roles/
    my_role/
        files/
            ...
        handlers/
            main.yaml
        tasks/
            main.yaml
```

Demo!

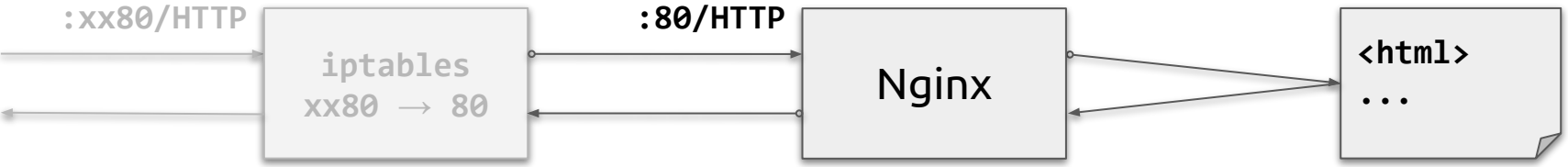
Questions?

ICA0002: IT Infrastructure Services

Web applications

Roman Kuchin
Juri Hudolejev
2021

Previous lab



Web server operation modes

Static documents:

- web server sends files from local filesystem as is

Dynamic documents:

- web server runs scripts to generate the resource on the fly (dynamically) and sends that generated resource to the client

Proxy mode:

- web server forwards request to other services

Web server operation modes

Static documents:

- web server sends files from local filesystem as is

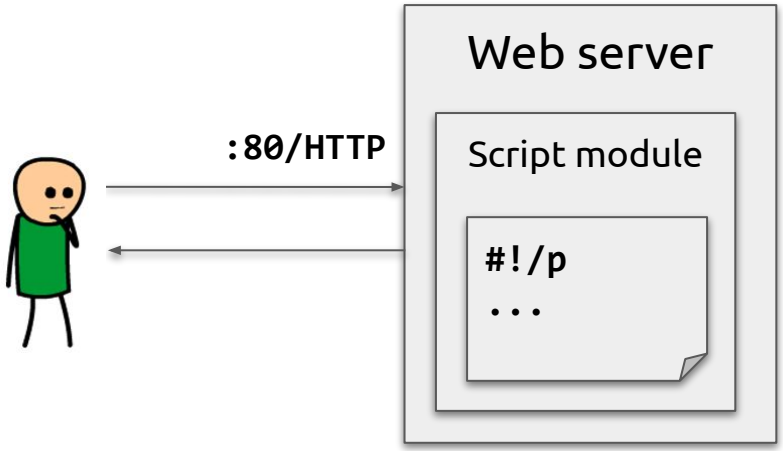
Dynamic documents:

- web server runs scripts to generate the resource on the fly (dynamically) and sends that generated resource to the client

Proxy mode:

- web server forwards request to other services

Web server script modules



Server runs the script inside the main process using the extension module

- Apache HTTPd: Perl module, PHP module etc.
- Nginx: Lua module, JavaScript module etc.

Dynamic resource example

```
<?php

echo '<h1>It works!</h1>';
```

It works!

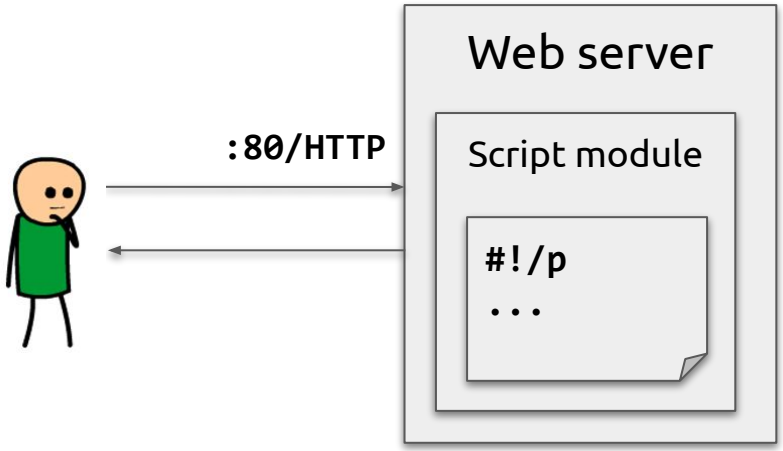
```
<?php

phpinfo();
```

PHP Version 5.2.3-1ubuntu6.3

System	Linux grenadine 2.6.18-xenU #3 SMP Thu Jan 10 15:56:11 CET 2008 i686
Build Date	Jan 10 2008 09:24:13
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
additional .ini files parsed	/etc/php5/apache2/conf.d/curl.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini, /etc/php5/apache2/conf.d/pspell.ini, /etc/php5/apache2/conf.d/tidy.ini

Web server script modules

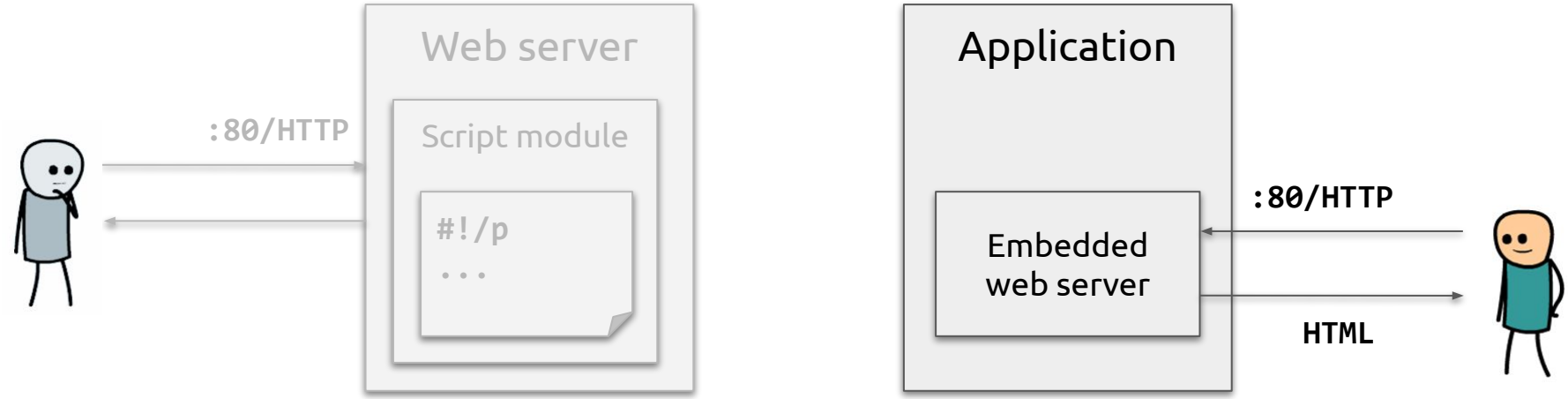


Probably the fastest method if configured correctly

Web server needs a custom module

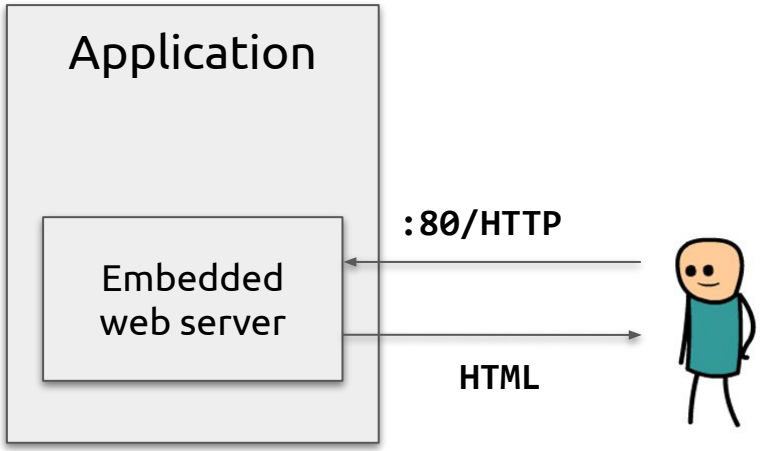
Script runs inside web server -- security issues

Embedded web servers



Instead of web server running an app (script) -- app could run a web server!

Embedded web servers



Upgrades are pain

Lack of features as compared to standalone web servers

Reimplementing the web server on every programming language

Performance issues: works for Java etc. (sort of) but not for scripting languages

External scripts

Script is executed by Web server as a separate process

The simplest and the earliest known method

External scripts

Slow and very inefficient

Script runs in the context of web server -- security issues

No standard interface for servers to communicate with scripts

Gateway interfaces

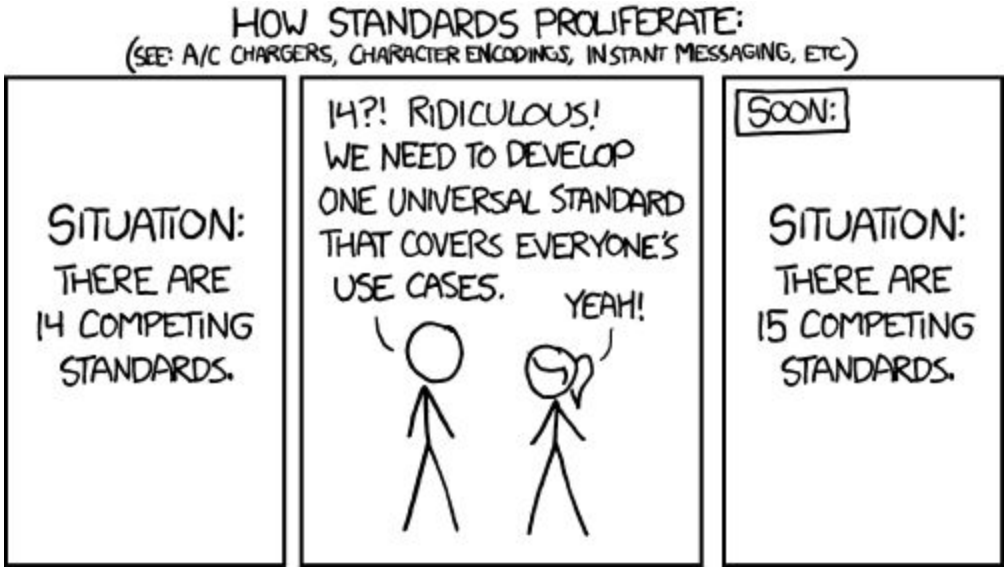
1993: Common Gateway Interface (CGI)

1996: FastCGI (binary protocol) -- scripts are run by a separate process

2001: Simple Common Gateway Interface (SCGI)

Netscape, Microsoft, Apache etc. developed their own protocols

Web server modules to run scripts are still there



https://xkcd.com/927

Gateway interfaces

1993: Common Gateway Interface (CGI)

1996: FastCGI (binary protocol) -- scripts are run in a separate process

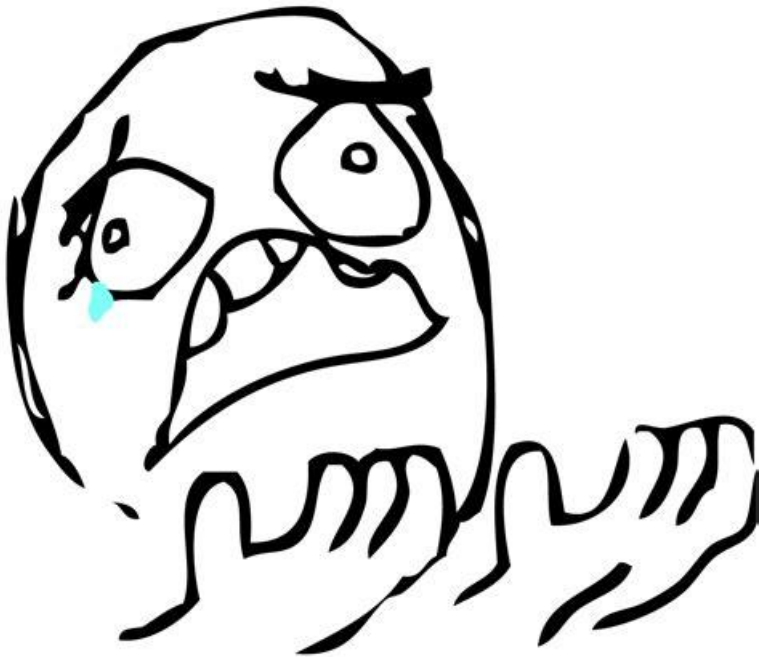
2001: Simple Common Gateway Interface (SCGI)

2003: Web Server Gateway Interface (WSGI) for Python

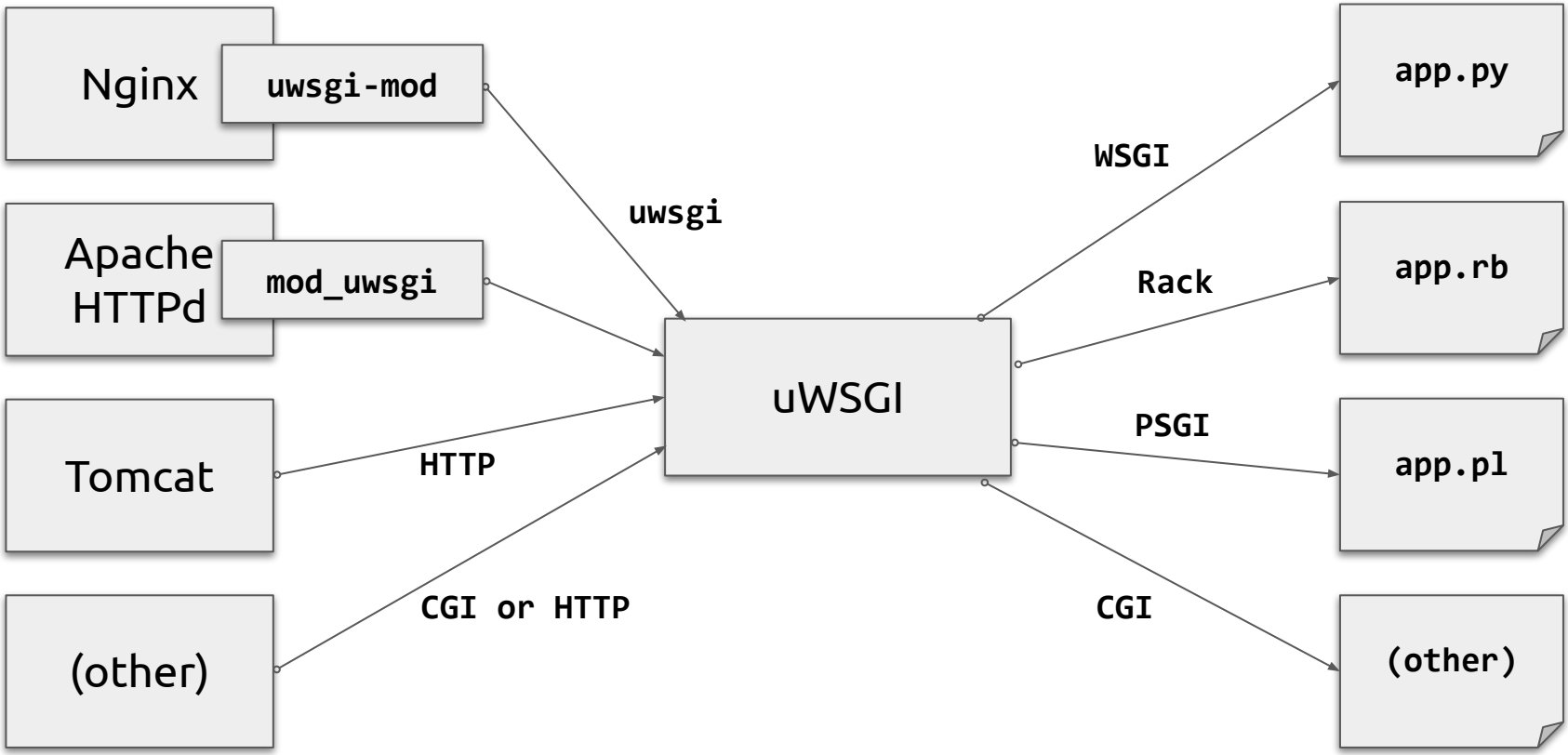
Followed by JSGI for JavaScript, PSGI for Perl, Rack for Ruby etc.

Good read: https://docs.python.org/3.4/howto/webservers.html

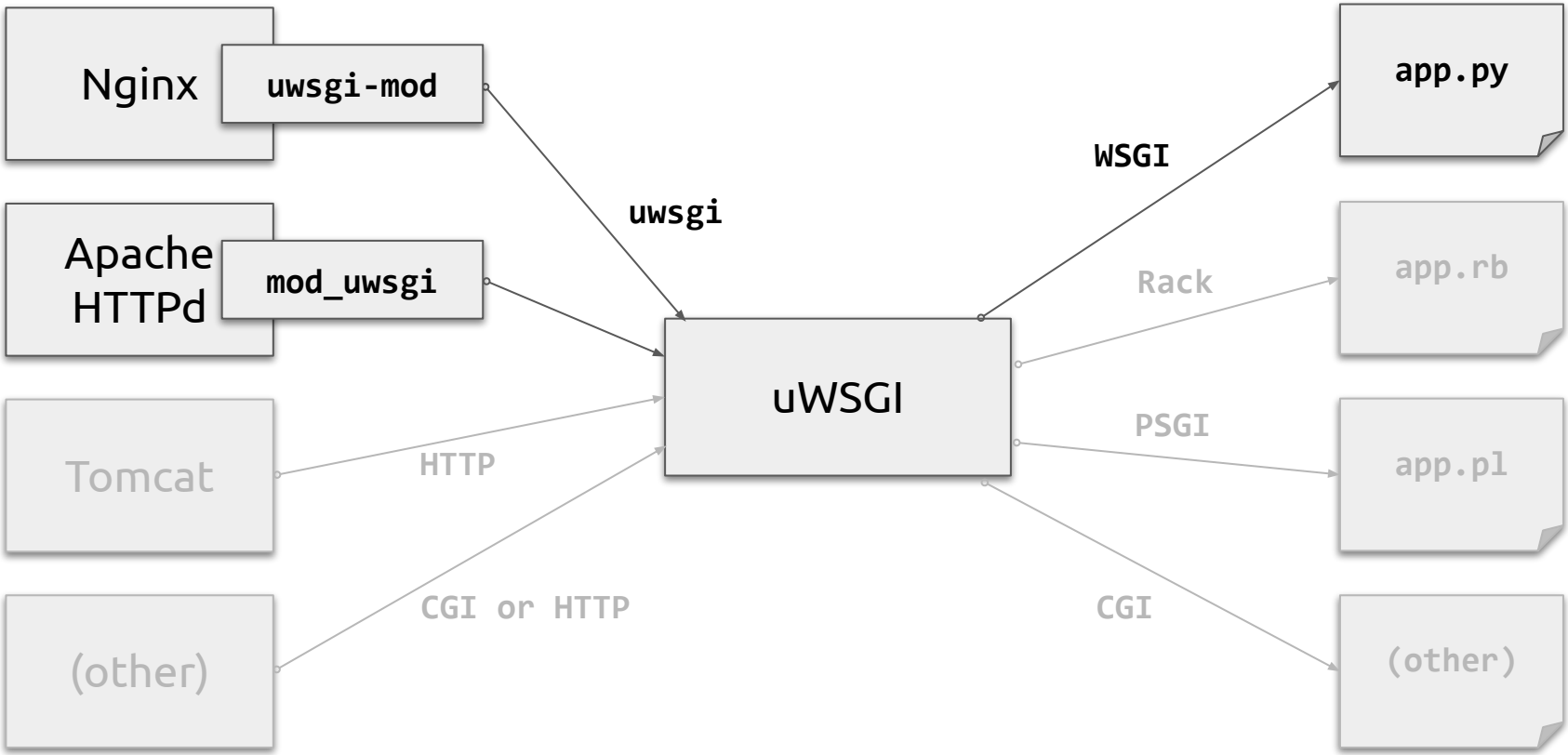
CGI SCGI FastCGI PSGI WSGI and all of these GI-s...



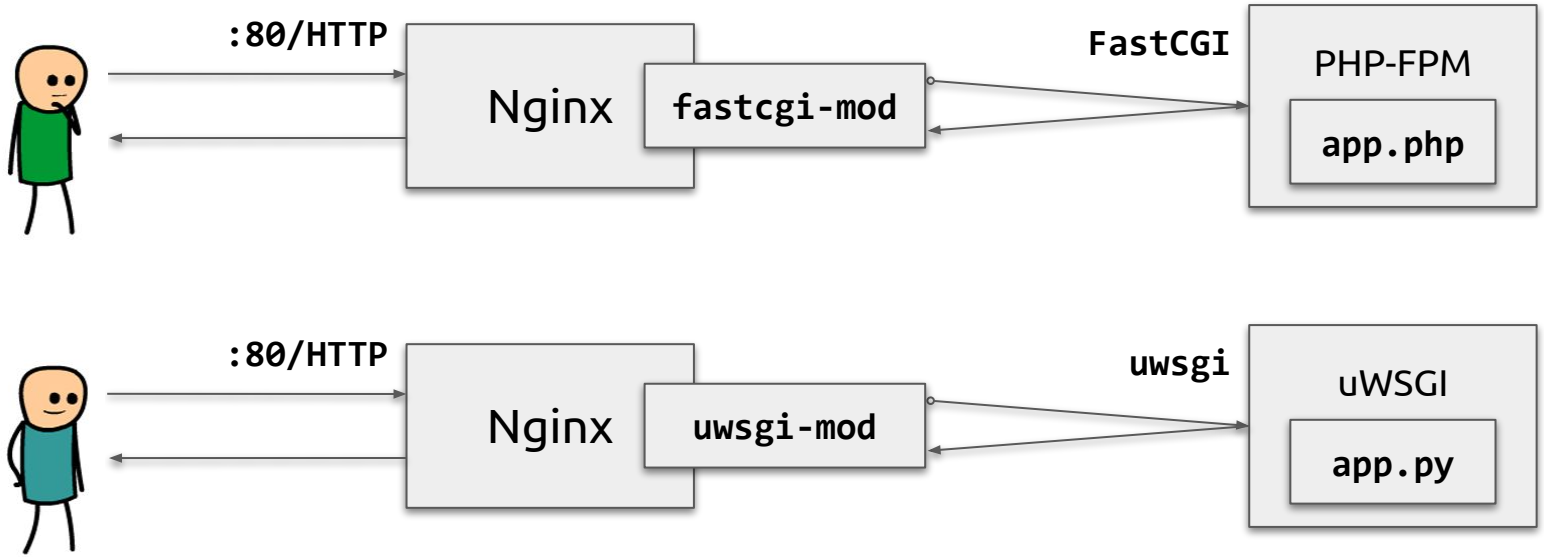
uWSGI mission



uWSGI mission



FastCGI and uwsgi examples



Script is executed by **application server** (FPM, uWSGI, Unicorn etc.)

Nginx FastCGI configuration example

```
server {
    listen 80;

    location / {
        fastcgi_pass 127.0.0.1:9000;  # may be remote host as well
        include fastcgi_params;       # found in /etc/nginx/
    }
}
```

uWSGI example is almost identical (**fastcgi_pass** → **uwsgi_pass**)

Dynamic web resources

1. Web server runs the script (app) to generate the resource

   Easier to set up but not very resource efficient and has security issues

2. App generates the resource and runs the embedded web server to serve it

   Language-specific solution, lack of features

3. Web server communicates with app server that generates the resource

   More complex to set up but is usually preferred for larger deployments

Principle 1 of Unix philosophy:

Write programs that do **one** thing and do it well.

Questions?

ICA0002: IT Infrastructure Services

Ansible variables

Roman Kuchin
Juri Hudolejev
2021

Nginx + uWSGI example

```
#roles/nginx/files/default

server {
  uwsgi_pass localhost:5000;
}

#roles/nginx/tasks/main.yaml

- name: Nginx configuration
  ansible.builtin.copy:
    src: default
    dest: /etc/.../default
```

```
#roles/uwsgi/files/app.ini

[uwsgi]
socket = localhost:5000;

#roles/nginx/tasks/main.yaml

- name: uWSGI configuration
  ansible.builtin.copy:
    src: app.ini
    dest: /etc/.../app.ini
```

Problems

1.

What if we need to change the port?

How to ensure that **all** needed files are changed?

Problems

1.

What if we need to change the port?

How to ensure that **all** needed files are changed?

2.

How to make this code support similar but **different** installations?

Server A: port 5001, server B: port 5002 etc.

Variables

Variable -- stored value identified by name

Generic example (not very Ansible related), set variable:

```
x = 42      # value: 42, name: x
```

Use variable:

```
print(x)    # addressed by name -- value (42) is printed
```

Ansible variables

Example, set:

```
uwsgi_addr: localhost:5000
```

Use:

```
socket = {{ uwsgi_addr }}
```

More: docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Why {{ … }}?

`{{ … }}` -- variable syntax used by Jinja (templating engine for Python)

Indicates that block is a variable name that should be replaced with a value:

```
"x value is x".render(x=42)          → "x value is x"
"x value is {{ x }}".render(x=42)    → "x value is 42"
```

We'll learn more about Jinja in the following labs

Jinja documentation: https://jinja.palletsprojects.com

What will Ansible do?

```
#roles/demo/files/demo.txt

x value is {{ x }}
```

```
#/opt/demo.txt

(What will we get here?)
```

```
#roles/demo/tasks/main.yaml

- name: Demo file
  ansible.builtin.copy:
    src: demo.txt
    dest: /opt/demo.txt
```

What will Ansible do?

```
#roles/demo/files/demo.txt

x value is {{ x }}
```

```
#/opt/demo.txt

x value is {{ x }}
```

```
#roles/demo/tasks/main.yaml

- name: Demo file
  ansible.builtin.copy:
    src: demo.txt
    dest: /opt/demo.txt
```

What will Ansible do?

```
#roles/demo/files/demo.txt

x value is {{ x }}
```

```
#/opt/demo.txt

(What will we get here?)
```

```
#roles/demo/tasks/main.yaml

- name: Demo file
  ansible.builtin.template:
    src: demo.txt
    dest: /opt/demo.txt
```

What will Ansible do?

```
#roles/demo/files/demo.txt

x value is {{ x }}
```

```
#/opt/demo.txt

(No changes)
```

```
#roles/demo/tasks/main.yaml

- name: Demo file
  ansible.builtin.template:
    src: demo.txt
    dest: /opt/demo.txt
```

Ansible will fail:

```
Could not find or access 'demo.txt'
Searched in:
  - roles/demo/templates/demo.txt
  - ...
```

What will Ansible do?

```
#roles/demo/templates/demo.txt

x value is {{ x }}
```

```
#/opt/demo.txt

x value is 42
```

```
#roles/demo/tasks/main.yaml

- name: Demo file
  ansible.builtin.template:
    src: demo.txt
    dest: /opt/demo.txt
```

Nginx + uWSGI example

```
#roles/nginx/templates/default

server {
  uwsgi_pass {{ uwsgi_addr }};
}

#roles/nginx/tasks/main.yaml

- name: Nginx configuration
  ansible.builtin.template:
    src: default
    dest: /etc/.../default
```

```
#roles/uwsgi/templates/app.ini

[uwsgi]
socket = {{ uwsgi_addr }};

#roles/nginx/tasks/main.yaml

- name: uWSGI configuration
  ansible.builtin.template:
    src: app.ini
    dest: /etc/.../app.ini
```

Files vs. templates

Ansible module **`ansible.builtin.copy`**:

- Uploads files from **`files/`** directory
- Files are uploaded as is, without modifications

Ansible module **`ansible.builtin.template`**:

- Uploads files from **`templates/`** directory
- Variables (**`{{ ... }}`**) are replaced with values on managed host

Ansible variables

Can be defined in multiple places:

- Command line
- Inventory file
- Separate **`group_vars`** files
- Separate **`host_vars`** files
- Plays
- Tasks
- Etc.

More: docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Ansible variables in inventory file

```
vm-1  ansible_host=193.40.156.86  ansible_port=9922  ...
```

{{ ansible_host }} → 193.40.156.86

{{ ansible_port }} → 9922

Ansible variables in this course

`hosts`

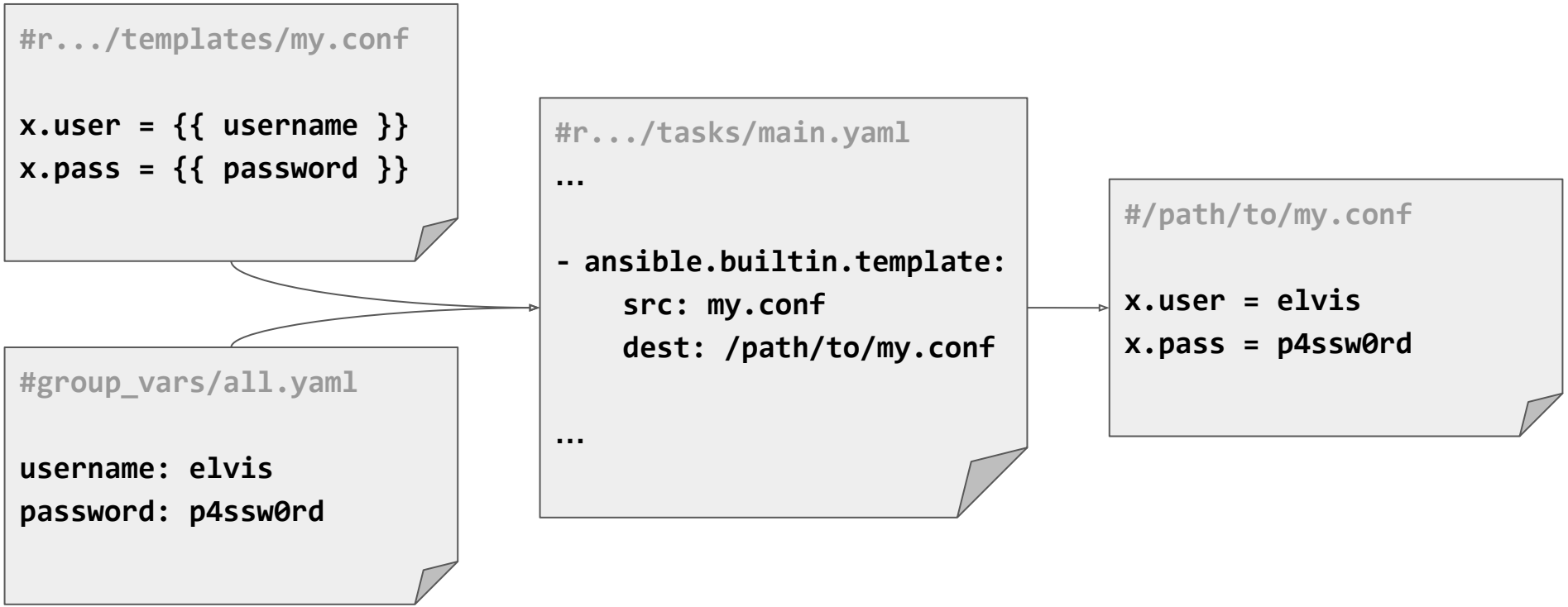
Individual host connection variables only (**`ansible_host`**, **`ansible_port`**, **`ansible_user`**)

`group_vars/all.yaml`

All the other variables; should you need to change the service port or domain name on the exam -- only this file is modified and nothing more

No variables via command line!

Ansible variables



Problems

3.

We cannot store plain text secrets in Git!

```
#group_vars/all

username: elvis
password: p4ssw0rd
```

Ansible Vault

Command line tool to encrypt and decrypt sensitive data

Can encrypt files or separate variables

More: https://docs.ansible.com/ansible/latest/user_guide/vault.html

Ansible Vault: master password

Password that is used to encrypt and decrypt all other secrets

Stored in a separate file **outside** of Ansible repository, for example,

```
~/.ansible/vault_password
```

Configure Ansible to read master password from file (add this to **ansible.cfg**):

```
[defaults]
inventory = hosts
vault_password_file = ~/.ansible/vault_password
```

Ansible Vault: master password

```
$ ansible-vault encrypt_string p4ssw0rd

!vault |
    $ANSIBLE_VAULT;1.1;AES256
    66313066386566346263353134373763333466303863656133343232623162303562643366383562
    62363733646535343761646330636639373863393337653 20a31653838636366373338393962633 3
    66343661343564353934373239316538346666656436303035623635363661636631393261353966
    31666132343136396 40a6134313062356433333731646436666664373563306306166306431366662
    6161

Encryption successful
```

Ansible Vault: decryption

```
#group_vars/all.yaml
my_user: admin
my_password: !vault |
    $ANSIBLE_VAULT;1.1;AES256
    66313066386566346263353134373763333466303863656133343232623162303562643366383562
    62363733646535343761646330636639373863393337653 20a31653838636366373338393962633 3
    66343661343564353934373239316538346666656436303035623635363661636631393261353966
    31666132343136396 40a6134313062356433333731646436666664373563303061663064313666626
    6161

#roles/demo/tasks/main.yaml
- name: Print credentials (Unsafe! Use for demo only!)
  ansible.builtin.debug:
    msg: "User is {{ my_user }} and password is {{ my_password }}"
```

Questions?

ICA0002: IT Infrastructure Services

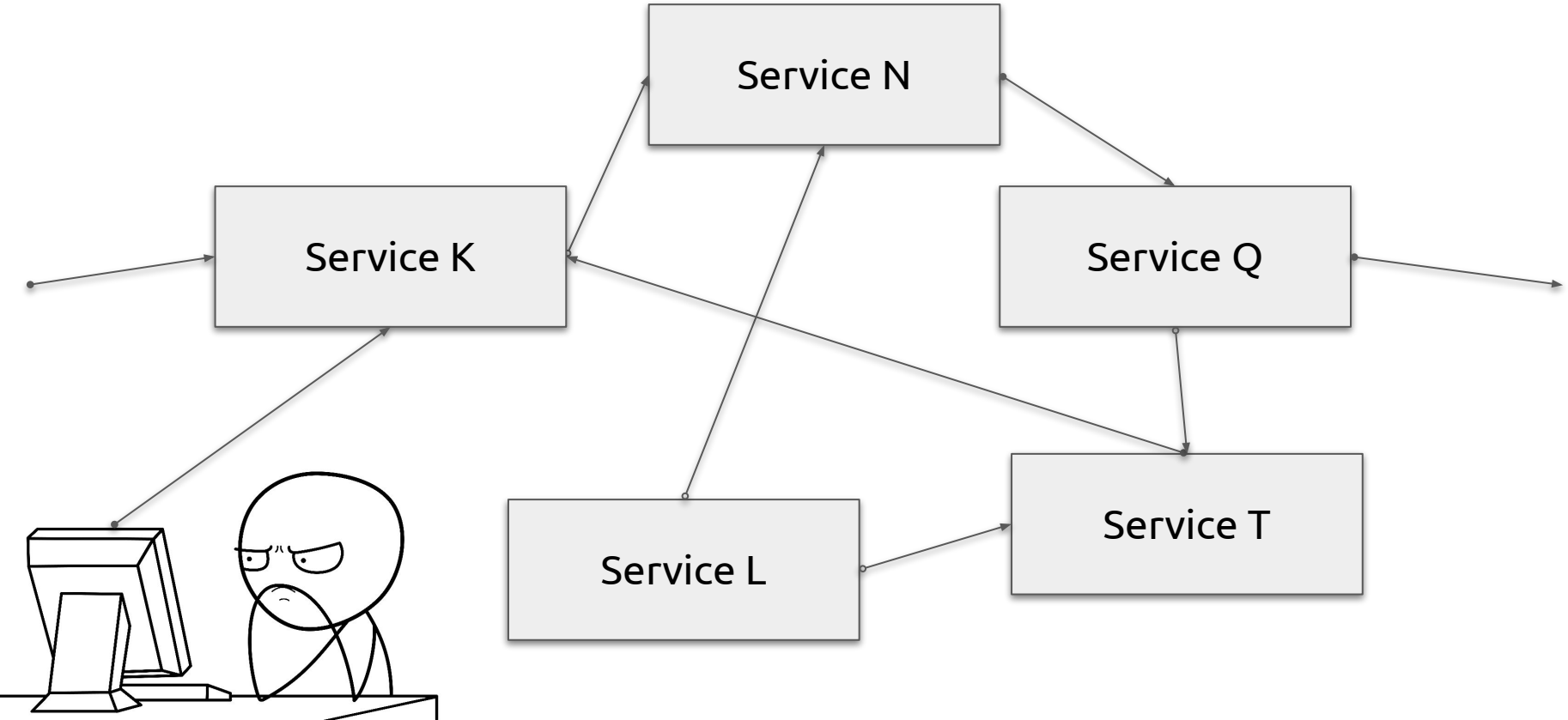
Troubleshooting

Roman Kuchin
Juri Hudolejev
2021

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: nginx-site.conf
  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
- http://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

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

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

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:

- No silver bullet exists -- every service has its own specifics
- Make sure that service **is actually running**
- Check configuration
  - Almost certainly a configuration **logic** problem
  - If the daemon has started -- configuration syntax is okay
- Check logs
  - Use verbose (debug) mode -- **temporary measure! Disable once done debugging!**
- Still not solved? Search for the solution in the Internet
  - 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"

Problem: service not working correctly

Logs:

- `/var/log/<service-name>`
- `/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`

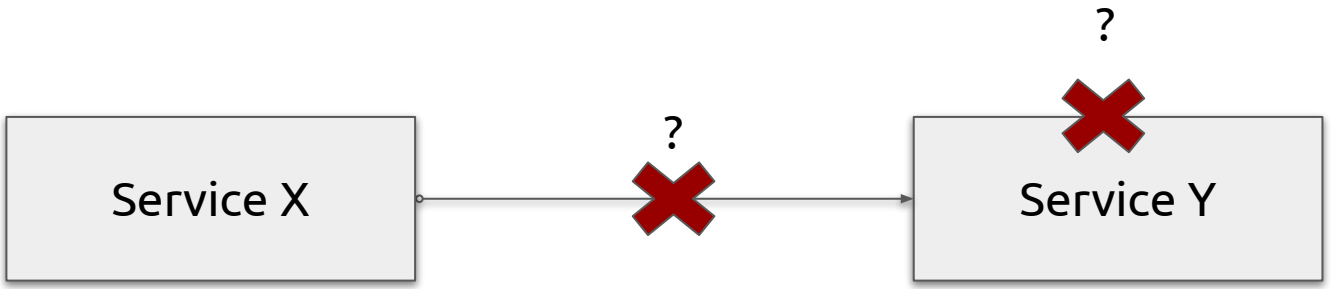
Nginx configuration file: `error_log /var/log/nginx/error.log debug;`

^^^ 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 X cannot connect to another machine's port N
- Service Y is listening on wrong interface and/or port

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 if ping to remote machine is blocked by firewall:

```
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
```

Service communication issues

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

```
sudo netstat -lnptu
```

Needs admin privileges to list process names

Alternative if **`netstat`** is not available:

```
sudo ss -lnptu
```

Service communication issues

Example command to list all services listening on TCP ports:

```
sudo netstat -lnpt
```

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

```
Proto  Recv-Q  Send-Q  Local Address    Foreign Address  State   PID/Program name
tcp    0       0       0.0.0.0:22       0.0.0.0:*        LISTEN  821/sshd
tcp    0       0       0.0.0.0:80       0.0.0.0:*        LISTEN  948/nginx
tcp    0       0       0.0.0.0:8080     0.0.0.0:*        LISTEN  948/nginx
```

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

```
tcp    0       0       127.0.0.1:3306   0.0.0.0:*        LISTEN  946/mysqld
```

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`
2. Ask Google, DuckDuckGo, Bing etc.
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?

Troubleshooting cheat sheet

Services and Processes

Process running?	`ps ax | grep <process>`
Service status:	`systemctl status <service>`
Config check:	`nginx -t`
Verbose mode:	`curl -v; ssh -vvv`
Service logs:	`/var/log/<service>/*` `journalctl -fu <service>`
System logs:	`/var/log/syslog` `journalctl -xn`
Help:	`<x> --help; man <x>; info <x>`

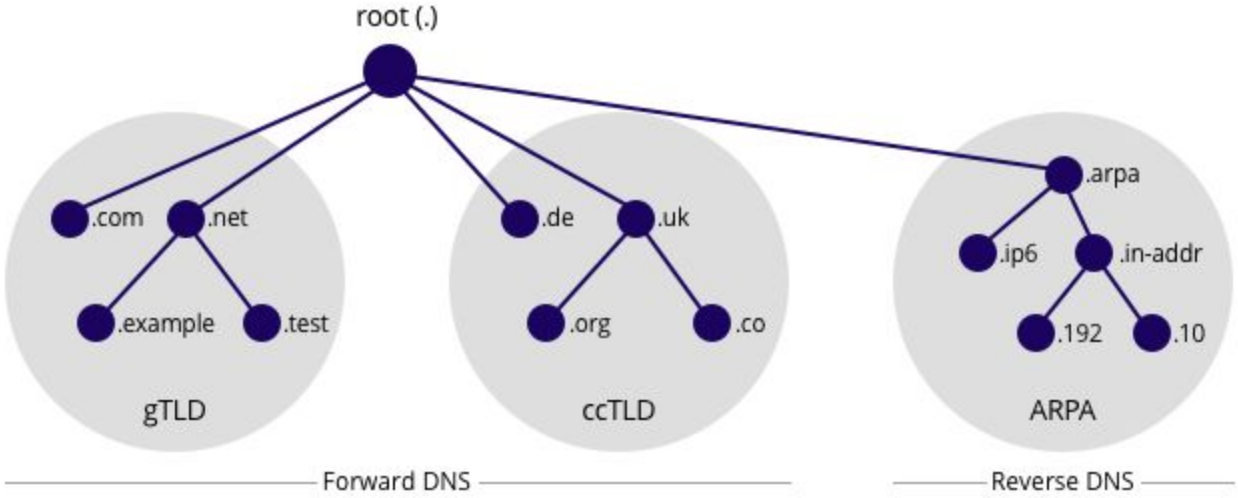
Connectivity

Network connected?	`ping 1.1.1.1`
IP address info:	`ip a`
Route info:	`route`
Reach other node:	`ping <ip>`
Probe remote port:	`nc -vz <ip> <port>`
Resolve hostname:	`host <hostname> [<ns>]`
Listening services:	`netstat -lnptu`

IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

DNS


root (.)
.com
.net
.example
.test
gTLD
.de
.uk
.org
.co
ccTLD
.arpa
.ip6
.in-addr
.192
.10
ARPA
Forward DNS
Reverse DNS

DNS record types

- A
- AAAA
- CNAME
- NS
- SOA
- MX
- TXT
- Many more: https://en.wikipedia.org/wiki/List_of_DNS_record_types

DNS troubleshooting

- https://isitdns.com/
- nslookup <name>
- host <ip or name> [dns-server]
- dig [type] <name> [@dns-server]

Bind9

Docs: https://kb.isc.org/docs/aa-01031

Jinja2 loops

https://jinja.palletsprojects.com/en/2.10.x/

Jinja2 online compiler: https://j2live.ttl255.com/

```
# vars
bgp_neighbors:
  -   10.0.2.15
  -   10.0.3.34
  -   10.2.0.45
  -   10.77.0.1
```

```
# template
router bgp 65500
 {% for n in bgp_neighbors %}
 neighbor {{ n }} remote-as 65500
 {% endfor %}
```

```
router bgp 65500
  neighbor 10.0.2.15 remote-as 65500
  neighbor 10.0.3.34 remote-as 65500
  neighbor 10.2.0.45 remote-as 65500
  neighbor 10.77.0.1 remote-as 65500
```

Jinja2 conditions

```
# vars_R1
bgp_neighbors:
  - 10.0.2.15
  - 10.0.3.34
  - 10.2.0.45
  - 10.77.0.1
```

```
# vars_R2
ospf_area: 0
```

```
# template
{% if bgp_neighbors %}
router bgp 65500
 {% for n in bgp_neighbors %}
 neighbor {{ n }} remote-as 65500
 {% endfor %}
{% endif %}
```

```
R1:
router bgp 65500
  neighbor 10.0.2.15 remote-as 65500
  neighbor 10.0.3.34 remote-as 65500
  neighbor 10.2.0.45 remote-as 65500
  neighbor 10.77.0.1 remote-as 65500
```

```
R2:
```

Ansible facts

```
RomanKuc-MBP:01-demo romank$ ansible-playbook test_ansible.yaml

PLAY [Ansible connection test] *************************************************

TASK [Gathering Facts] *********************************************************
ok: [my_vm]

TASK [test_connection : Ansible ping module] ***********************************
ok: [my_vm]

PLAY RECAP *********************************************************************
my_vm                      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
---
 - name: Ansible connection test
   hosts: my_vm
   gather_facts: no   ← default is yes
   roles:
     - test_connection
```

Ansible variables not from /vars folder

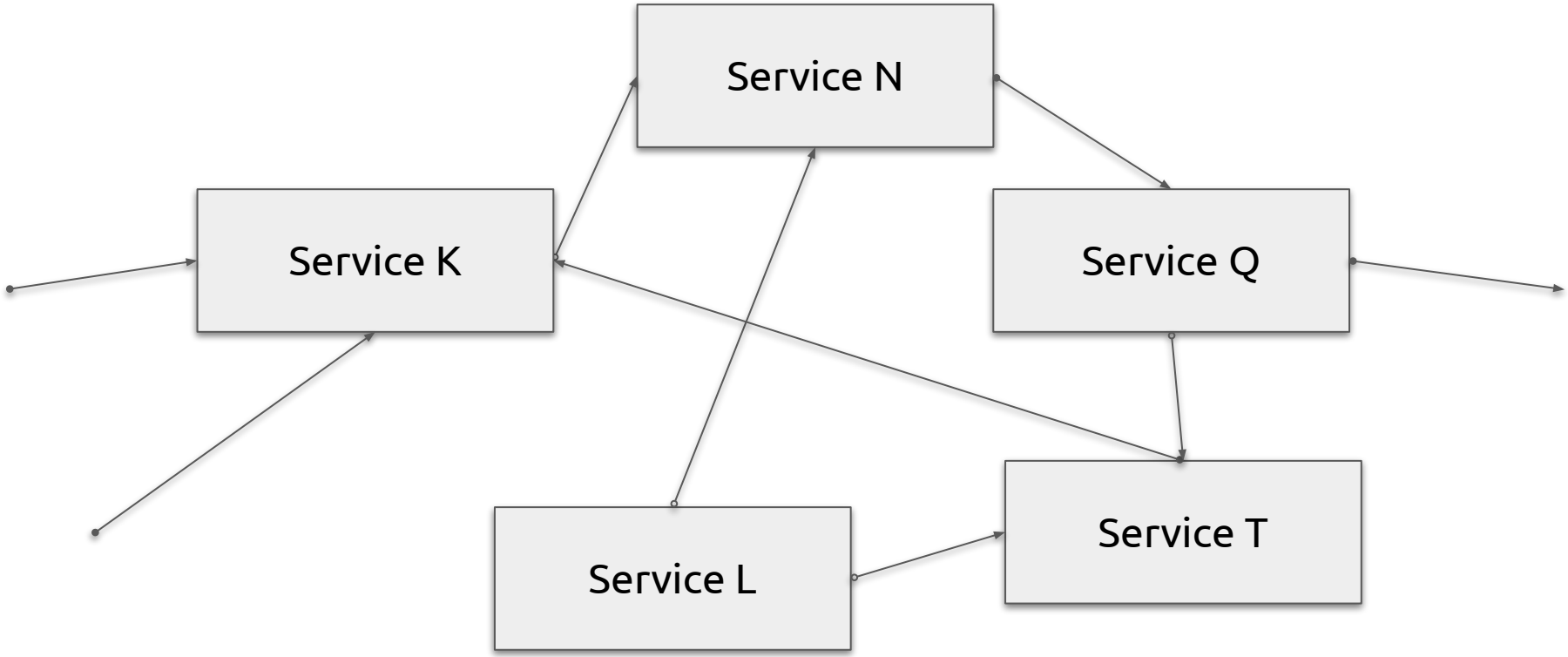
```
---
  - name: Show some variables
    hosts: my_vm
    gather_facts: no
    tasks:
      - debug:
            var: groups
      - debug:
            var: hostvars
      - debug:
            msg: This VM IP is {{ hostvars[‘my_vm’]['ansible_default_ipv4']['address'] }}

  - name: Show some variables
    hosts: my_vm
    tasks:
      - debug:
            msg: This VM IP is {{ hostvars[‘my_vm’]['ansible_default_ipv4']['address'] }}
```

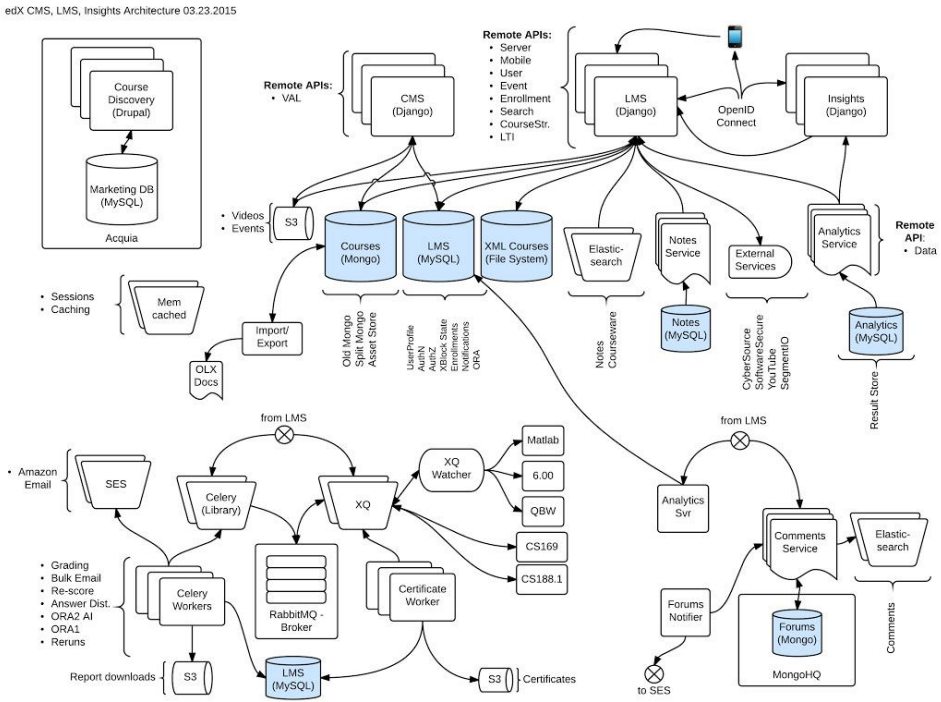
IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

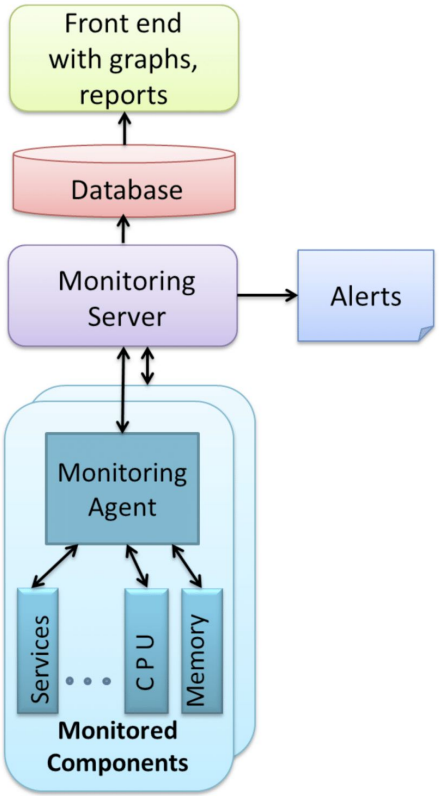
Small infrastructure



Not that small



Monitoring



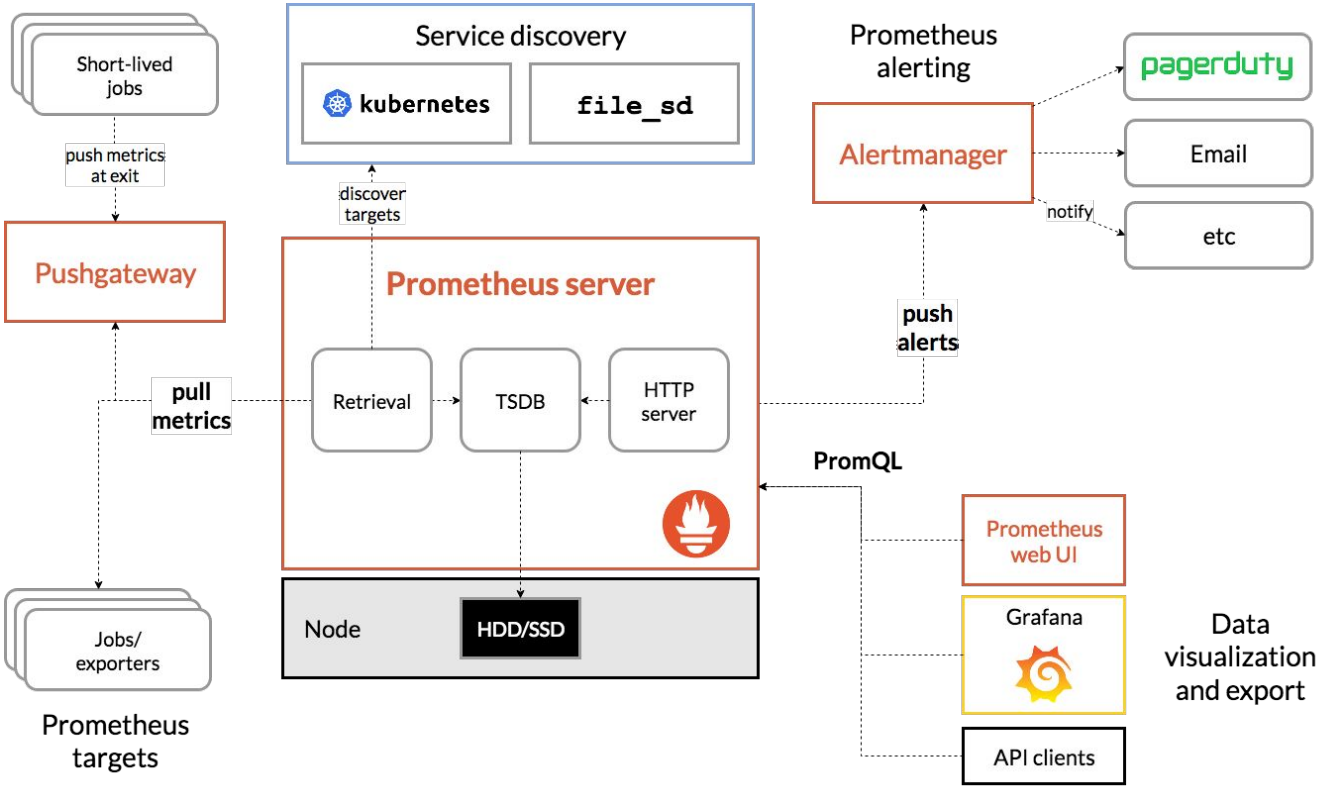
- Agent (server specific)
- Agentless (ssh, snmp)

Monitoring applications

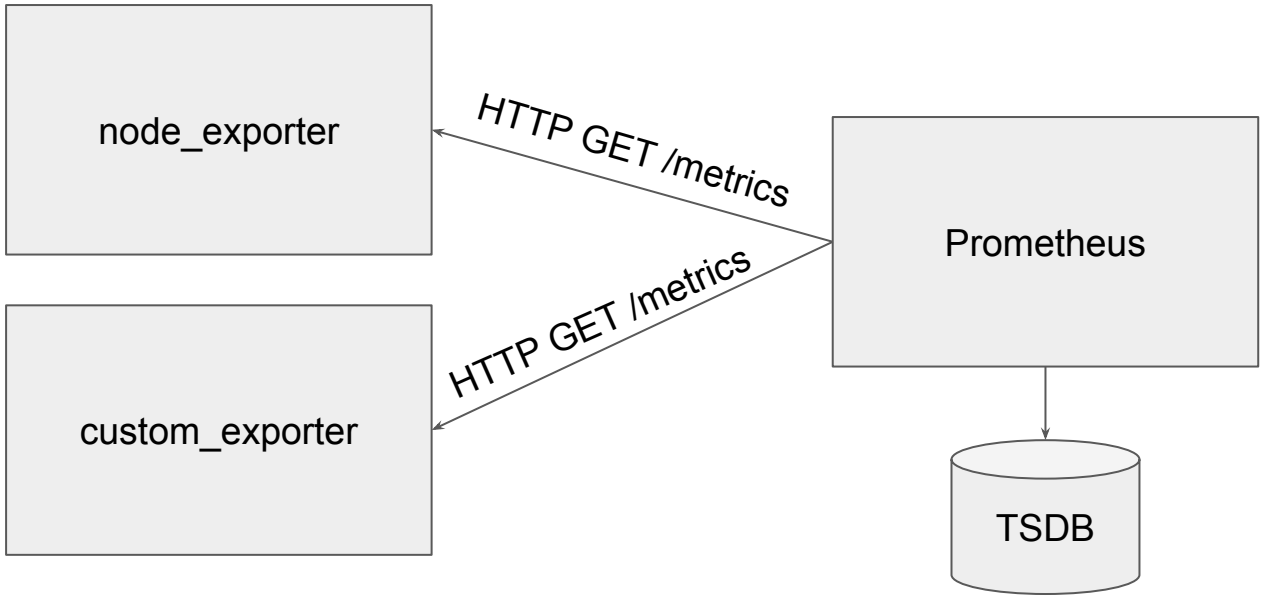
Long list here, doesn’t fit the screen :(

Prometheus

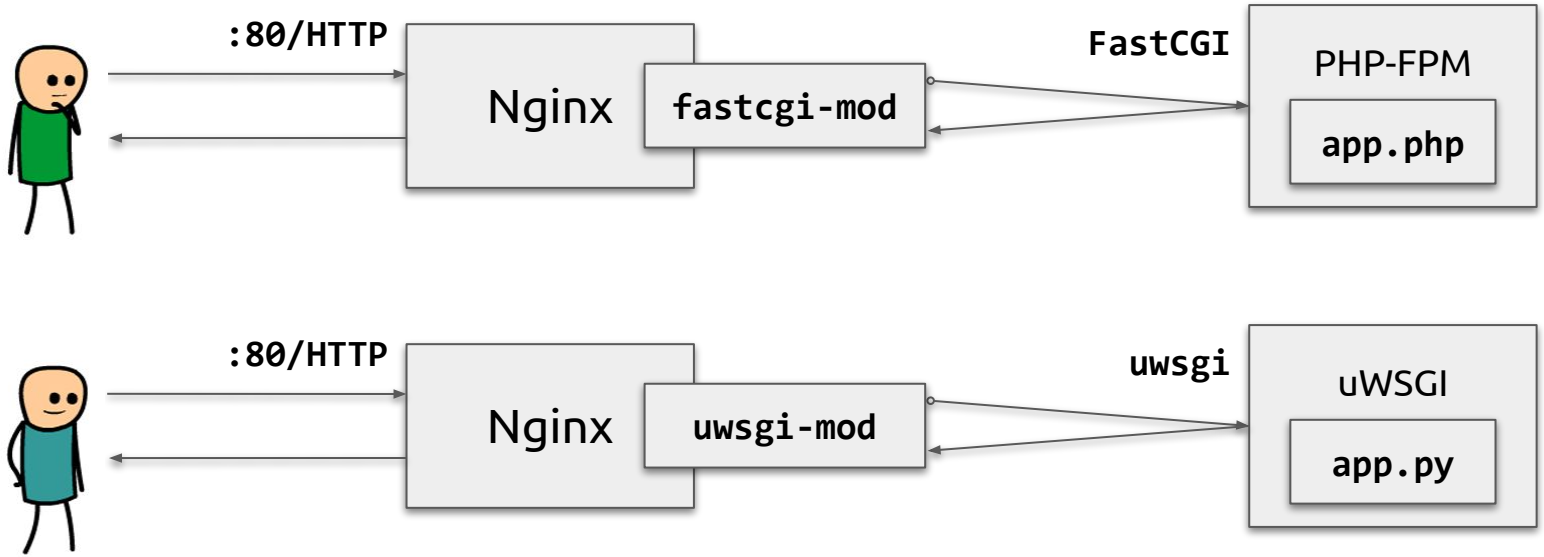
https://prometheus.io/docs/introduction/overview/



Prometheus

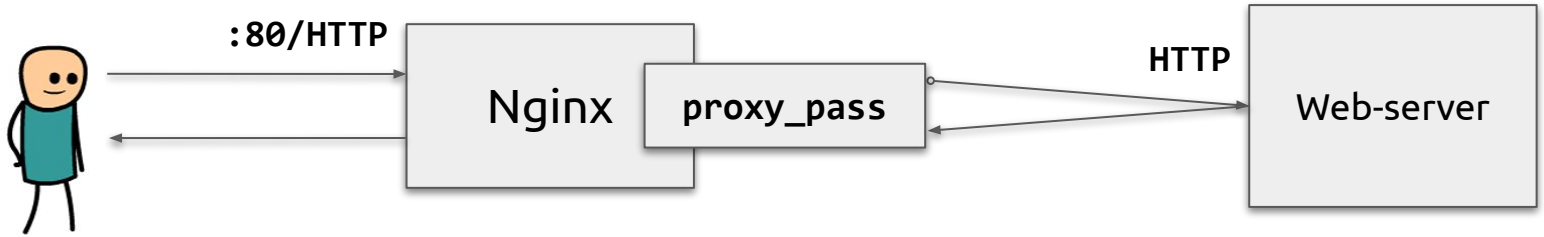

node_exporter
custom_exporter
HTTP GET /metrics
HTTP GET /metrics
Prometheus
TSDB

FastCGI and uwsgi examples



Script is executed by **application server** (FPM, uWSGI, Unicorn etc.)

Reverse proxy



Example config from docs.nginx.com:

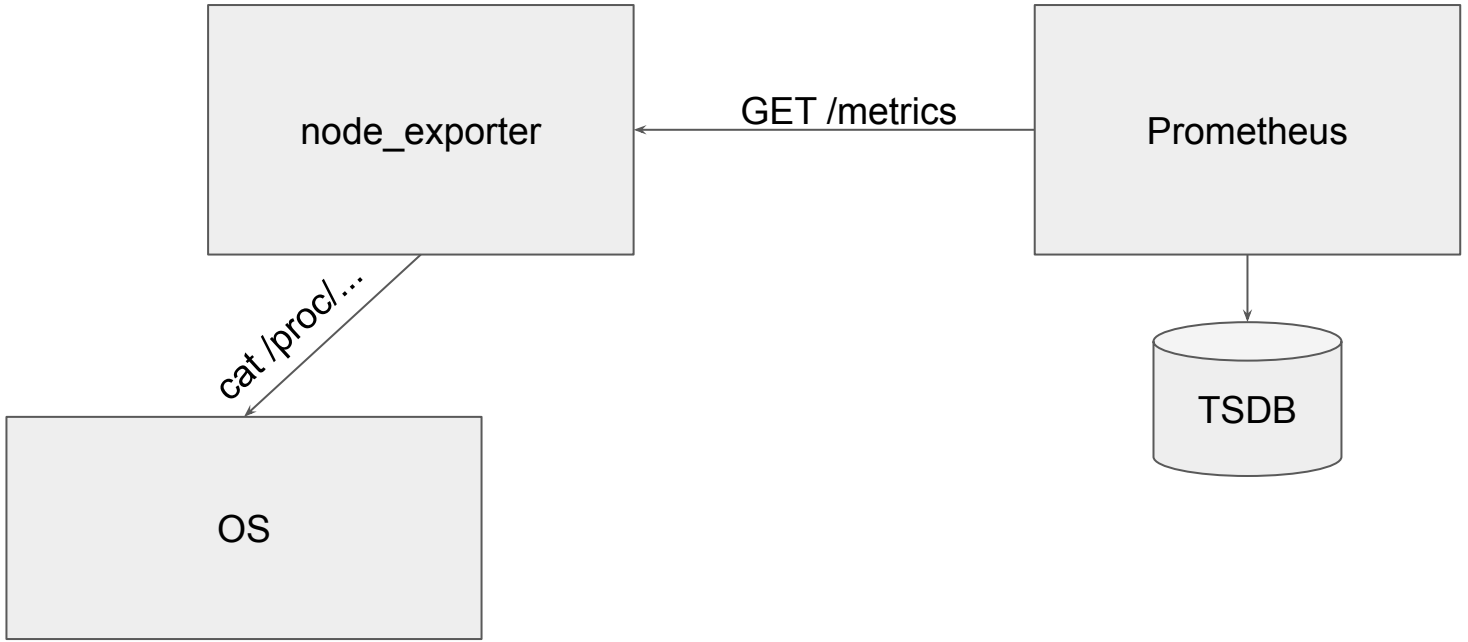
```
location /some/path/ {
        proxy_pass http://example.com/path;
}
```

IT Infrastructure services

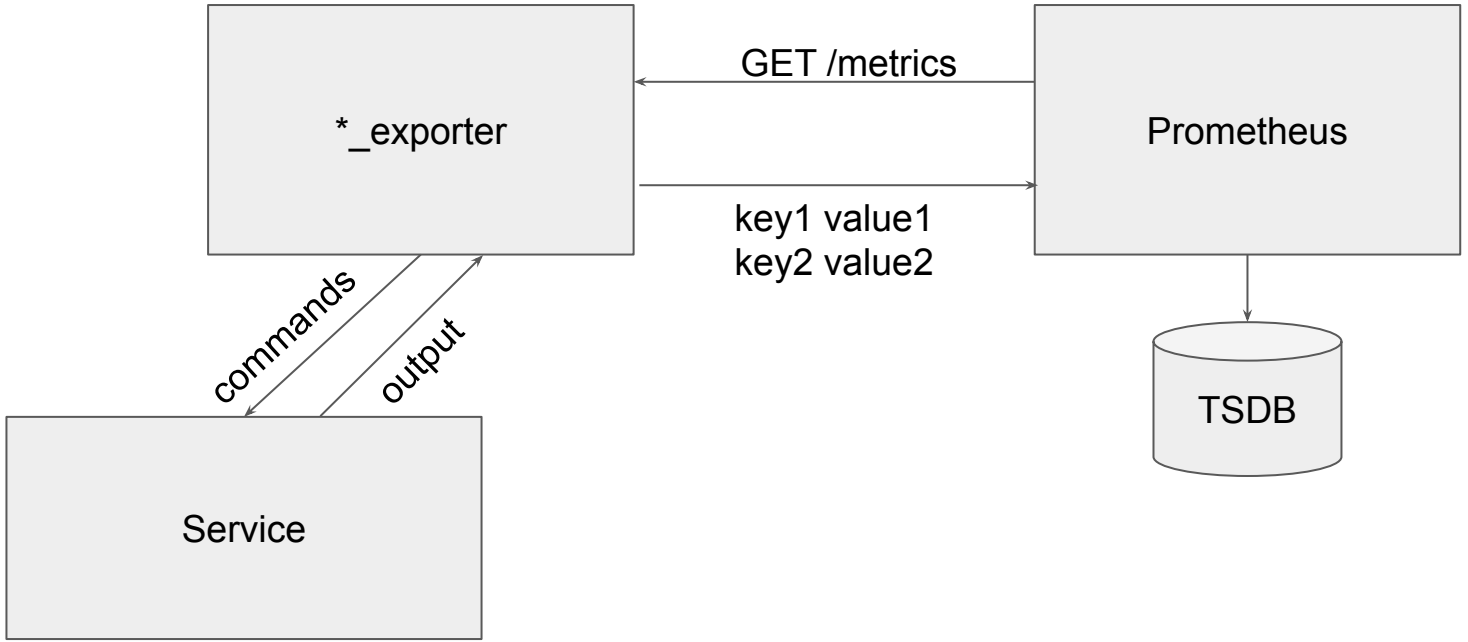
Roman Kuchin
Juri Hudolejev
2021

https://github.com/prometheus/node_exporter

Prometheus exporters

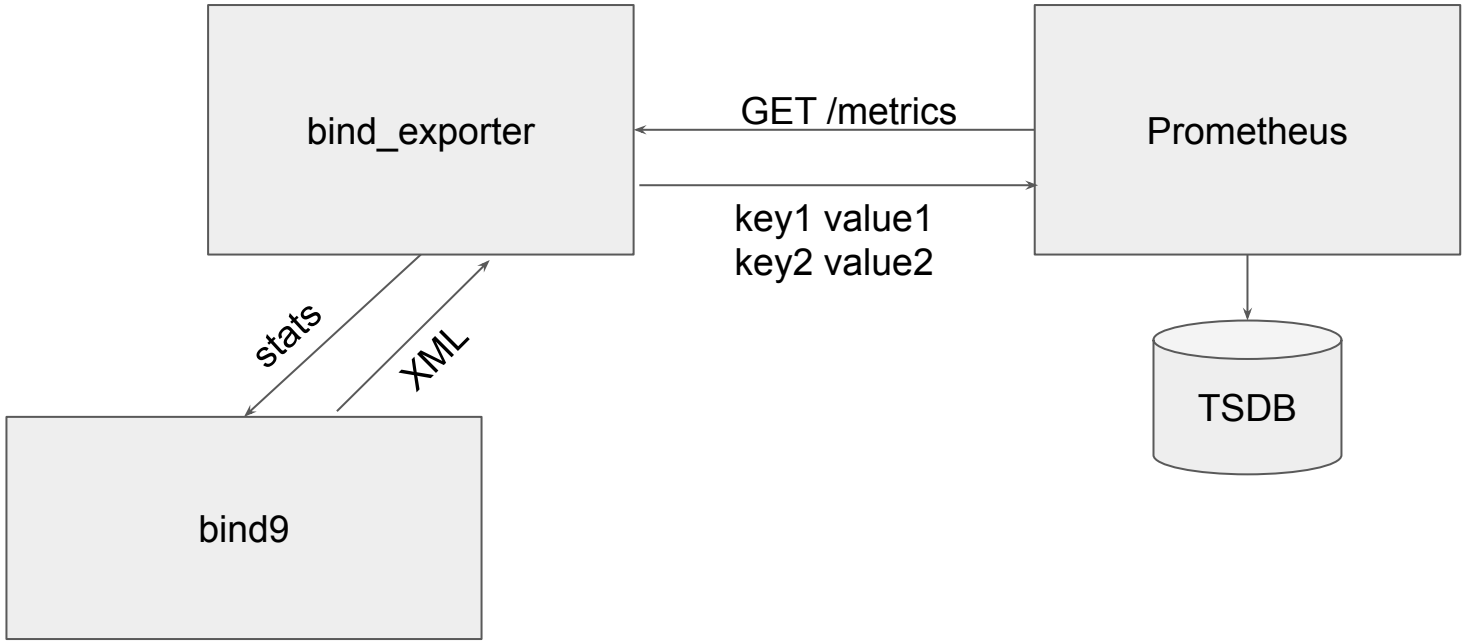


Prometheus exporters



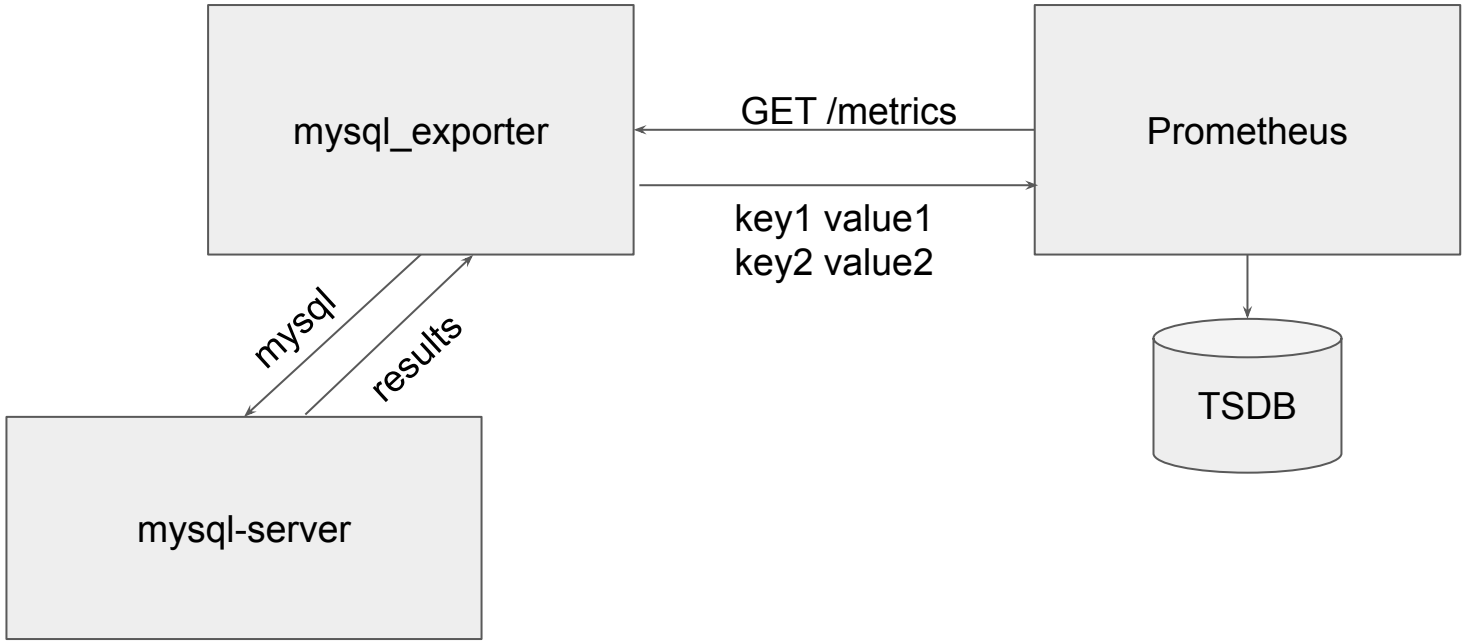
Prometheus exporters

https://github.com/digitalocean/bind_exporter



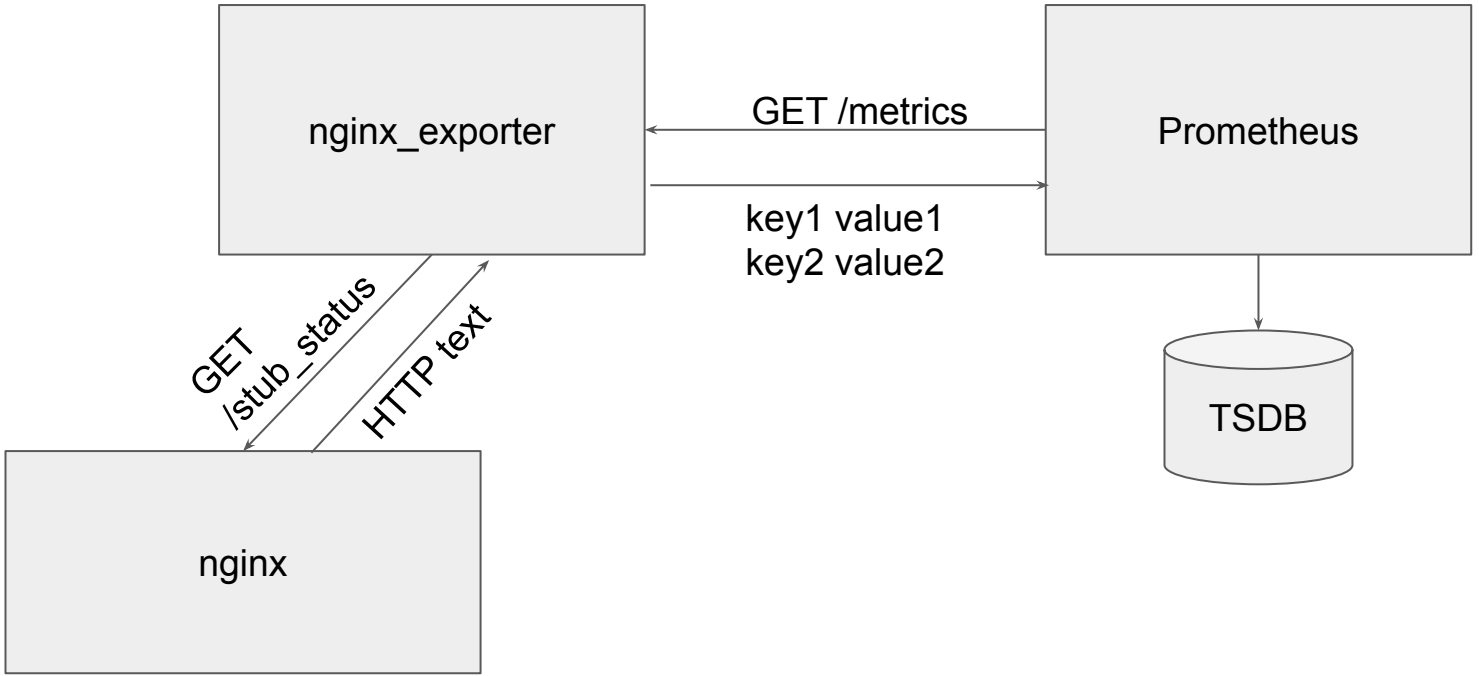
https://github.com/prometheus/mysqld_exporter

Prometheus exporters



https://github.com/nginxinc/nginx-prometheus-exporter

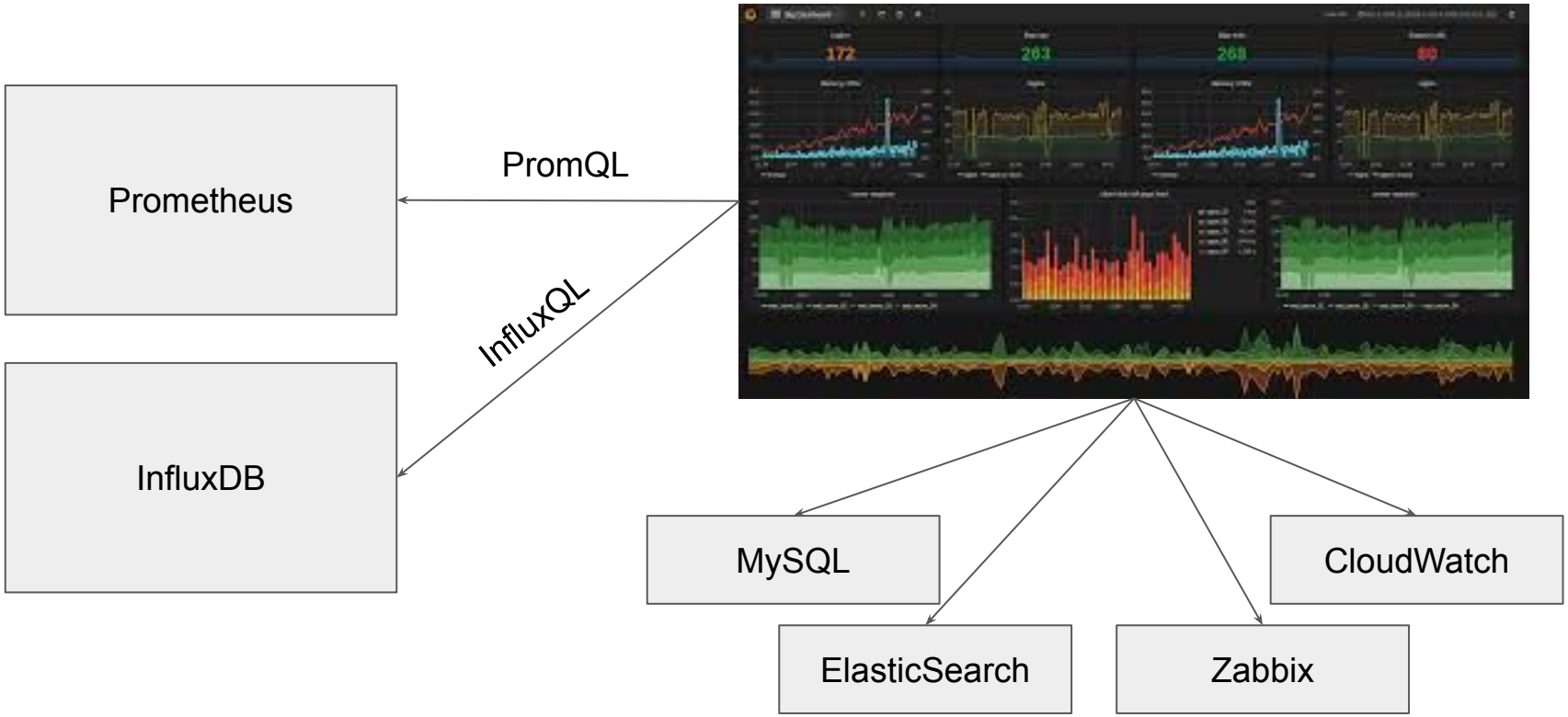
Prometheus exporters



Grafana

https://grafana.com/oss/grafana/

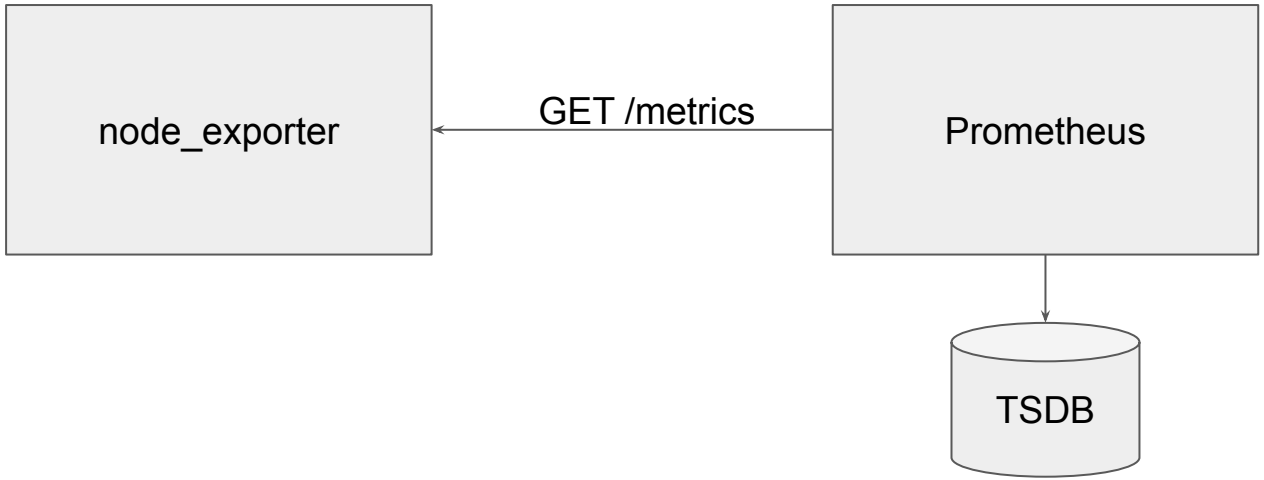
https://devopsart.tumblr.com/



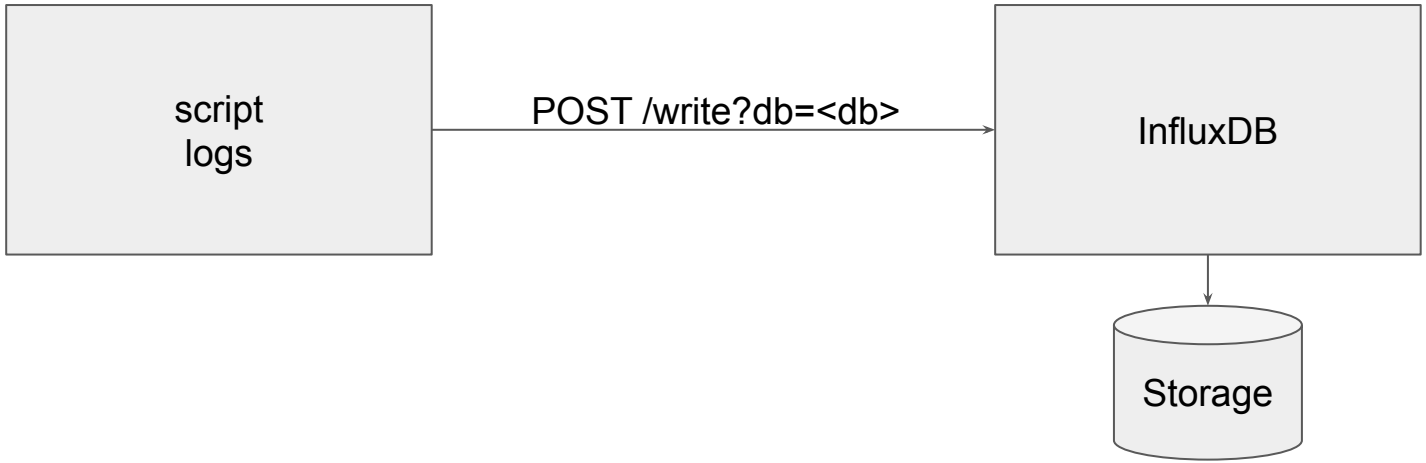
IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

Prometheus



InfluxDB

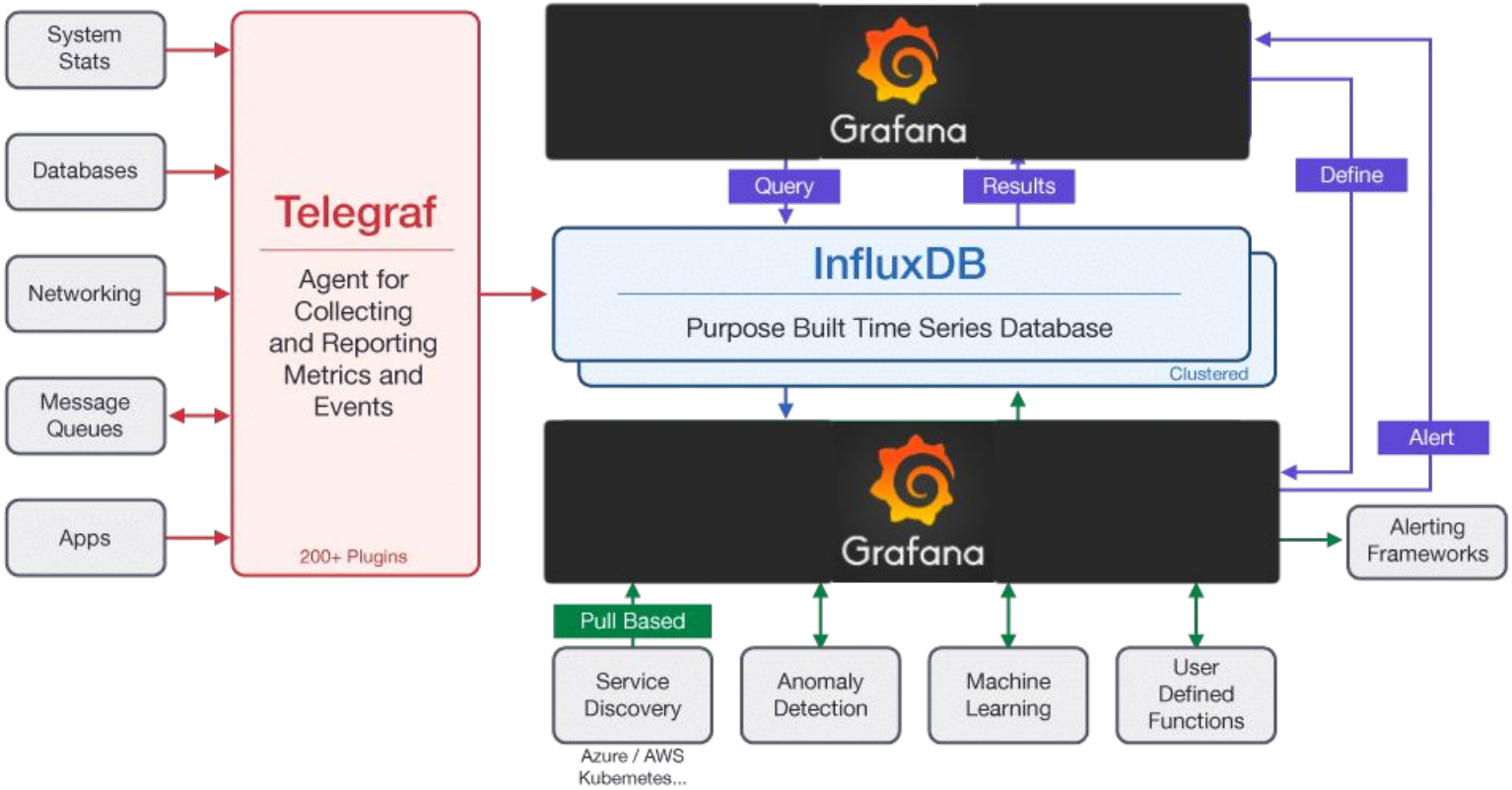


Request examples

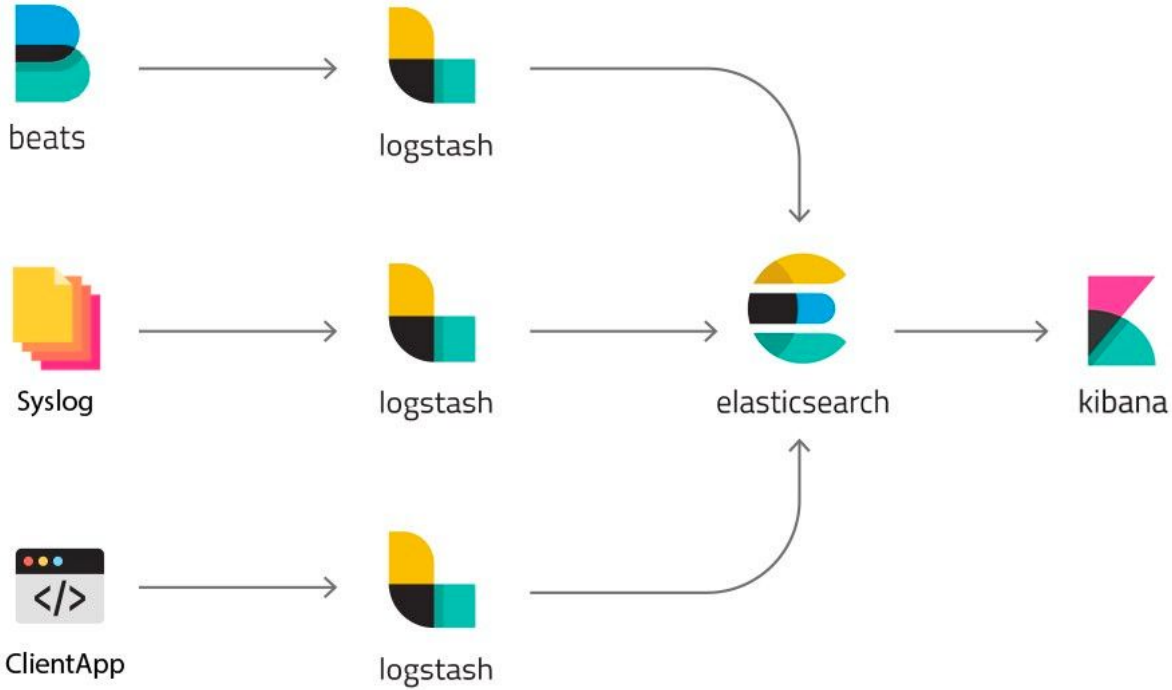
```
curl -i -XPOST -H 'Content-Type: text/plain' 'http://influxdb:8086/write?db=rtt' --data-binary
"latency google_latency=10”
```

```
show databases
use rtt
show measurements
select * from latency where time > now() - 5m
```

ELK vs TIG

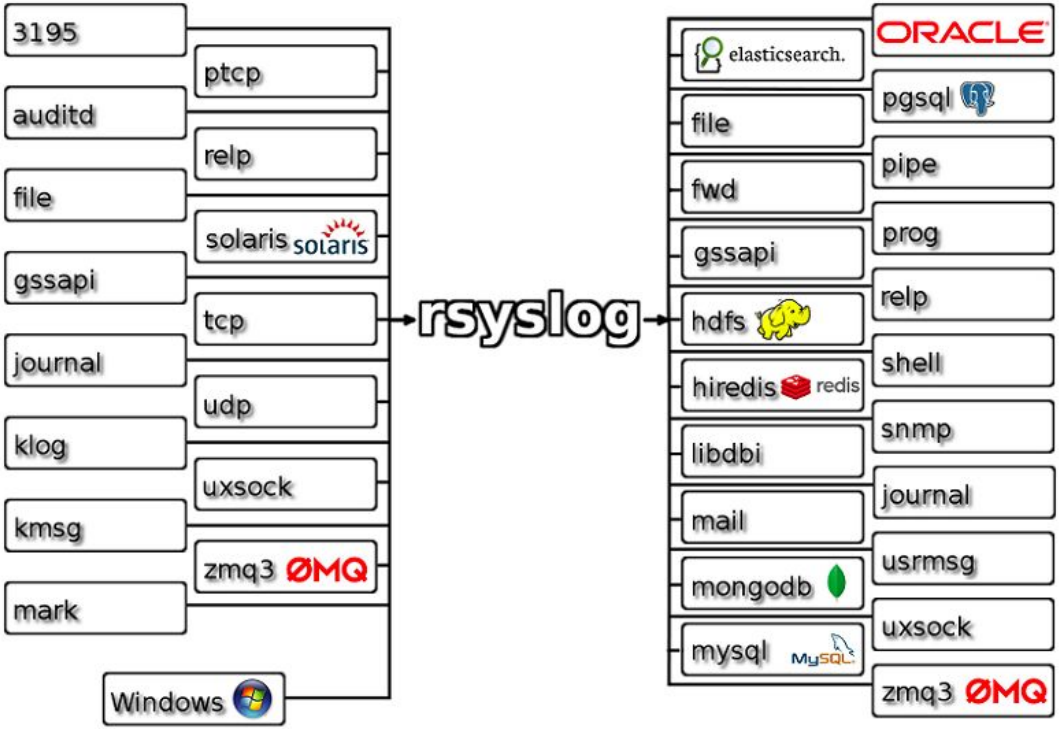

System Stats
Databases
Networking
Message Queues
Apps
Telegraf
Agent for Collecting and Reporting Metrics and Events
200+ Plugins
Grafana
Query
Results
Define
InfluxDB
Purpose Built Time Series Database
Clustered
Alert
Grafana
Alerting Frameworks
Pull Based
Service Discovery
Azure / AWS Kubemetes...
Anomaly Detection
Machine Learning
User Defined Functions

ELK vs TIG


beats
logstash
Syslog
logstash
elasticsearch
kibana
ClientApp
logstash

https://www.rsyslog.com/

Rsyslog



ICA0002: IT Infrastructure Services

Backups

Roman Kuchin
Juri Hudolejev
2021

What is a backup?

Backup

1. A copy of data that can be used to restore the original

2. A process of copying the data for restoration purposes

^^^ *these are fine if you want to explain your non-IT friend what a backups is*

Backup as an infrastructure service

Systematic established **process** of

- **Making** a copy of data that can be used to restore the original
- **Storing** that data for required period of time
- **Verifying** the usability of that data for restoration purposes
- **Documenting** the regular data restoration and disaster recovery plans
- **Verifying** the compliance of the documentation and the actual processes

Backup as an infrastructure service

In simple words, good backup

- Is created in time
- Is stored as long as needed
- Is available when needed
- Can be used to restore the needed service state

And every step of this process is documented and verified

Purpose of the backup

Ability to restore the **service** back to some certain state

- More than just restoring the data!

Important part of security policies

- Guarantees system availability and usability

Backup

1. Coverage
2. Frequency
3. Retention
4. Usability
5. Storage
6. Security
7. Documentation

Coverage: what to backup?

Ideally, everything

In real world, as much as your backup system can handle

Better to have multiple copies of the same data than no copies at all

Coverage: what to backup?

Data backups

- Must have in any case
- Usually universal format, can be restored to a different system
- Storage can be optimized (incremental / differential / deduplication)
- Creation and restore can be time consuming

Examples: SQL dump, JSON export, XML export, …

Coverage: what to backup?

File backups

- Why export data if we can backup the original file?
- Storage can still be optimized
- Creation and restore might require less time
- System being restored must support the file format

Example: cannot restore MySQL 5.6 files backup to MySQL 5.7 server

Coverage: what to backup?

Virtual machine and/or filesystem snapshots

- Creation and restore might require even less time
- May require machine or service downtime to create a backup
- Higher storage and bandwidth requirements

Example: your own machine (laptop, VirtualBox) snapshots

Coverage: what to backup?

Virtual machine images

- Why would you need them?
- The image can always be downloaded from the internet, right?

What if

- Public image is not available?
- You use custom modified images?

Frequency: how often to backup?

Examples: every night, every Sunday

More often:

- More backup storage is required
- More resources (CPU, memory) are needed to create and verify the backups
- Less service availability (downtimes or reduced HA during backup creation)

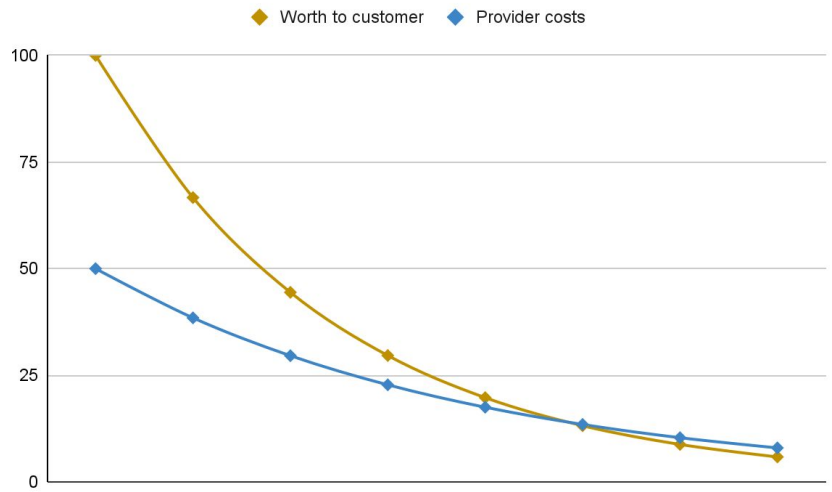
Less often:

- RPO: recovery point objective
- Time interval between two backups -- acceptable data loss

Retention: how long to store the backup?

Backup storage has its limits

Data value reduces over time



Retention: how long to store the backup?

Backup storage has its limits

Data value reduces over time

Backup retention must be longer than interval between backups!

-- Captain Obvious

How many versions of the same data are stored?

Ideally, as many versions as possible

In real world:

```
version_count =  frequency * retention
```

Usability: is the backup usable?

Ensure that you can restore the service from the backup

- Was the backup created **in time**?
- Is the stored backup **readable**?
- Is the stored backup **enough** to restore the service?
- Could the service be restored **to the needed state**?

Solution: test backup restores regularly

Usability is likely the most important feature of the backup

Storage: where to store backups?

3-2-1 rule:

At least 3 copies of the data -- 2 storage types onsite -- 1 copy offsite

Separate cloud or physical location -- preferably both!

Monitoring physical conditions: temperature, humidity etc.

Storage: where **not** to store backups?

"We have redundant hardware (RAID), so the backups are automated!"

"We are keeping backup on the same server to save costs!"

"We are keeping backup in the same datacenter, so the restore is really fast!"

"We are using cloud platforms, they do the backups themselves!"

What is wrong with these approaches? What are the risks?

Security: how to protect backups?

Good backup contains all the data needed to restore the system.

Backup should be protected as severely as the live system itself.

Store the backup in the safe place:

- Physical copies -- vault
- Online copies -- encrypt the transport, storage -- whatever possible

Documentation: how to describe backups?

Three main documents:

1. Backup SLA (service level agreement)
2. Backup restore plan
3. Technical documentation for backup operators

Related documents:

- DR (disaster recovery) plan

Documentation: how to describe backups?

Backup SLA:

1. Coverage -- what is being backed up
2. RPO -- how often backups are made, acceptable data loss
3. Retention -- how many versions of the backups are stored and for how long
4. Usability checks -- how and how often is the backup usability verified
5. Restoration criteria -- how to decide when to restore from the backup
6. RTO -- how long will it take to restore the service

(RTO -- recovery **time** objective)

RPO vs. RTO

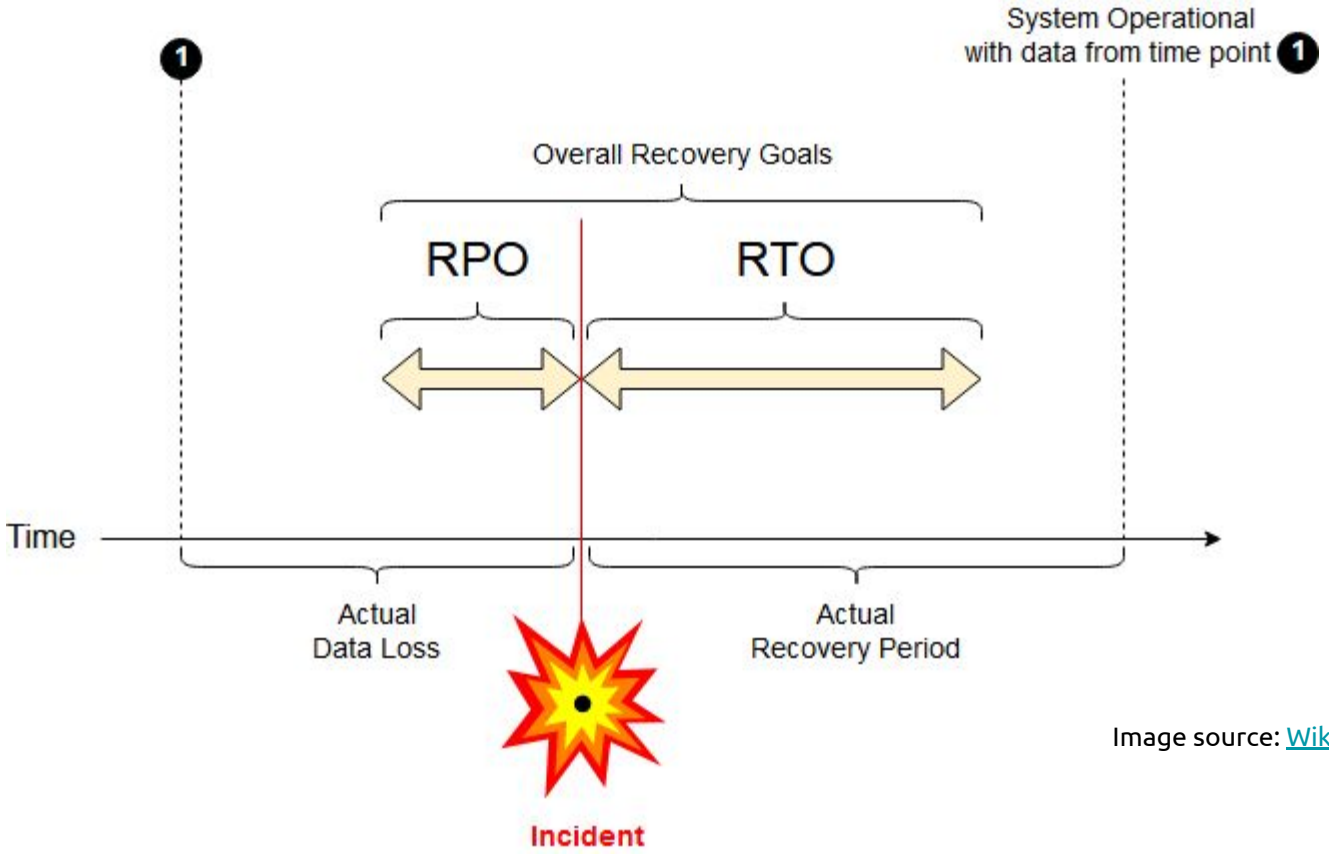


Image source: Wikipedia | CC BY 4.0

Documentation: how to describe backups?

Backup restore plan:

- Documented process of the backup restoration
- Explains in details who does what, how and when
- Ideally any employee with needed permissions could to restore the service

Documentation: how to describe backups?

Technical documentation for operators:

- How to check if the backup was created
- How to verify the backup
- How to install, configure and administer the backup system

Backup is your last resort!

Build your systems so that you would not need backups:

- High availability and redundancy
- Multiple copies of the data

But still do backups properly!

Never underestimate the importance of the backups

Questions?

What is **not** a backup?

Backup vs. archive

Backup:

- Main focus: fast service restore to a certain state
- Retention period: weeks, sometimes months

Archive:

- Main focus: data consistency + low cost of ownership
- Retention period: years, sometimes decades
- Ideally retyped to a new media from time to time

Backup vs. configuration management

Configuration management

- Helps to restore the service faster
- Does **not** replace the backup!

Goal: restore the **service**, not the data

Some components are easier to recreate than to recover

Better to have a backup unused than not have it all

Only wimps use tape backup; real men just upload their important stuff on ftp, and let the rest of the world mirror it ;)

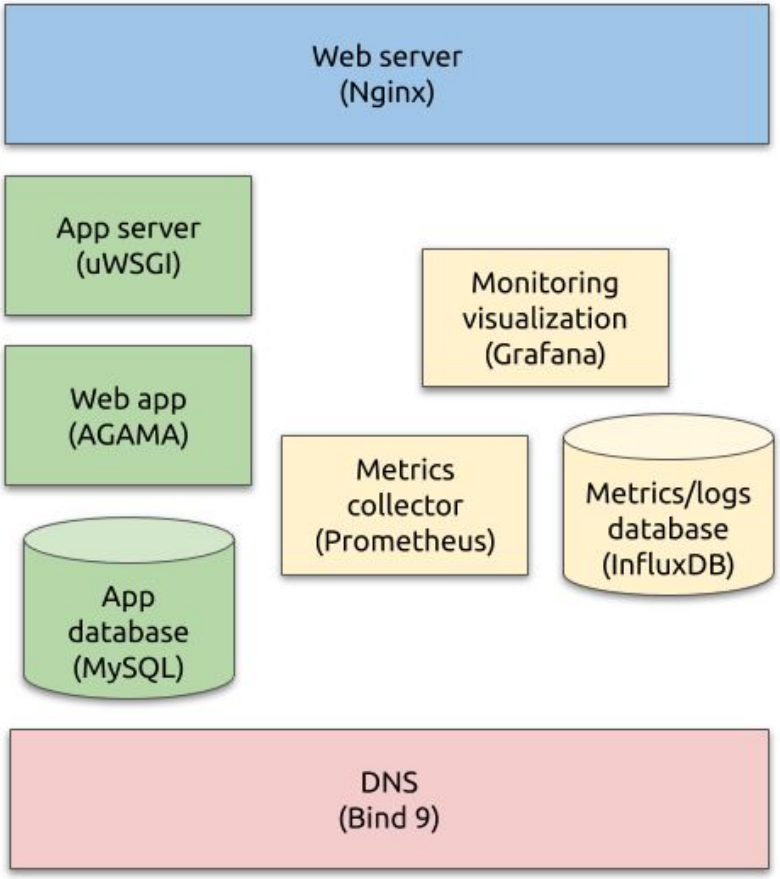
-- Linus Torvalds

ICA0002: IT Infrastructure Services

Backups again

Roman Kuchin
Juri Hudolejev
2021

Current setup

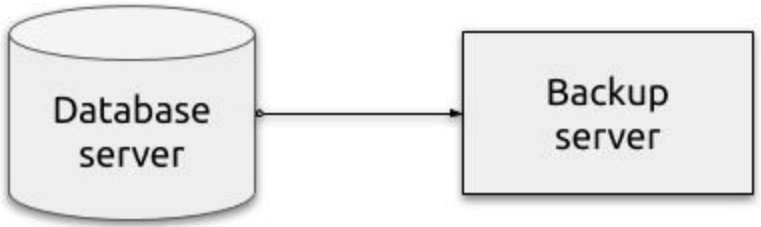


Reminder: backup as an infrastructure service

Systematic established process of

1. Making a copy of data that can be used to restore the original
2. Storing that data for required period of time
3. Verifying the usability of that data for restoration purposes
4. Documenting the regular data restoration and disaster recovery plans
5. Verifying the compliance of the documentation and the actual processes

Demo: backup process in details



Demo time!

Demo backup

1. Coverage: MySQL dumps (data/file backups)
2. Frequency: full backup -- weekly, incremental backup -- daily
3. Retention: (not covered)
4. Usability: only tried manual backup restore
5. Storage: only on-site storage part covered
6. Security: (not covered)
7. Documentation: you did it on previous lab, right? :)

Questions?

ICA0002: IT Infrastructure Services

High availability

Roman Kuchin
Juri Hudolejev
2021

Availability

Probability that the system works as required

Different ways to count:

- May include downtimes for a planned maintenance, or
- May exclude them

Part of system reliability engineering

Mentioned in the service SLA

- Example SLAs: AWS, Atlassian, Google Workspace, Slack

Availability

Let's say that some system may be unavailable less than 1% of the time

System is available at least 99% of the time

In the observed period it makes less than

- 36 seconds per hour (in average)
- 14 minutes 24 seconds per day
- 1 hour 41 minutes per week and so on

Availability

Let's say that some system may be unavailable less than 1% of the time

System is available at least 99% of the time -- **"2 nines"**

In the observed period it makes less than

- 36 seconds per hour (in average)
- 14 minutes 24 seconds per day
- 1 hour 41 minutes per week and so on

"Nines"

Nines	Availability	Allowed downtime			
		Daily	Weekly	Monthly	Yearly
1	90%	<2.5 hours	<17 hours	~3 days	<40 days
2	99%	<15 min	<2 hours	<8 hours	<4 days
3	99.9%	<2 min	~10 min	<45 min	<9 hours
4	99.99%	<10 sec	~1 min	<5 min	<1 hour
5	99.999%	<1 sec	<10 sec	<30 sec	<6 min
...	...	...	...	...	...

Availability

How to get more "nines":

- Less unexpected downtimes
- Faster recovery

90% cannot be achieved without backups and documentation

99% cannot be achieved without monitoring and pager duty

99.9% cannot be achieved without automation and redundancy

Availability

How to get more "nines":

- Less unexpected downtimes
- Faster recovery

90% cannot be achieved without backups ✓ and documentation ✓

99% cannot be achieved without monitoring ✓ and pager duty

99.9% cannot be achieved without automation ✓ and redundancy

Redundancy

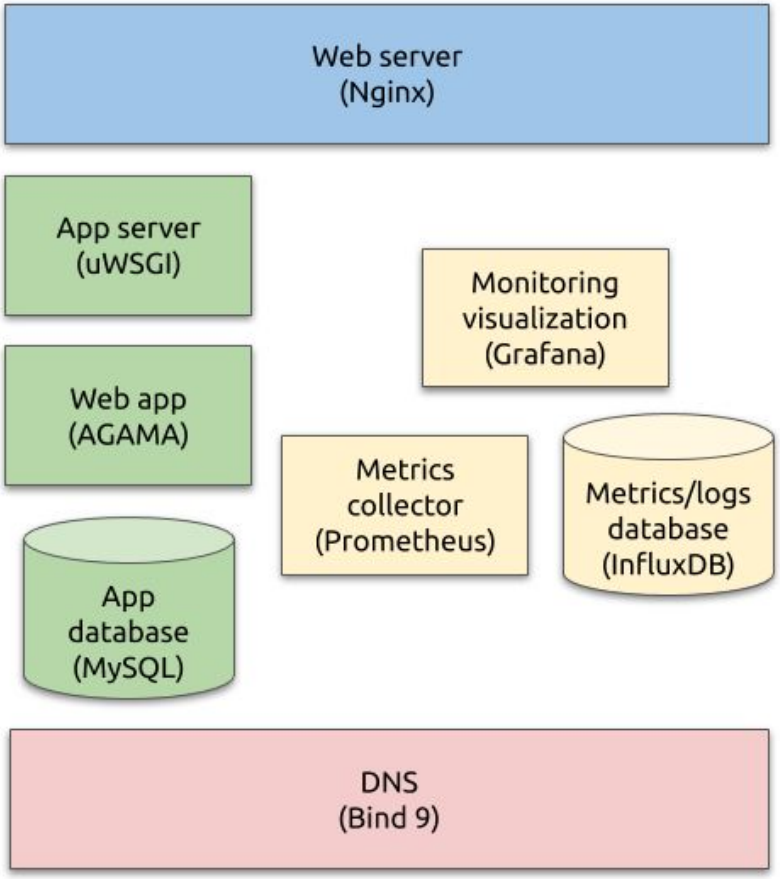
Duplication of critical components of the system

N+1

2N

2N+1

Current setup



MySQL redundancy

Replication

- 1 primary node (write enabled; also called 'source', 'master')
- 1 or more read only replica nodes (also called 'slaves')

Clustering

- 3 or more primary nodes (all write enabled)
- May also have read only replicas

MySQL replication

Available on any modern MySQL flavor (Oracle, Percona, MariaDB) out of the box

Load balancing: separate instances(s) for intensive reads, reports, backups etc.

- Application must support it!

Manual failover in case of primary node failure

- Can be automated but needs additional work

Single write enabled primary node -- single point of failure

More info: https://dev.mysql.com/doc/refman/8.0/en/replication.html

MySQL asynchronous replication

Client sends query to MySQL primary node

MySQL primary node:

- Accepts and executes the transaction
- Writes the transaction to the binary log if this transaction changes the data (INSERT, UPDATE, DELETE queries)

MySQL replica:

- Connects to the primary node to get the latest changes from binary log
- Updates local copy of the data with the changes from the primary node

MySQL asynchronous replication

Faster query execution:

- Synchronization is done in a separate thread, does not block transactions

Does not depend on network delays:

- Allows geographically distributed clusters

Data integrity is not always guaranteed

- Data loss occurs if primary fails before the replica has pulled the updates

MySQL semisynchronous replication

Client sends query to MySQL primary node

MySQL primary node:

- Accepts the transaction
- Notifies replicas if this transaction changes the data
- Waits until at least one replica confirms it has accepted the change
- Commits the transaction and sends the result to the client

More info: https://dev.mysql.com/doc/refman/8.0/en/replication-semisync.html

MySQL semisynchronous replication

Better data integrity guarantees

Longer query runs:

- Primary waits for data replication before commiting the transaction
- Query execution time depends on network delays (RTT)
- Query execution time depends on the fastest replica performance

Demo!

MySQL replication challenges

Your application must support it

- Different endpoints for different SQL queries
- Microservice architecture

Failover detection and automation

- Home-made scripts
- Corosync + Pacemaker
- External services: DNS, Consul/Etcd
- SQL proxies: HAProxy, ProxySQL etc.

GitHub case: https://github.blog/2012-09-14-github-availability-this-week

MySQL clustering

Several primary write enabled nodes: 3, 5, 7 etc.

- Should be odd number to avoid split-brain

Synchronous data replication

Widely used applications:

- MySQL NDB
- MySQL Galera Cluster / Percona XtraDB Cluster / MariaDB Galera Cluster

MySQL clustering challenges

Performance: cluster is as slow as its slowest member

Dependency on network delays

- Geographically distributed clusters can be a problem

Simultaneous distributed write queries: hotspots and deadlock errors

Database engine and schema restrictions

- InnoDB and XtraDB only
- Known LOCK and DELETE query limitations

MySQL redundancy

Replication

- Cheaper way to balance the load and simplify the **recovery** after the failure

Clustering

- More expensive and complex way to **prevent** the system downtimes

Can be used together in complex systems with high availability requirements

More reading

How MySQL replication works, in details:
https://dev.mysql.com/doc/refman/8.0/en/replication-implementation.html
and subpages

Galera cluster pros and cons:
https://www.percona.com/blog/2013/05/14/is-synchronous-replication-right-for-your-app

So how many "nines"
is high availability?

High availability

"High" in HA does not mean a certain number of nines

High availability is a characteristic of a system, and a principle of system design

- Better uptime
- Tolerate more failures
- Faster recovery

Don't create complex systems.

Create simple systems that scale well.

HA does not replace backups!

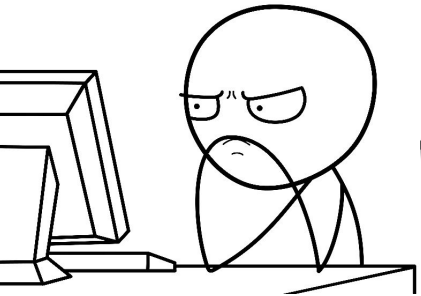
Questions?

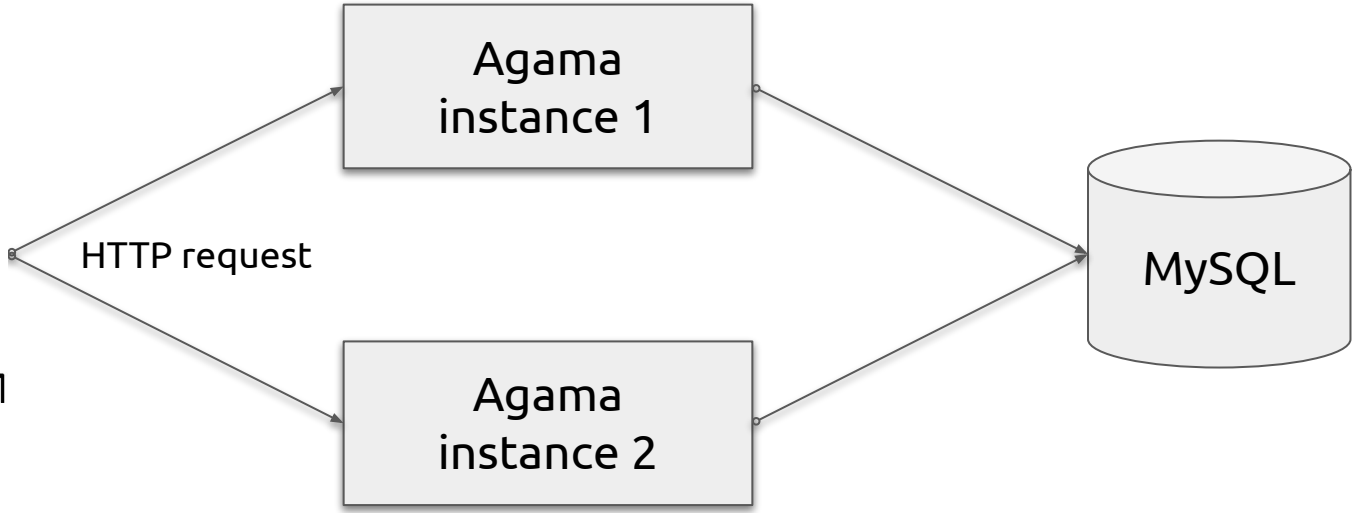
IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

Everything should die!

Webserver redundancy: round-robin DNS






```
$ host www.mydomain
www.mydomain has address 11.22.33.81
www.mydomain has address 11.22.33.82
```

Webserver redundancy: round-robin DNS

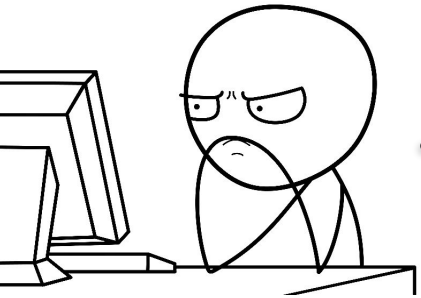
Pros:

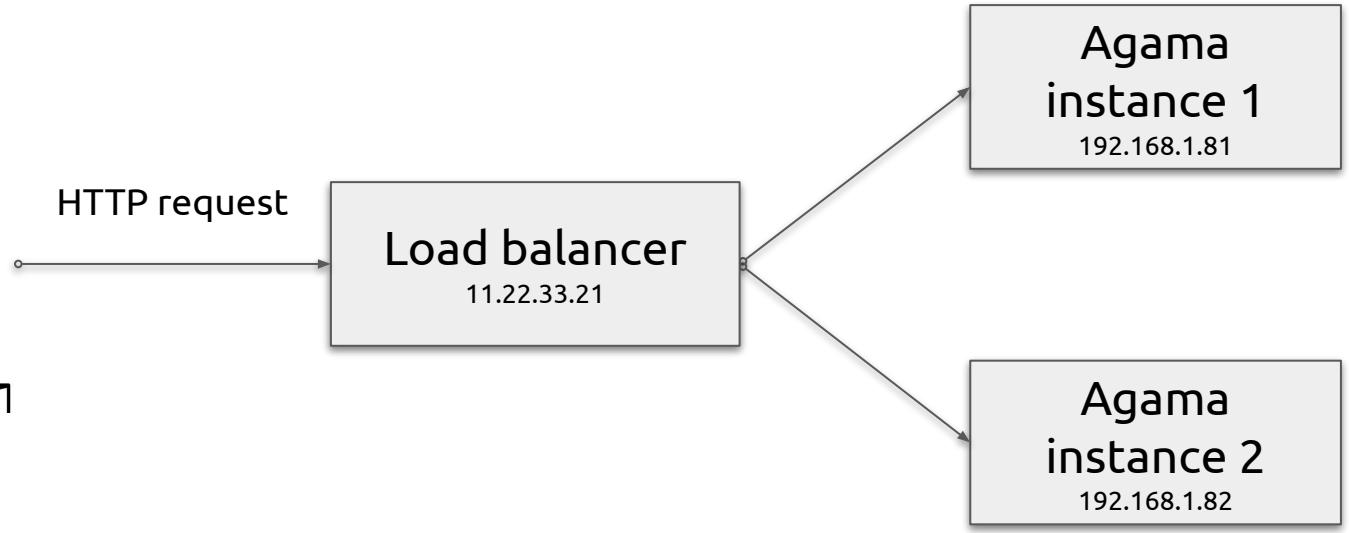
- Easy to implement
- No additional software/hardware required

Cons:

- DNS caching
- No service health checks (IP may resolve but the service itself is down)
- **The client should support this**

Server-side load balancing






```
$ host www.mydomain
www.mydomain has address 11.22.33.21
```

Server-side load balancing

Pros:

- Service health and utilization checks
- Load **balancing**, not just distribution

Cons:

- Additional software, hardware or cloud service required
- Another component that needs to be highly available

Server-side load balancing

Hardware

- Appliances by F5 Networks, A10, NetScaler etc.

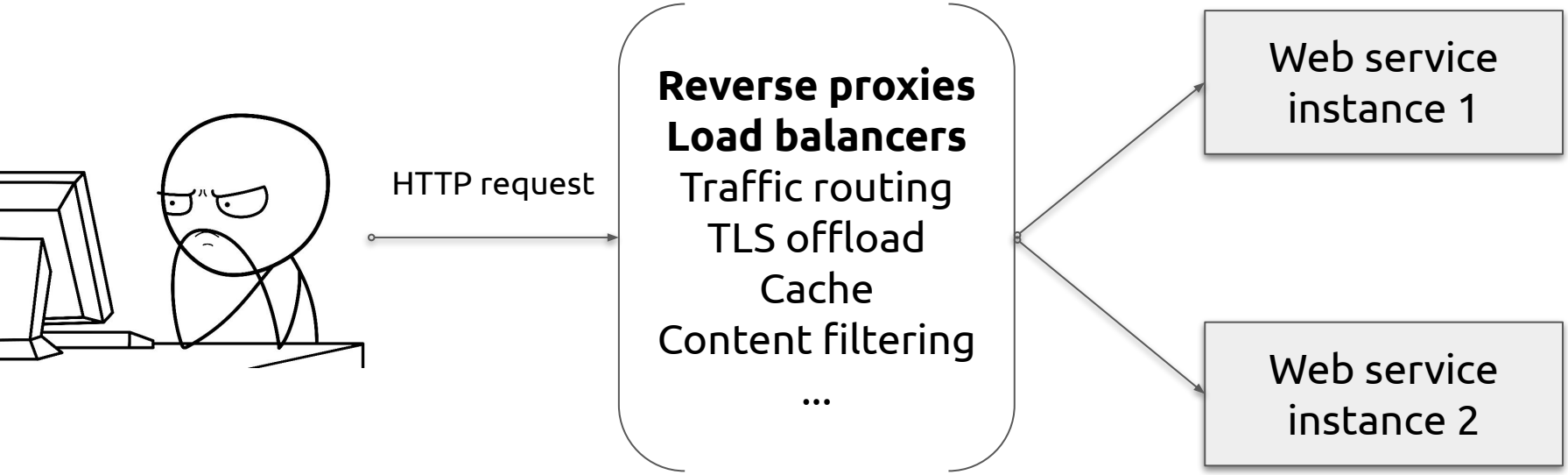
Cloud based

- Most cloud providers and IaaS platforms
- Every CDN

Software

- HAProxy, Nginx, Træfik etc.

Server-side load balancing and more



Server-side load balancing: HAProxy vs Nginx

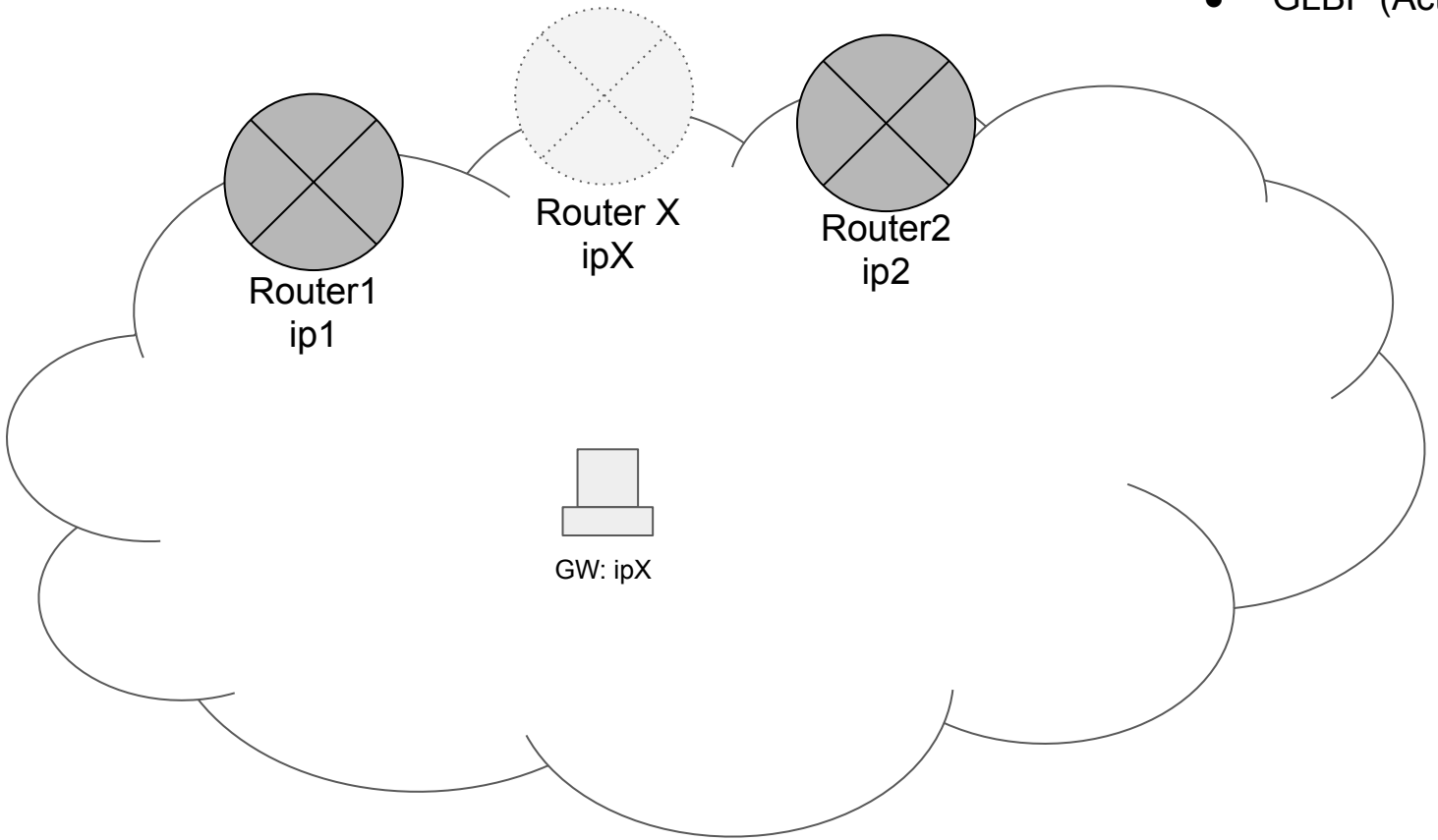
- **Both are fine for simple load balancing**
- If you want something more advanced, check the docs and choose wisely
- HAProxy is solely a proxy, Nginx is also a web server

Possible advise (very holy-war'ish)

- If you **already use** Nginx and need a simple load balancer, go on with Nginx
- If you don't have any load-balancing capable service yet, start with HAProxy

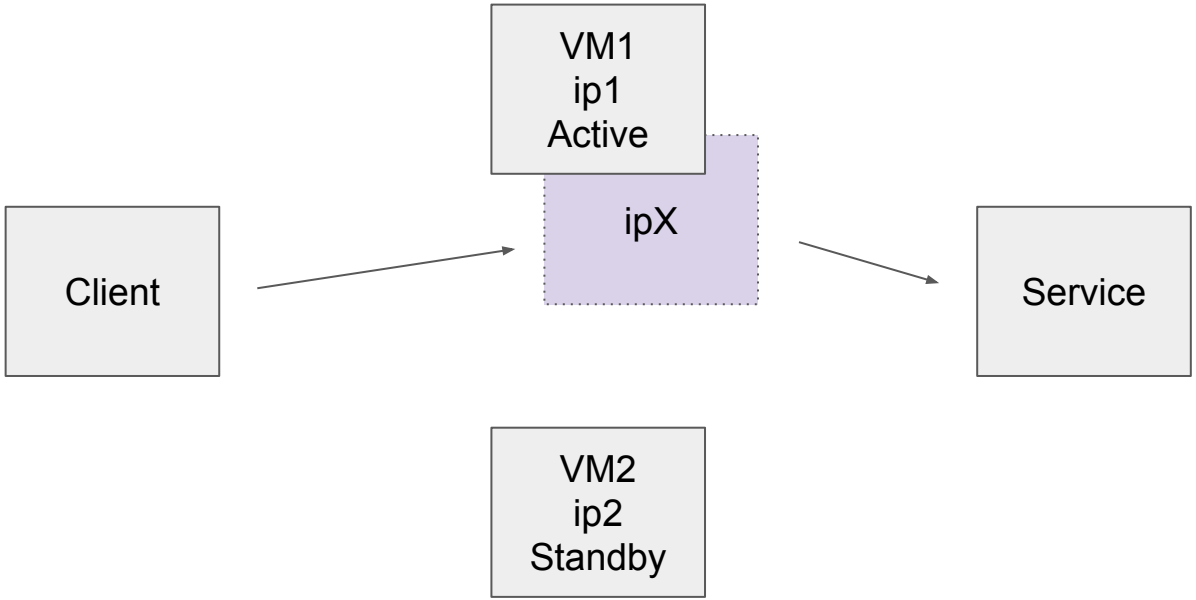
FHRP

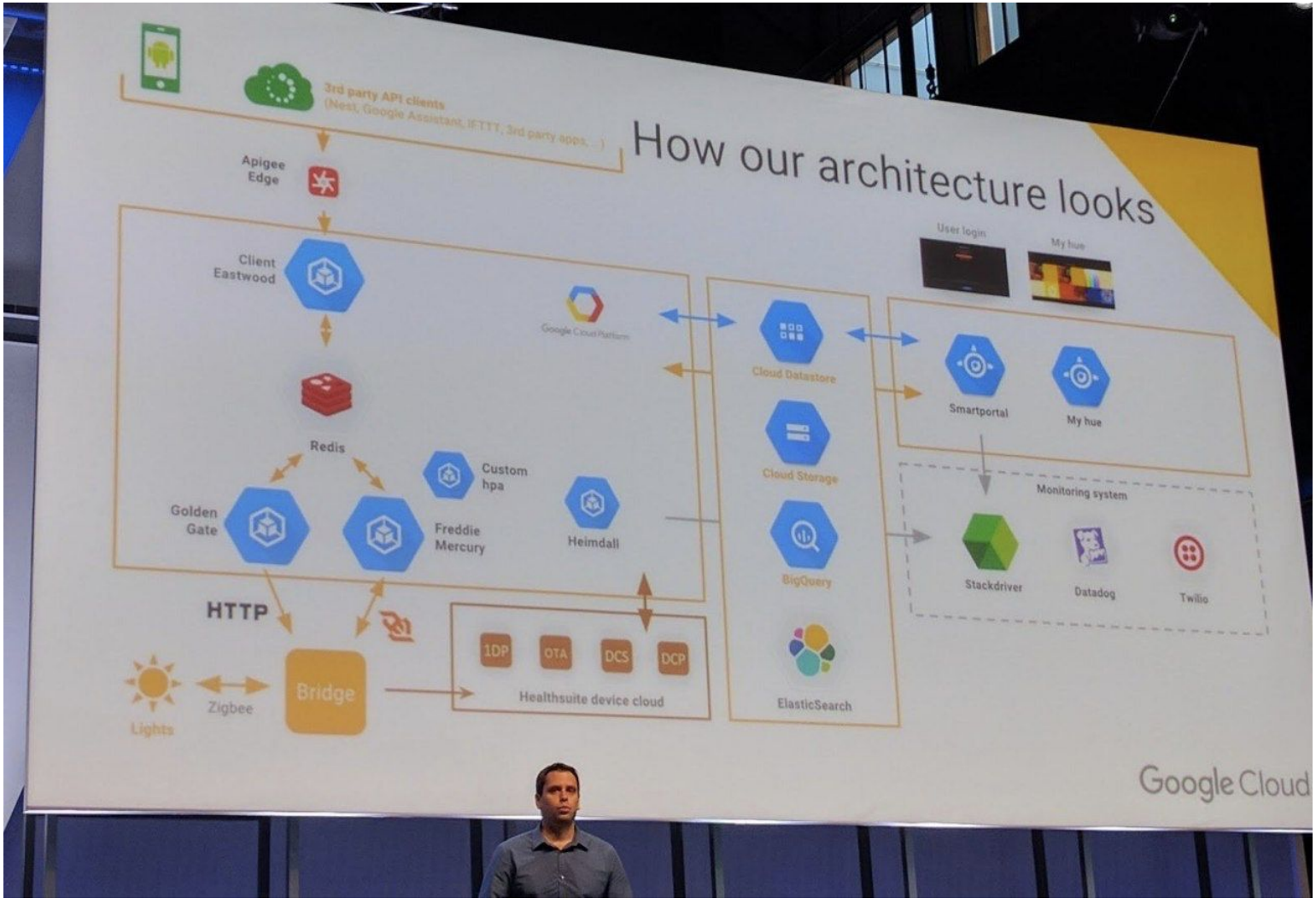
- HSRP (Active/Standby)
- VRRP (Active/Standby)
- GLBP (Active/Active)



Keepalived (VRRP)

https://www.keepalived.org/manpage.html



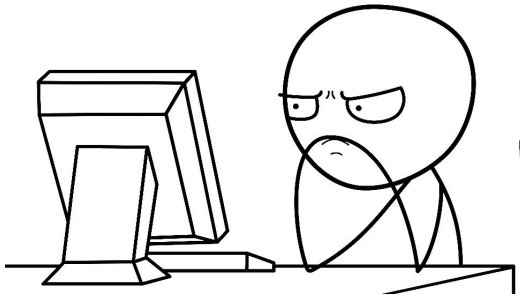

How our architecture looks
3rd party API clients
(Nest, Google Assistant, IFTTT, 3rd party apps, ...)
Apigee Edge
Client Eastwood
Redis
Golden Gate
Freddie Mercury
Custom hpa
Heimdall
HTTP
Bridge
Zigbee
Lights
1DP
OTA
DCS
DCP
Healthsuite device cloud
Google Cloud Platform
Cloud Datastore
Cloud Storage
BigQuery
ElasticSearch
User login
My hue
Smartportal
My hue
Monitoring system
Stackdriver
Datadog
Twilio
Google Cloud

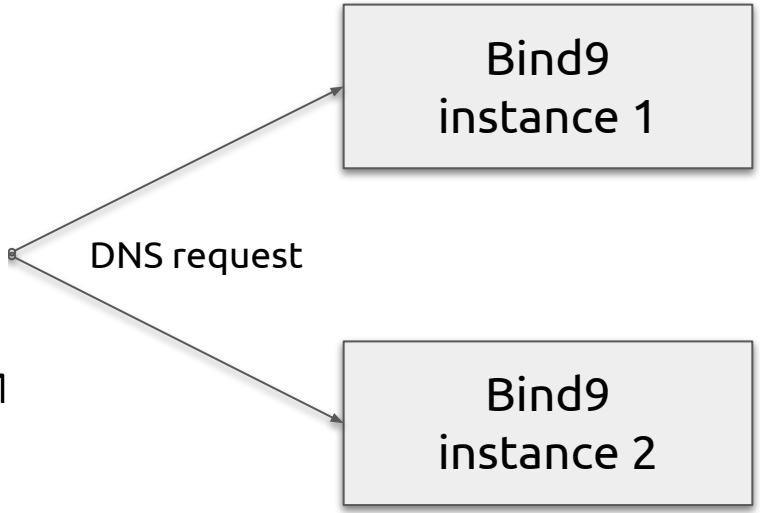
IT Infrastructure services

Roman Kuchin
Juri Hudolejev
2021

Everything should die!

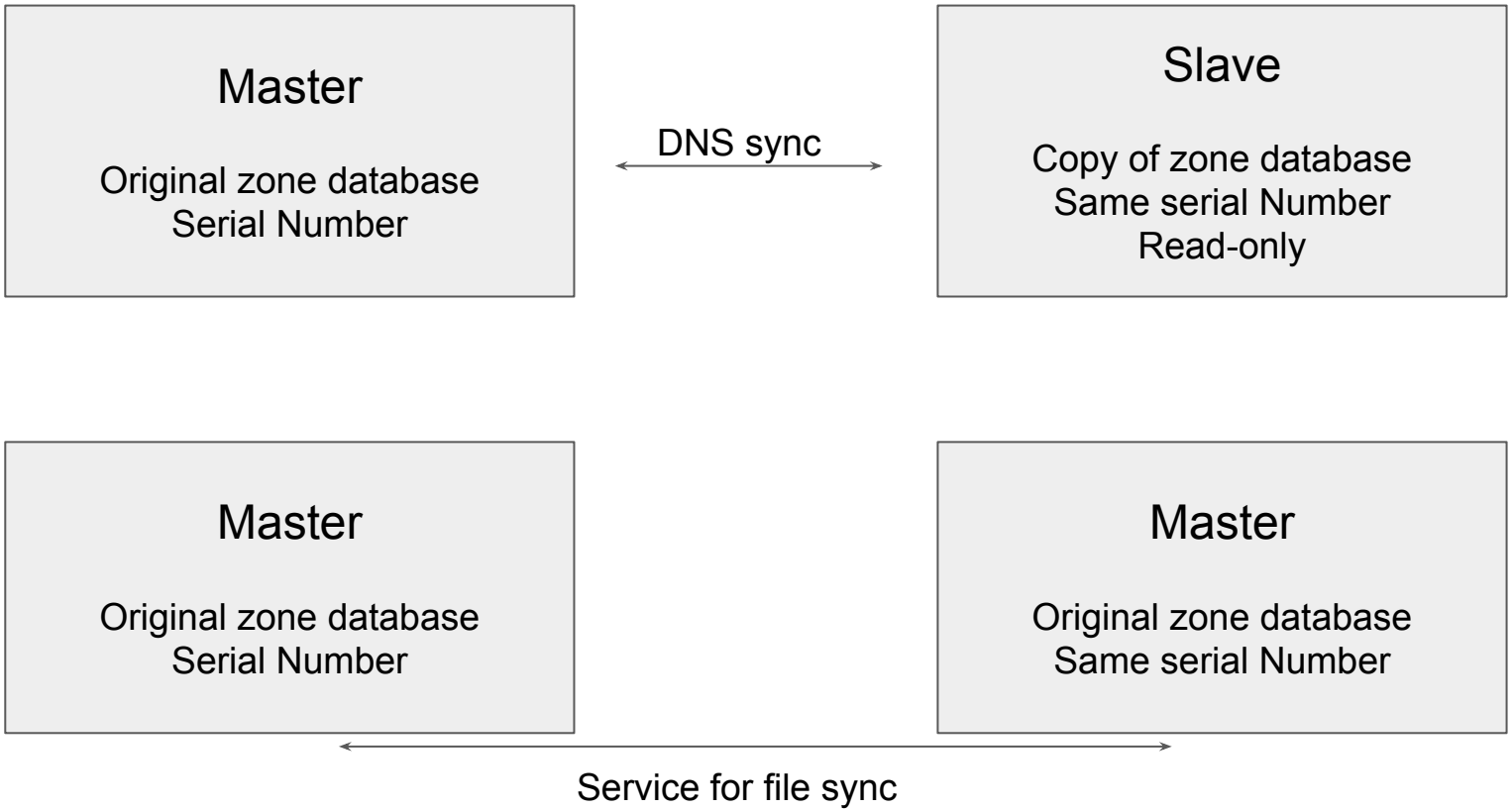
DNS redundancy: round-robin






```
$ cat /etc/resolv.conf
nameserver 10.0.0.1
nameserver 10.0.0.2
```

HA setups


Master
Original zone database
Serial Number
DNS sync
Slave
Copy of zone database
Same serial Number
Read-only
Master
Original zone database
Serial Number
Master
Original zone database
Same serial Number
Service for file sync

SOA record

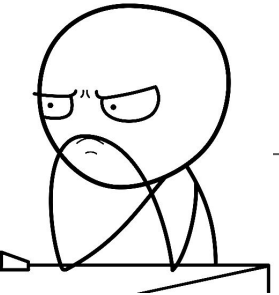
Example of zone database:

```
$TTL  604800
example.com.      IN    SOA   vm1.example.com. admin.example.com. (
                        1           ; Serial
                   604800           ; Refresh
                    86400           ; Retry
                  2419200           ; Expire
                   604800           ; Negative Cache TTL
);
example.com.      IN    NS    vm1
example.com.      IN    NS    vm2
vm1                     IN    A     10.0.0.1
vm2                     IN    A     10.0.0.2
```

Example of SOA request:

```
$ dig soa +short google.com
ns1.google.com. dns-admin.google.com. 344600895 900 900 1800 60
```

Our HA setup



nsupdate+key

Master

Original zone database
Serial Number

notify

SOA

AXFR+key

Slave

Copy of zone database
Same serial Number