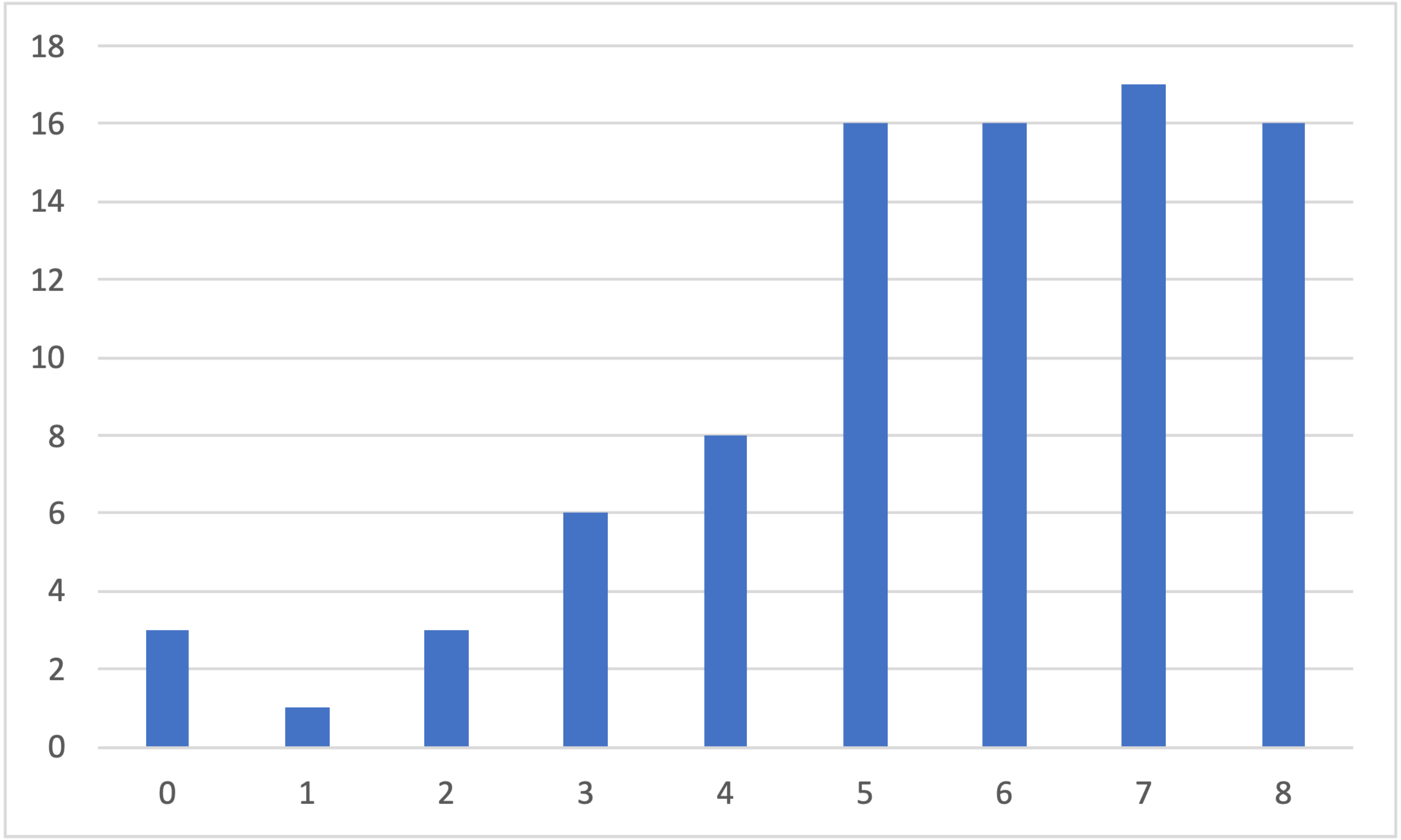
COMP.SE.140
Hosting and Deployment
24.10.2023

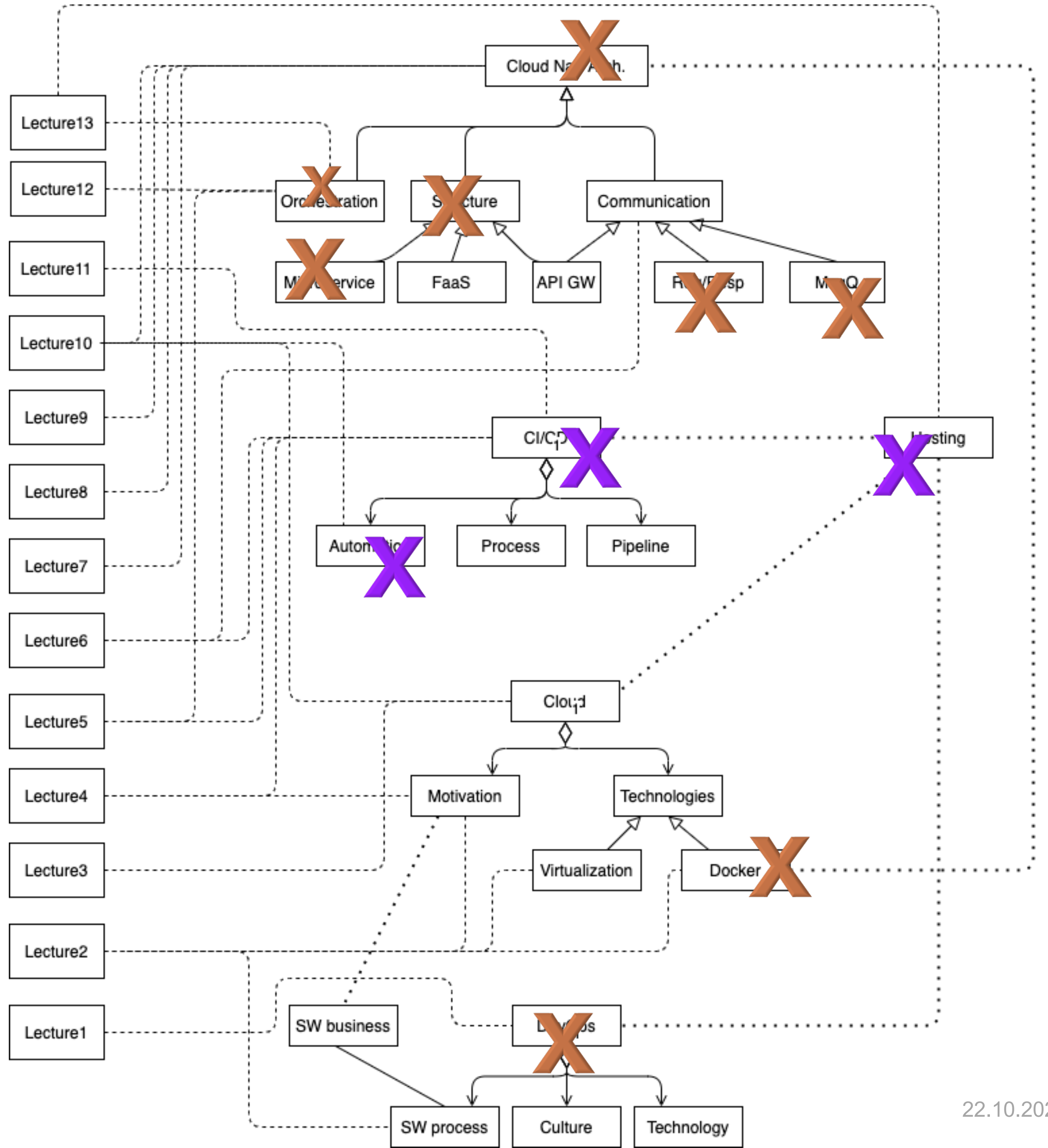
Tampere University

86 submissions - Point distribution



Tampere University

Content map



DevOps, processes, principles

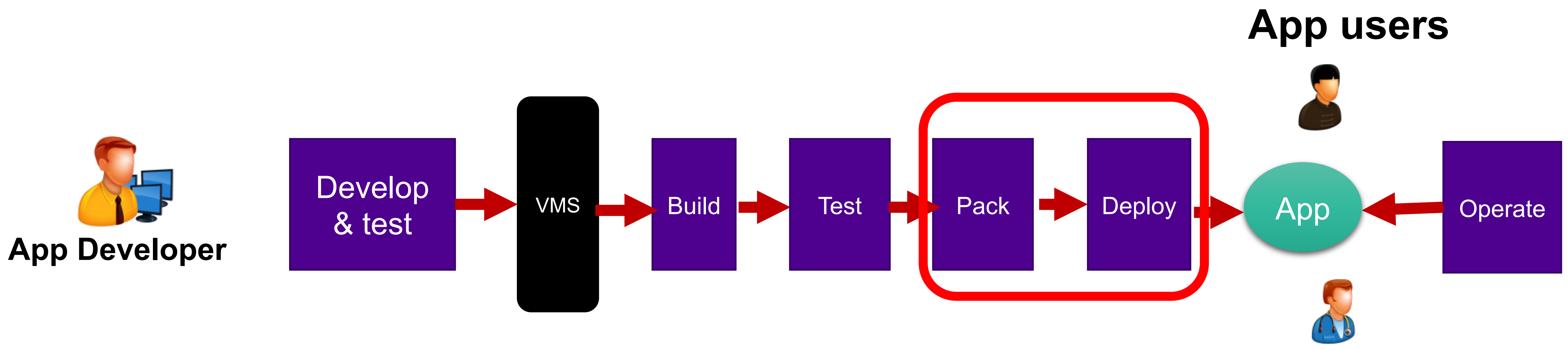
Continuous Deployment, pipeline

Deployment, hosting

Cloud-native architectures

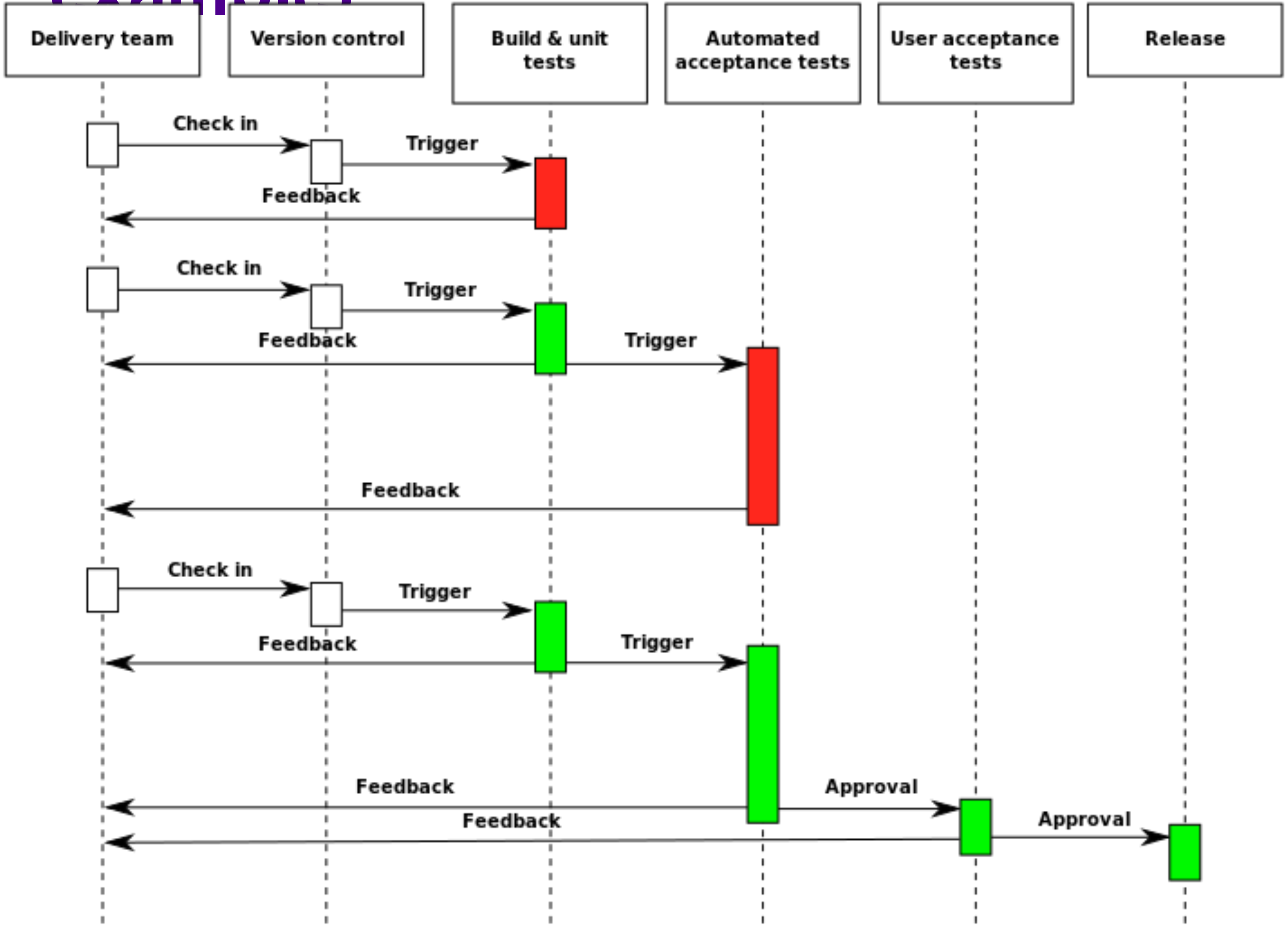
Cloud technologies, containers

Deployment


App users
App Developer
Develop & test
VMS
Build
Test
Pack
Deploy
App
Operate

Tampere University

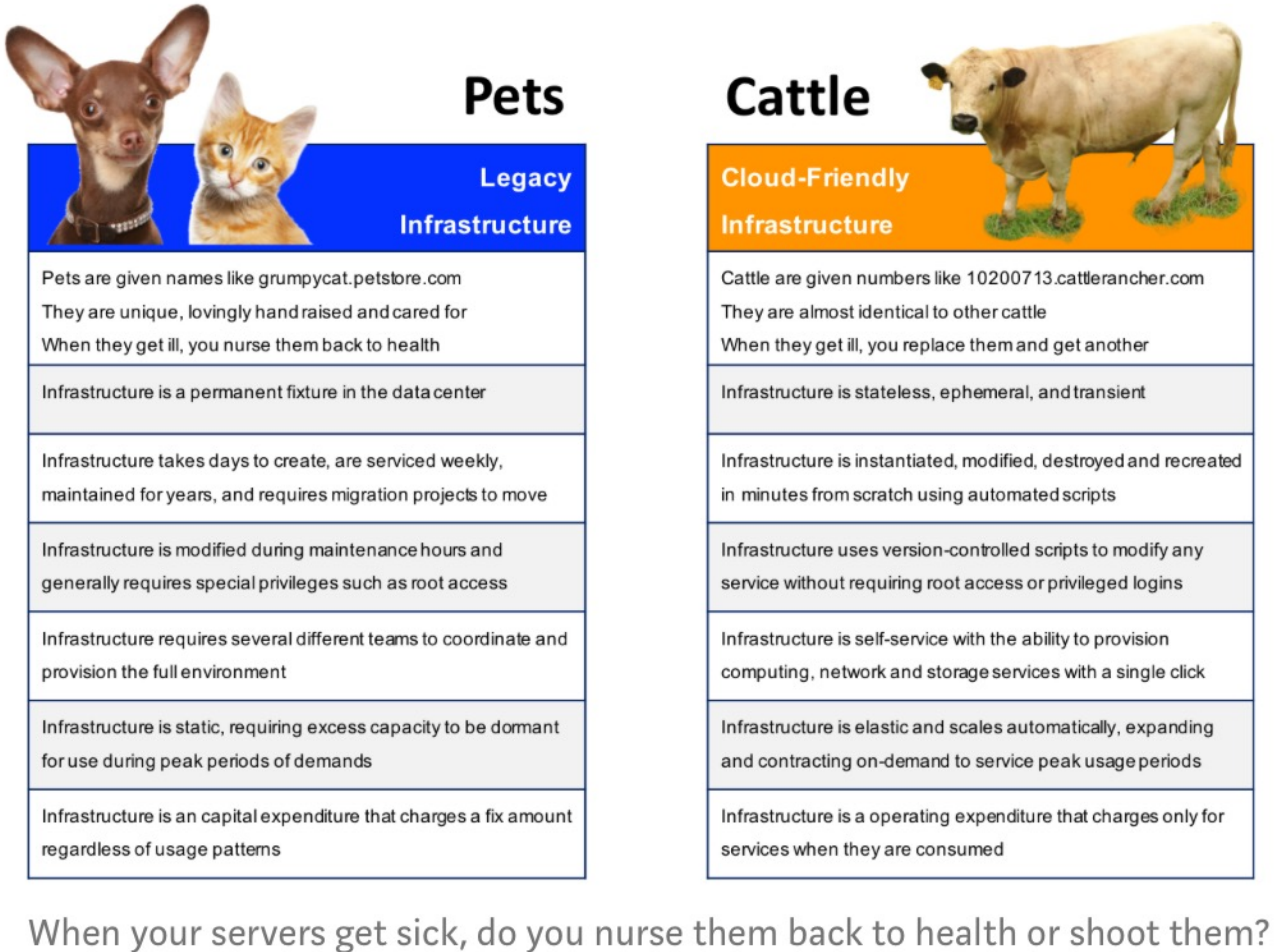
Deployment pipeline (a possible example)



Tampere University

Implications to developers

https://cloudrumblings.io/cloud-farm-pets-cattle-unicorns-and-horses-85271d915260

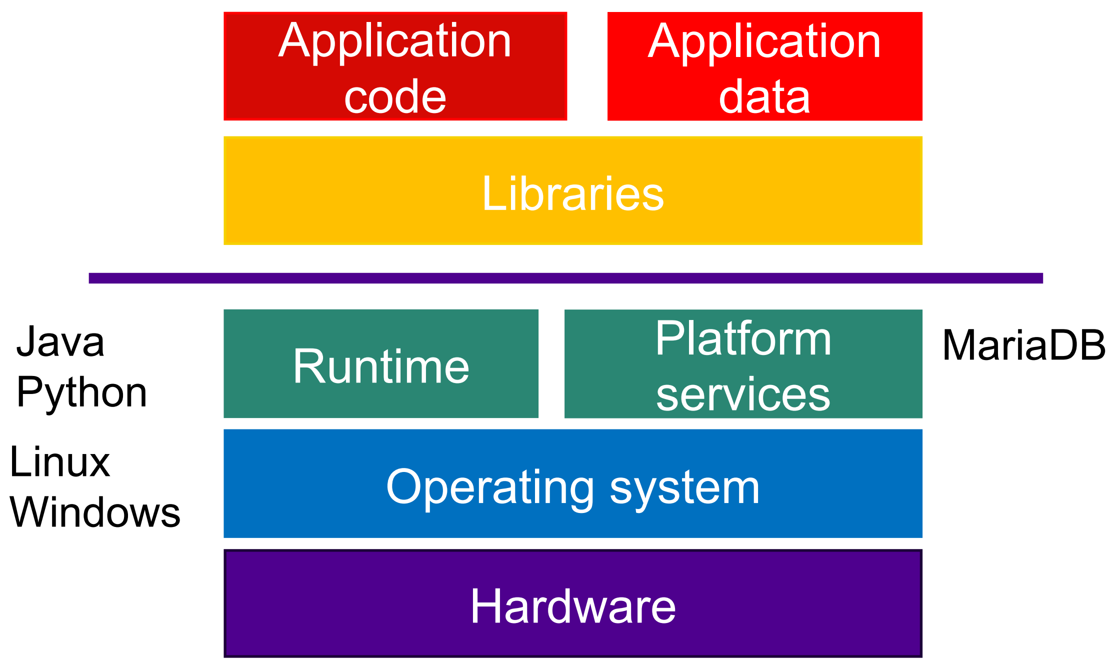


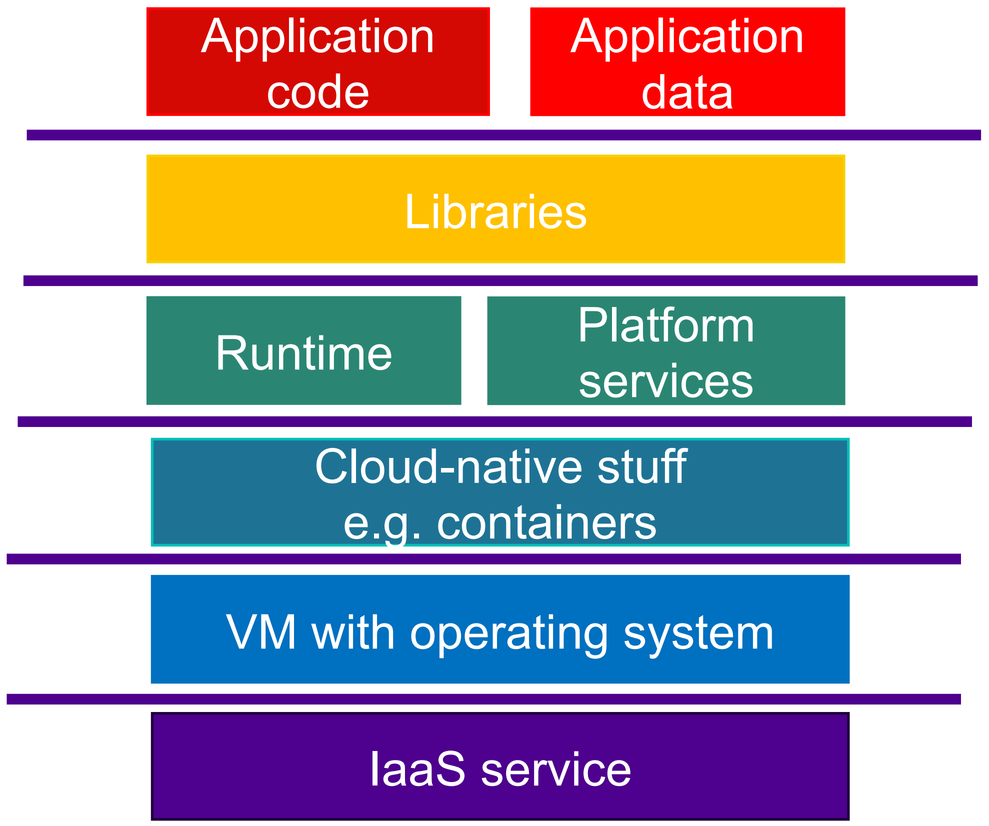
Pets Legacy Infrastructure	Cattle Cloud-Friendly Infrastructure
Pets are given names like grumpycat.petstore.com They are unique, lovingly hand raised and cared for When they get ill, you nurse them back to health	Cattle are given numbers like 10200713.cattlerancher.com They are almost identical to other cattle When they get ill, you replace them and get another
Infrastructure is a permanent fixture in the data center	Infrastructure is stateless, ephemeral, and transient
Infrastructure takes days to create, are serviced weekly, maintained for years, and requires migration projects to move	Infrastructure is instantiated, modified, destroyed and recreated in minutes from scratch using automated scripts
Infrastructure is modified during maintenance hours and generally requires special privileges such as root access	Infrastructure uses version-controlled scripts to modify any service without requiring root access or privileged logins
Infrastructure requires several different teams to coordinate and provision the full environment	Infrastructure is self-service with the ability to provision computing, network and storage services with a single click
Infrastructure is static, requiring excess capacity to be dormant for use during peak periods of demands	Infrastructure is elastic and scales automatically, expanding and contracting on-demand to service peak usage periods
Infrastructure is an capital expenditure that charges a fix amount regardless of usage patterns	Infrastructure is a operating expenditure that charges only for services when they are consumed

When your servers get sick, do you nurse them back to health or shoot them?

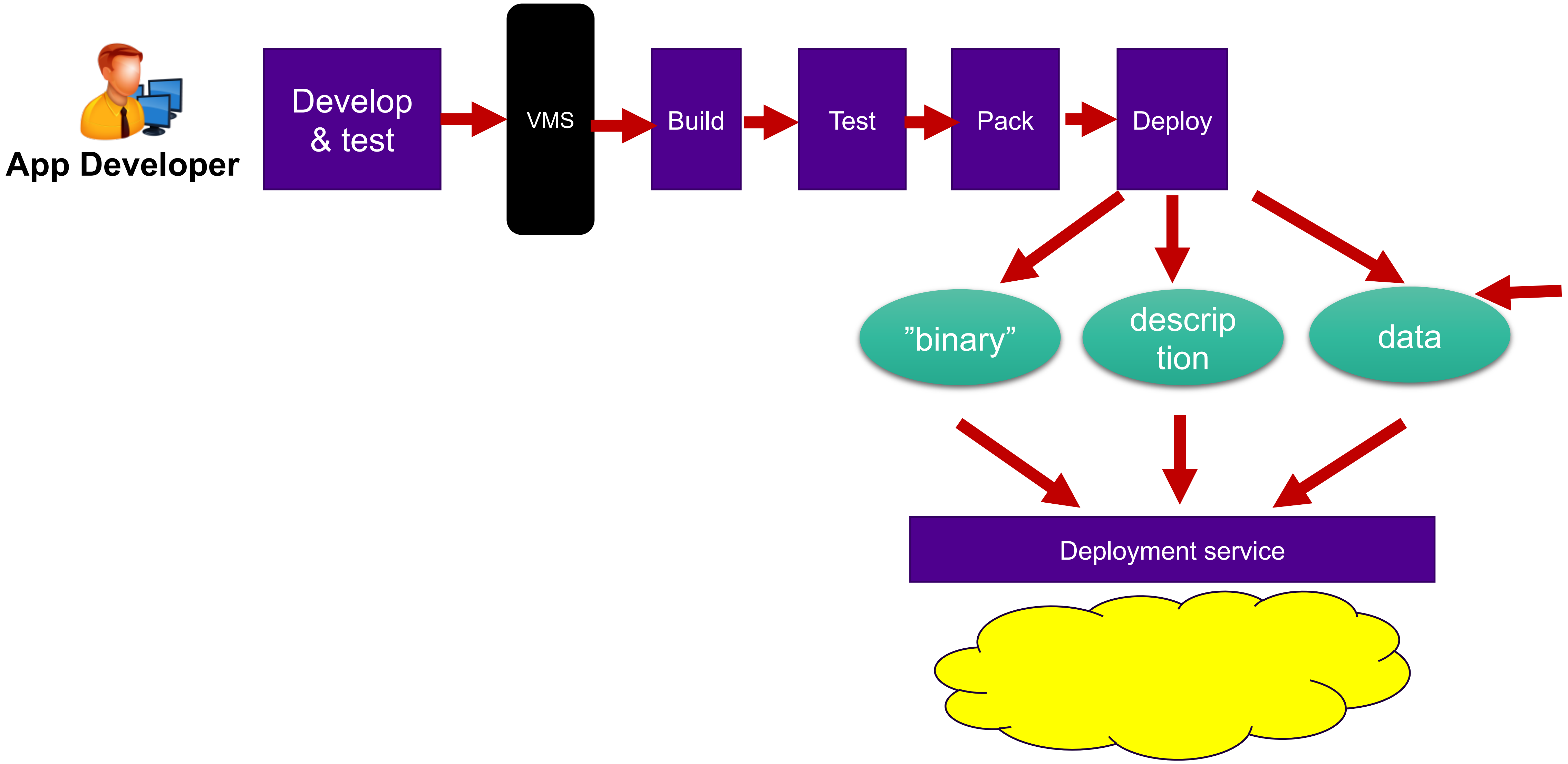
Pets are given names like grumpycat.petstore.com They are unique, lovingly hand raised and cared for When they get ill, you nurse them back to health	Cattle are given numbers like 10200713.cattlerancher.com They are almost identical to other cattle When they get ill, you replace them and get another
Infrastructure is a permanent fixture in the data center	Infrastructure is stateless, ephemeral, and transient
Infrastructure takes days to create, are serviced weekly, maintained for years, and requires migration projects to move	Infrastructure is instantiated, modified, destroyed and recreated in minutes from scratch using automated scripts
Infrastructure is modified during maintenance hours and generally requires special privileges such as root access	Infrastructure uses version-controlled scripts to modify any service without requiring root access or privileged logins
Infrastructure requires several different teams to coordinate and provision the full environment	Infrastructure is self-service with the ability to provision computing, network and storage services with a single click
Infrastructure is static, requiring excess capacity to be dormant for use during peak periods of demands	Infrastructure is elastic and scales automatically, expanding and contracting on-demand to service peak usage periods
Infrastructure is an capital expenditure that charges a fix amount regardless of usage patterns	Infrastructure is a operating expenditure that charges only for services when they are consumed

Deployment target?





Platform provisioning vs application deployment

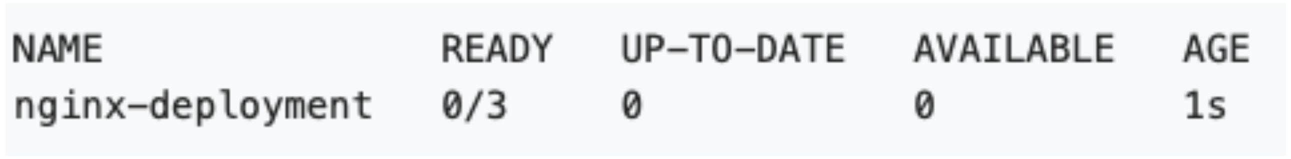

App Developer
Develop & test
VMS
Build
Test
Pack
Deploy
”binary”
descrip tion
data
Deployment service

Example: Deployment to Kubernetes

- Step 1: create Deployment (configuration)
- “A Deployment is responsible for creating and updating instances of your application”

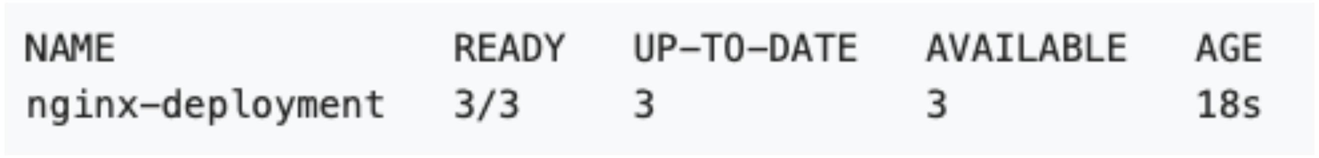
```
$ kubectl apply -f
    https://k8s.io/examples/controllers/nginx-deployment.yaml
```

```
$ kubectl get deployments
```

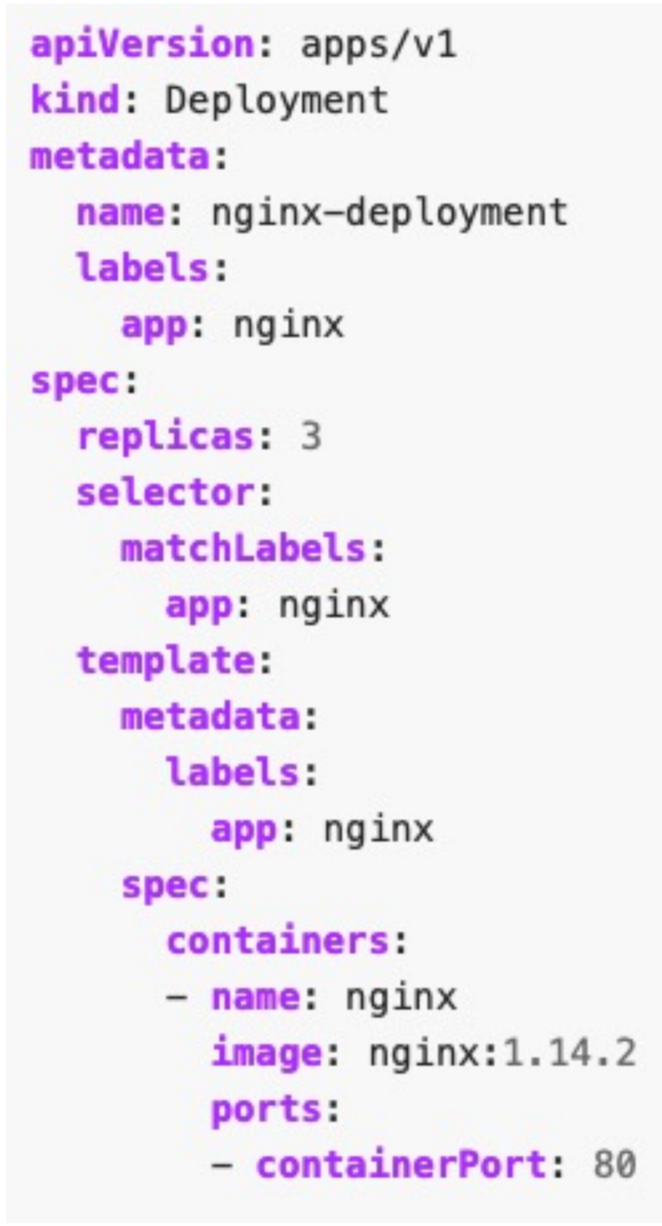


```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/3     0            0           1s
```

```
$ kubectl get deployments
```



```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           18s
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Tampere University

Example: with AWS codedeploy

https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-windows.html

(command-line version)

- Step 1: Launch a Windows Server Amazon EC2 instance
- Step 2: Configure your source content to deploy to the Windows Server Amazon EC2 instance
- Step 3: Upload your "hello, world!" application to Amazon S3
- Step 4: Deploy your Hello World application
- Step 5: Update and redeploy your "hello, world!" application
- Step 6: Clean up your "hello, world!" application and related resources

Example: with AWS codedeploy

https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-windows.html

(command-line version)

- Step 1: Launch a Windows Server Amazon EC2 instance
- Step 2: Configure your source content to deploy to the Windows Server Amazon EC2 instance
- Step 3: U load your "hello, world!" application to Amaz n S3
- Step 4: D y your Hello World application
- Step 5: U nd redeploy your "hello, world!" application
- Step 6: Cle ur "hello, world!" application and related reso

appspec.yaml

```
version: 0.0
os: windows
files:
 - source: \index.html
   destination: c:\inetpub\wwwroot
hooks:
  BeforeInstall:
   - location: \before-install.bat
     timeout: 900
```

before-install.bat

```
REM Install Internet Information Server (IIS).
powershell.exe -Command Import-Module -Name ServerManager
powershell.exe -Command Install-WindowsFeature Web-Server
```

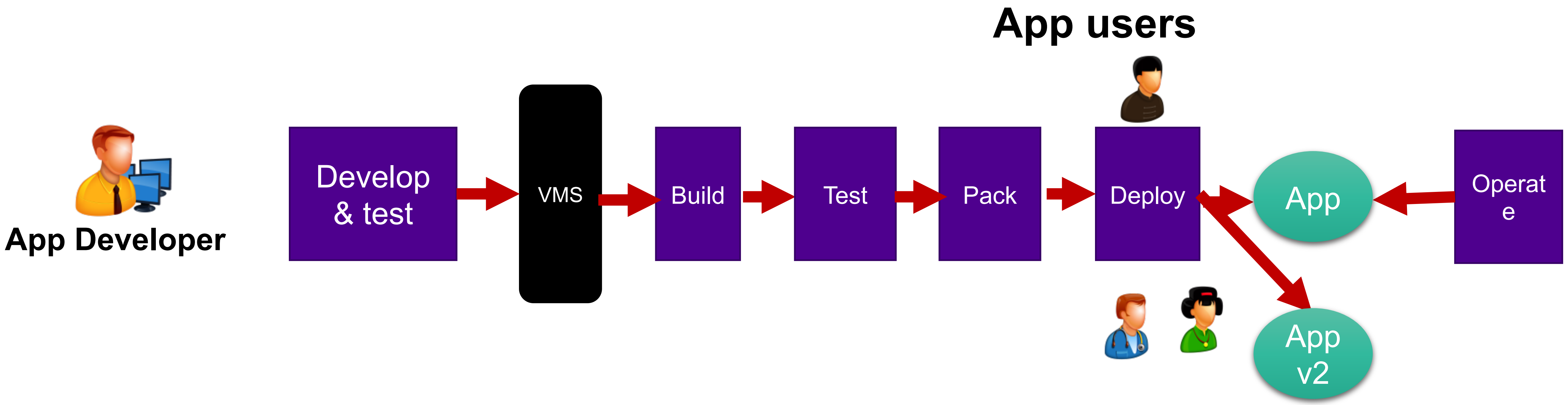
Tampere University

Example: with AWS codedeploy

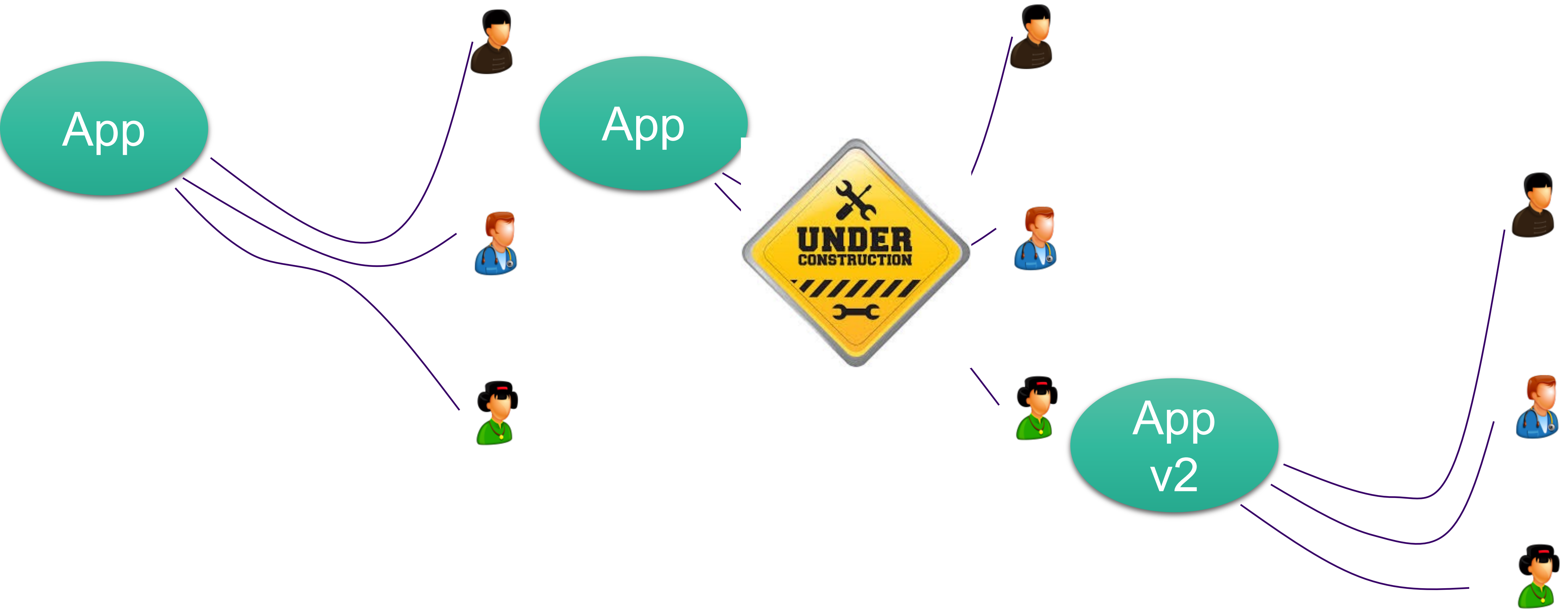
https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-windows.html

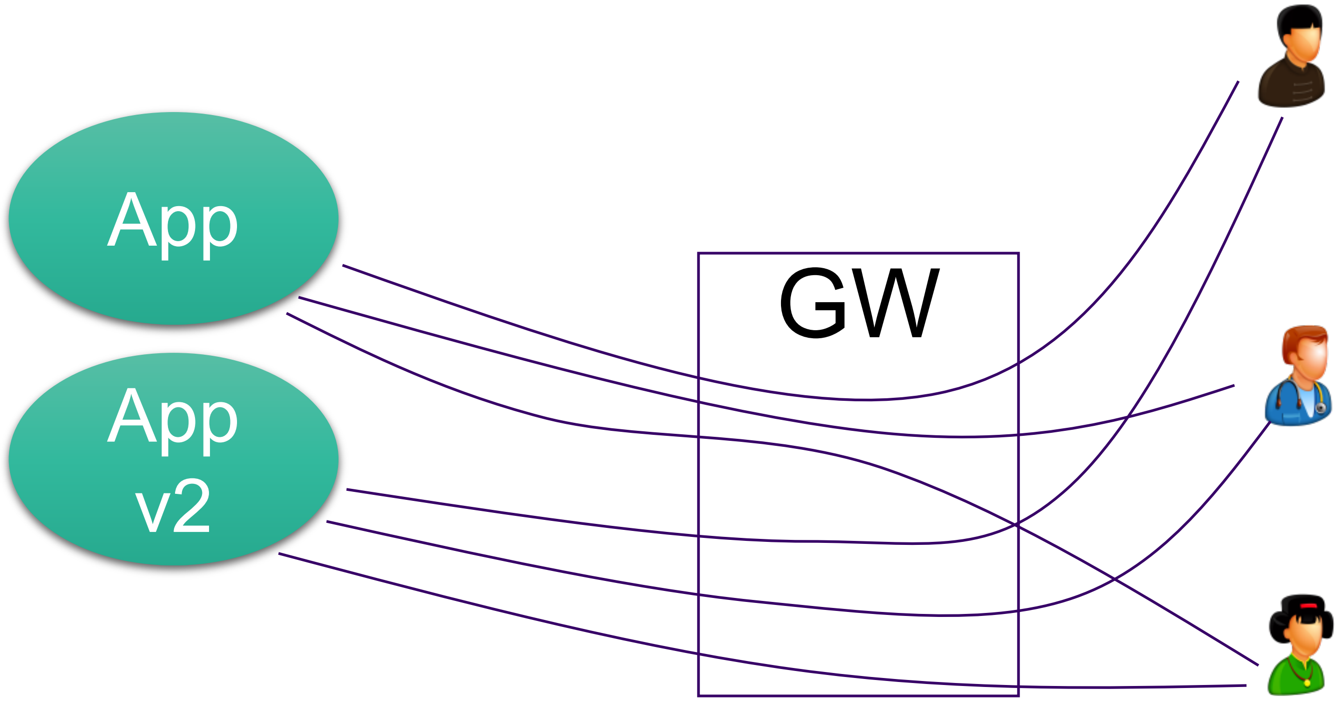
(command-line version)

- Step 1: Launch a Windows Server Amazon EC2 instance
- Step 2: Configure your source content to deploy to the Windows Server Amazon EC2 instance
- Step 3: Upload your "hello, world!" application to Amazon S3
- Step 4: Deploy your Hello World application
- Step 5: Update and redeploy your "hello, world!" application
- Step 6: Clean up your "hello, world!" application and related resources


App users
App Developer
Develop & test
VMS
Build
Test
Pack
Deploy
App
Operate
App v2

A possible strategy to deploy a new version?





Problems & issues?

Deployment strategies

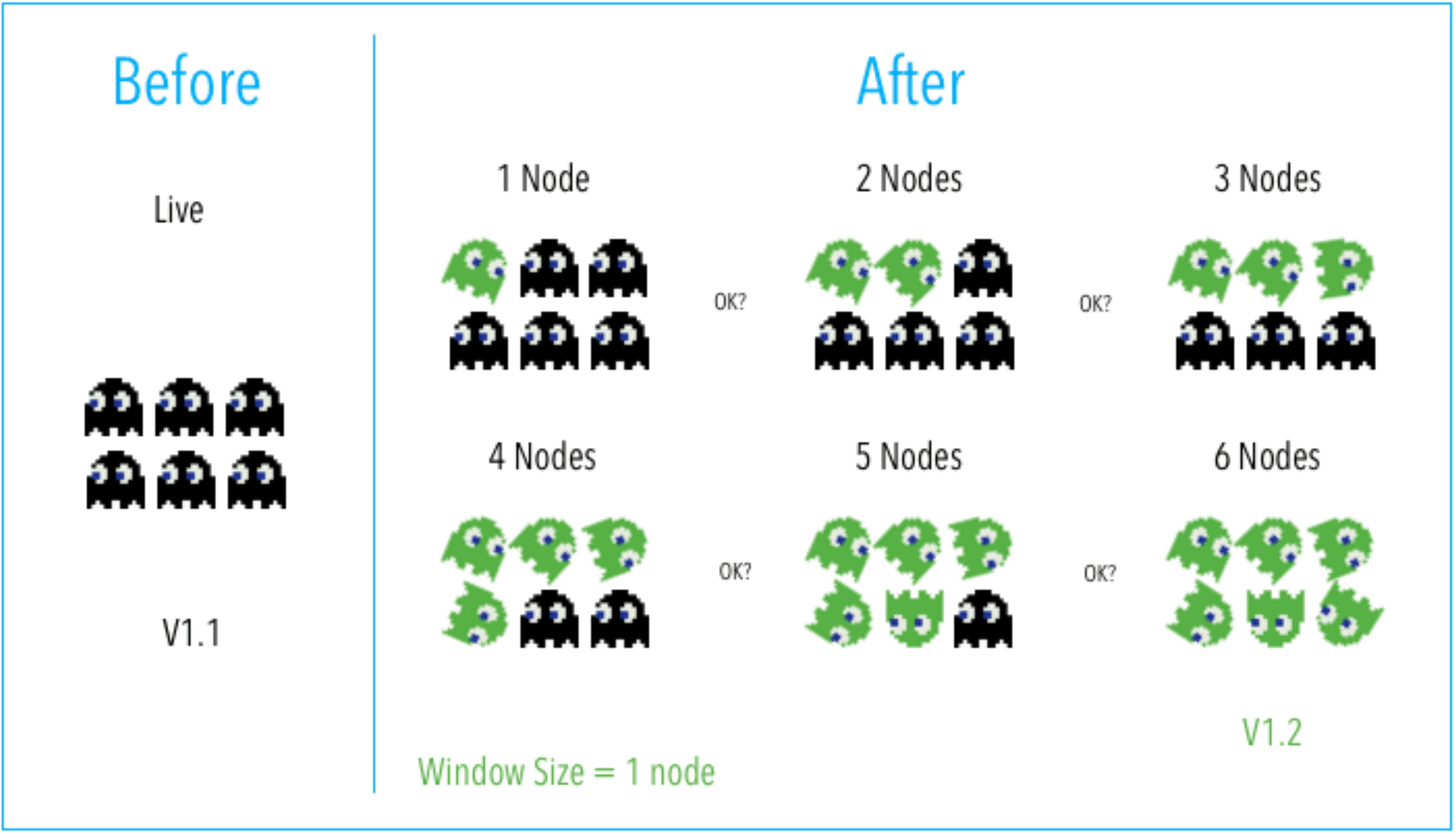
Tampere University

Basic Deployment (aka Suicide) (https://harness.io/2018/02/deployment-strategies-continuous-delivery/) all nodes are updated at the same time

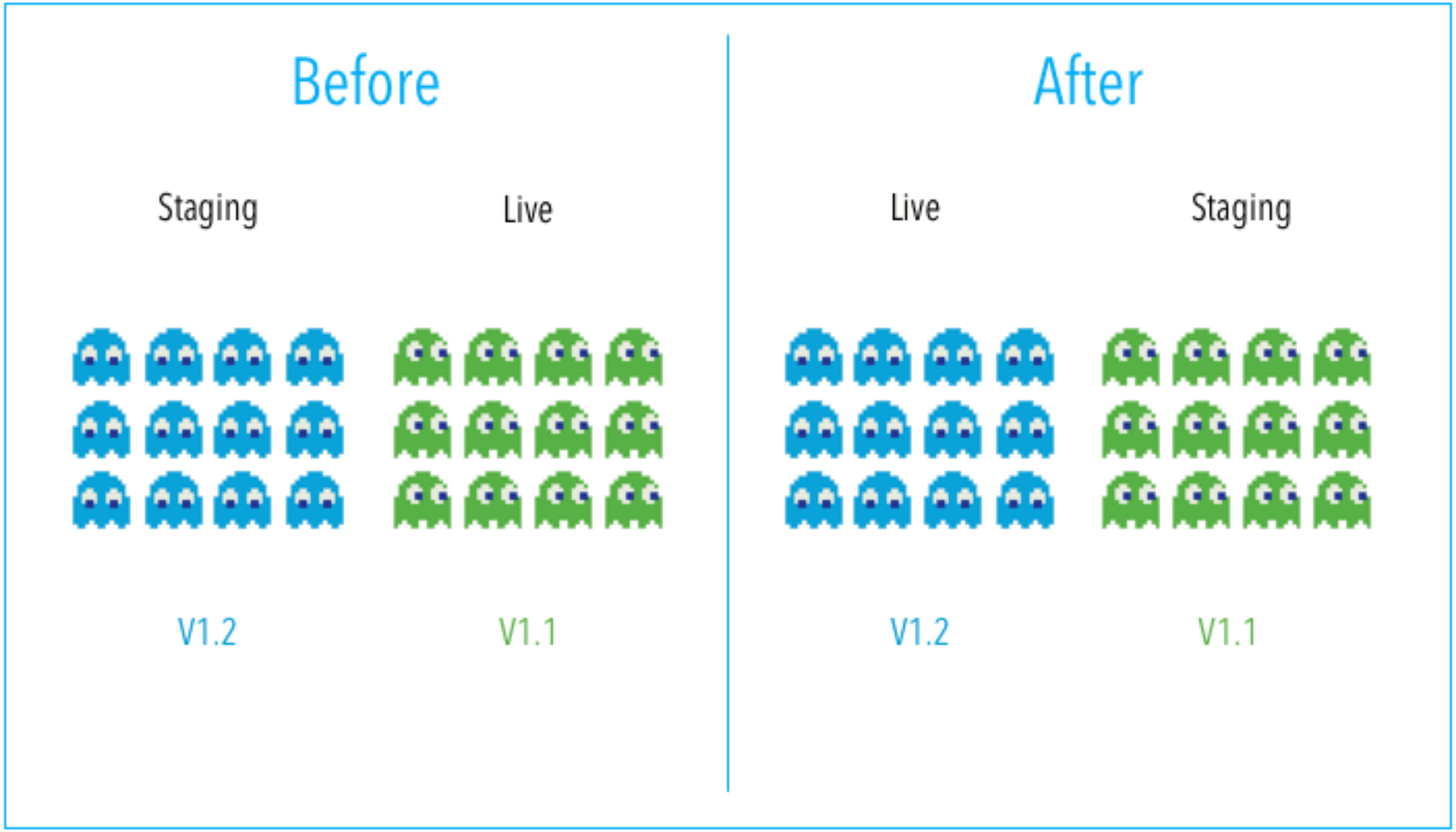
Before
After
Live
Live
Live
Oh $h!&
or
V1.0
V2.0
V2.0

Tampere University

Rolling Deployment (https://harness.io/2018/02/deployment-strategies-continuous-delivery/) nodes are updated incrementally

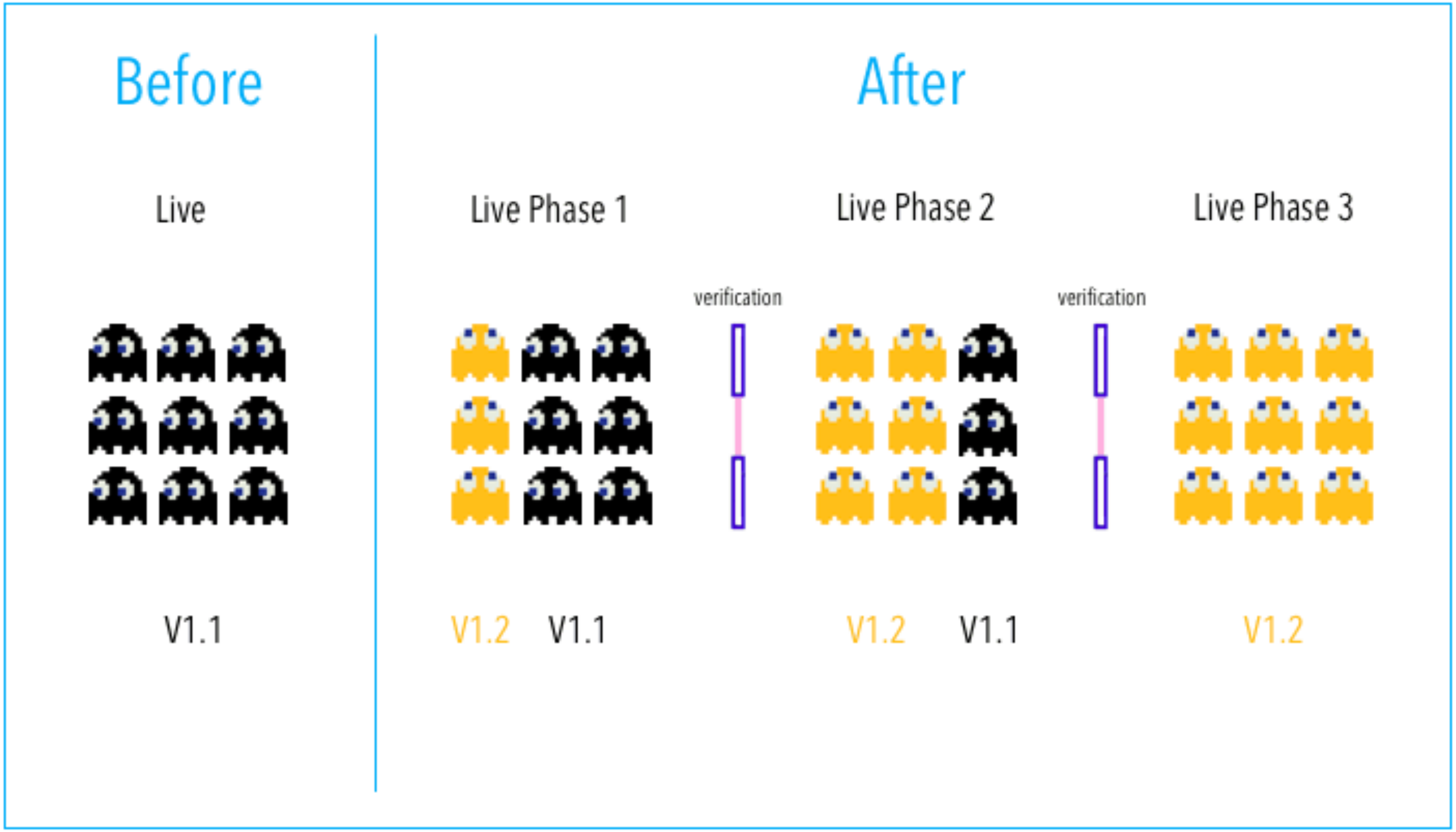


(http://martinfowler.com/bliki/BlueGreenDeployment.html) the new version (called green) is set up in parallel with the current (blue). When new (green) is ready, the router is switched to new (green) and blue is left as a backup. If something goes wrong with new, the router can be switched back to old - that means easy "rollback".

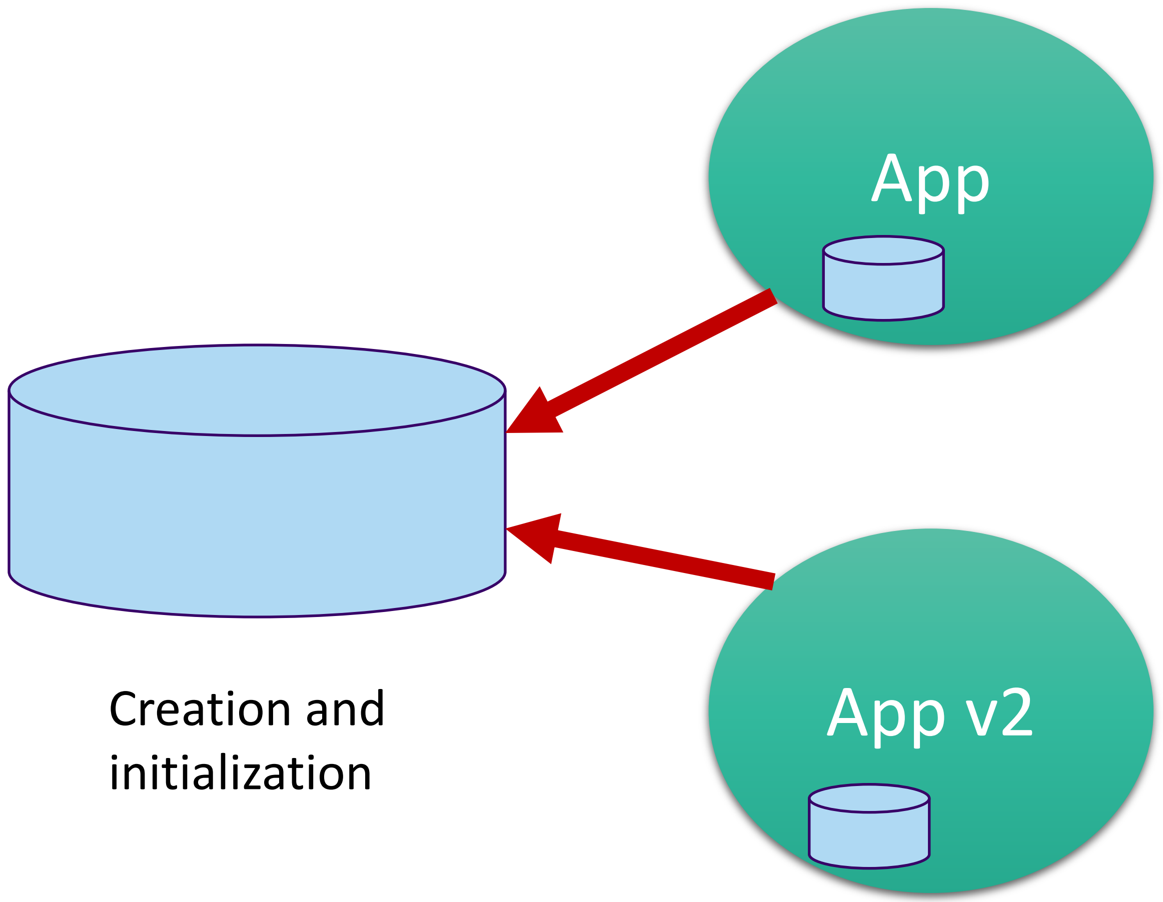


Tampere University

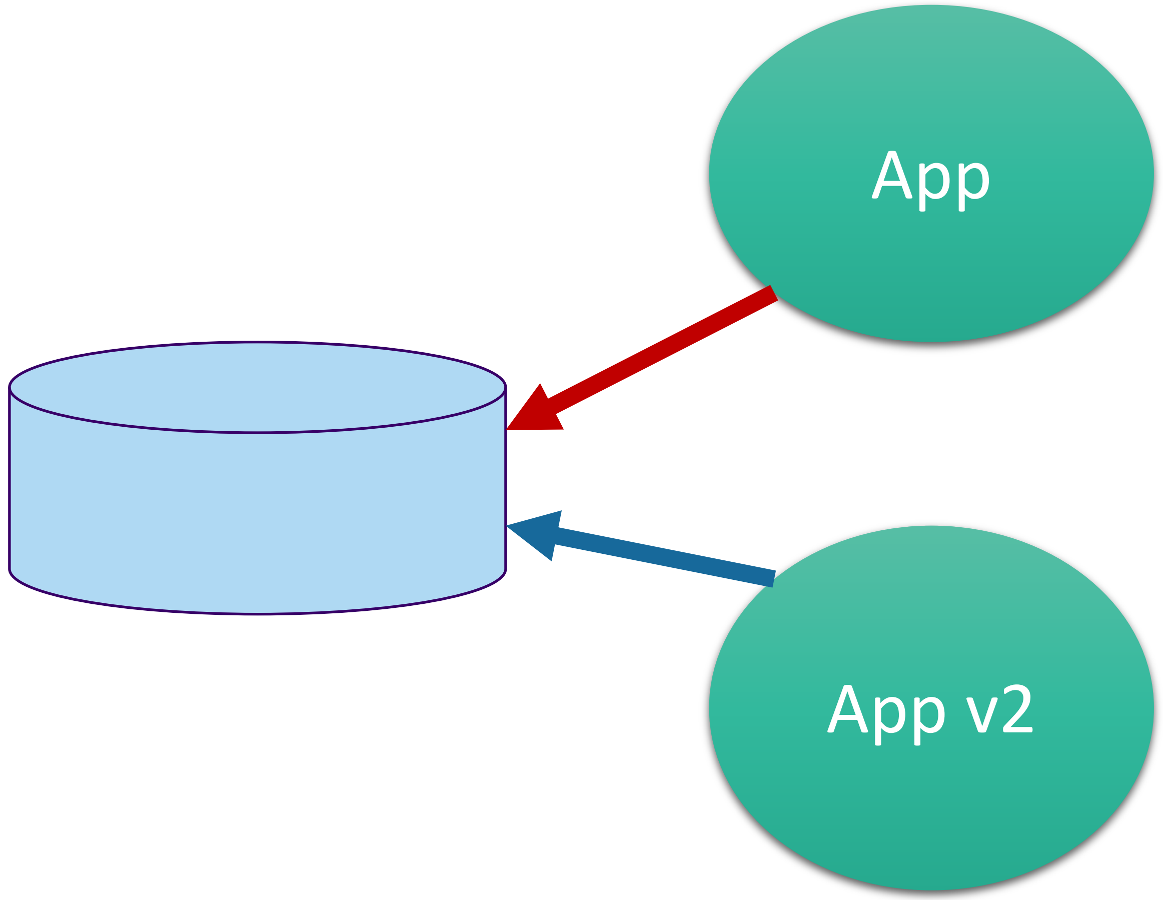
Canary Releases (http://martinfowler.com/bliki/CanaryRelease.html) implements the deployment incrementally. In this case the router first directs only part of the customers to the new version. If feedback is is good, the other customers are moved to new version, too



How about the data?



Creation and initialization

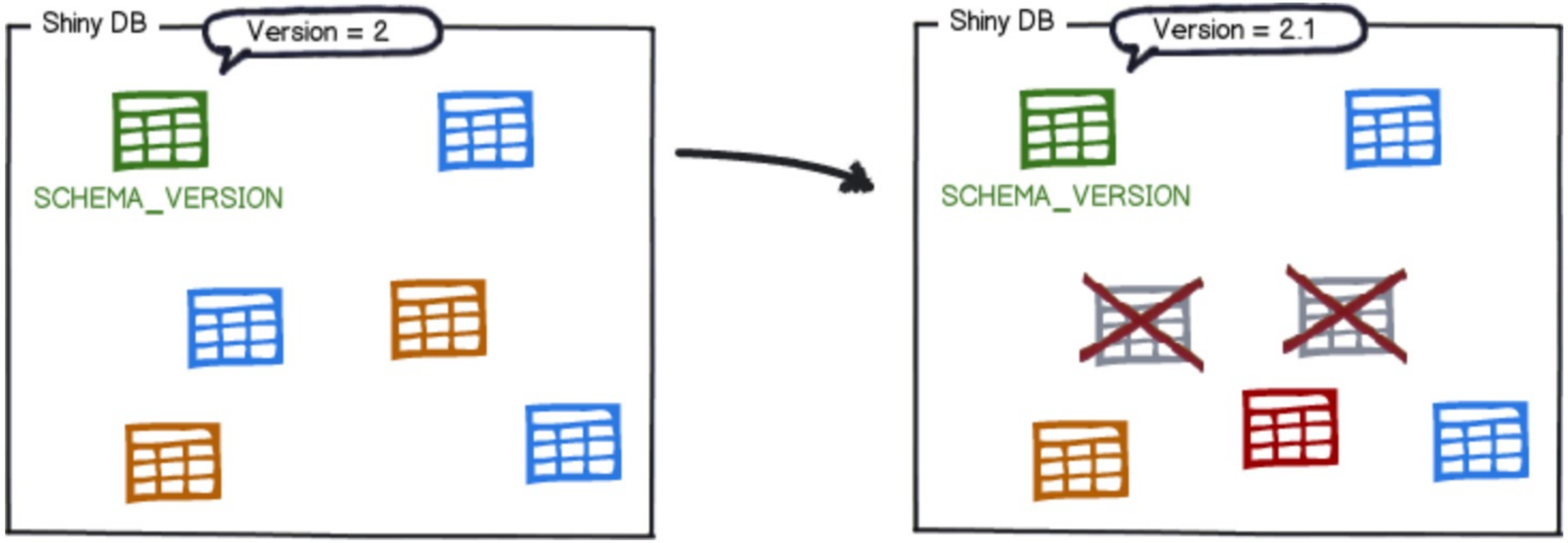


Tampere University

Data migration

- Versions of the data bases
- Data migration scripts are needed.
- Rollback need to be possible

They are then **sorted by version number** and **executed in order**:

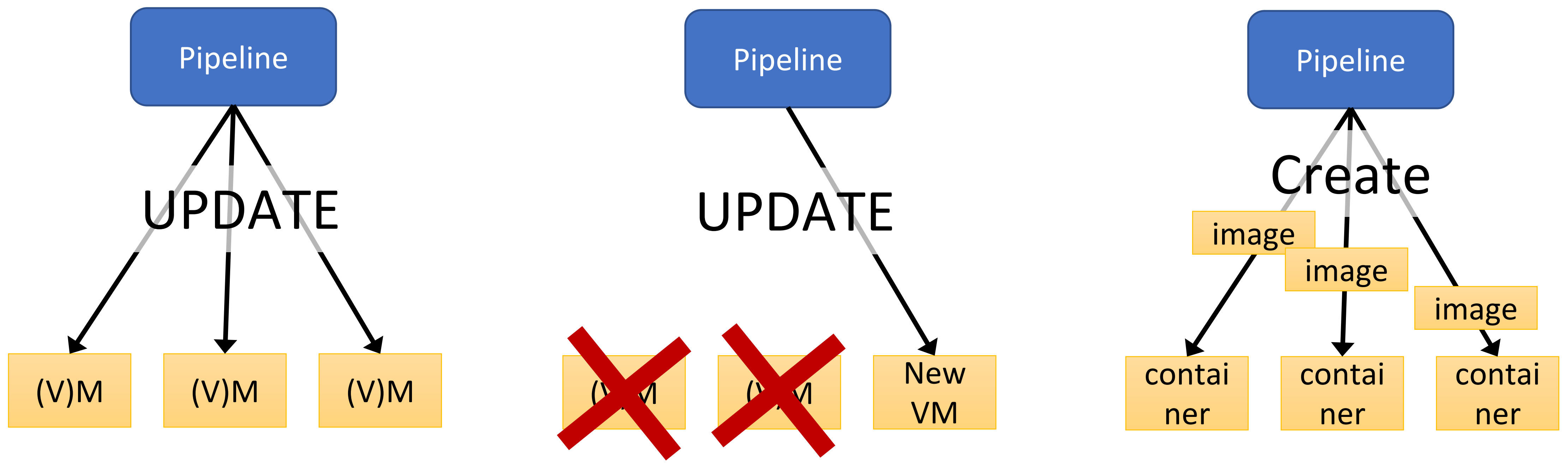


The **schema history table** is **updated** accordingly:

flyway_schema_history

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Initial Setup	SQL	V1__Initial_Setup.sql	1996767037	axel	2016-02-04 22:23:00.0	546	true
2	2	First Changes	SQL	V2__First_Changes.sql	1279644856	axel	2016-02-06 09:18:00.0	127	true
3	2.1	Refactoring	JDBC	V2_1__Refactoring		axel	2016-02-10 17:45:05.4	251	true

To summarize



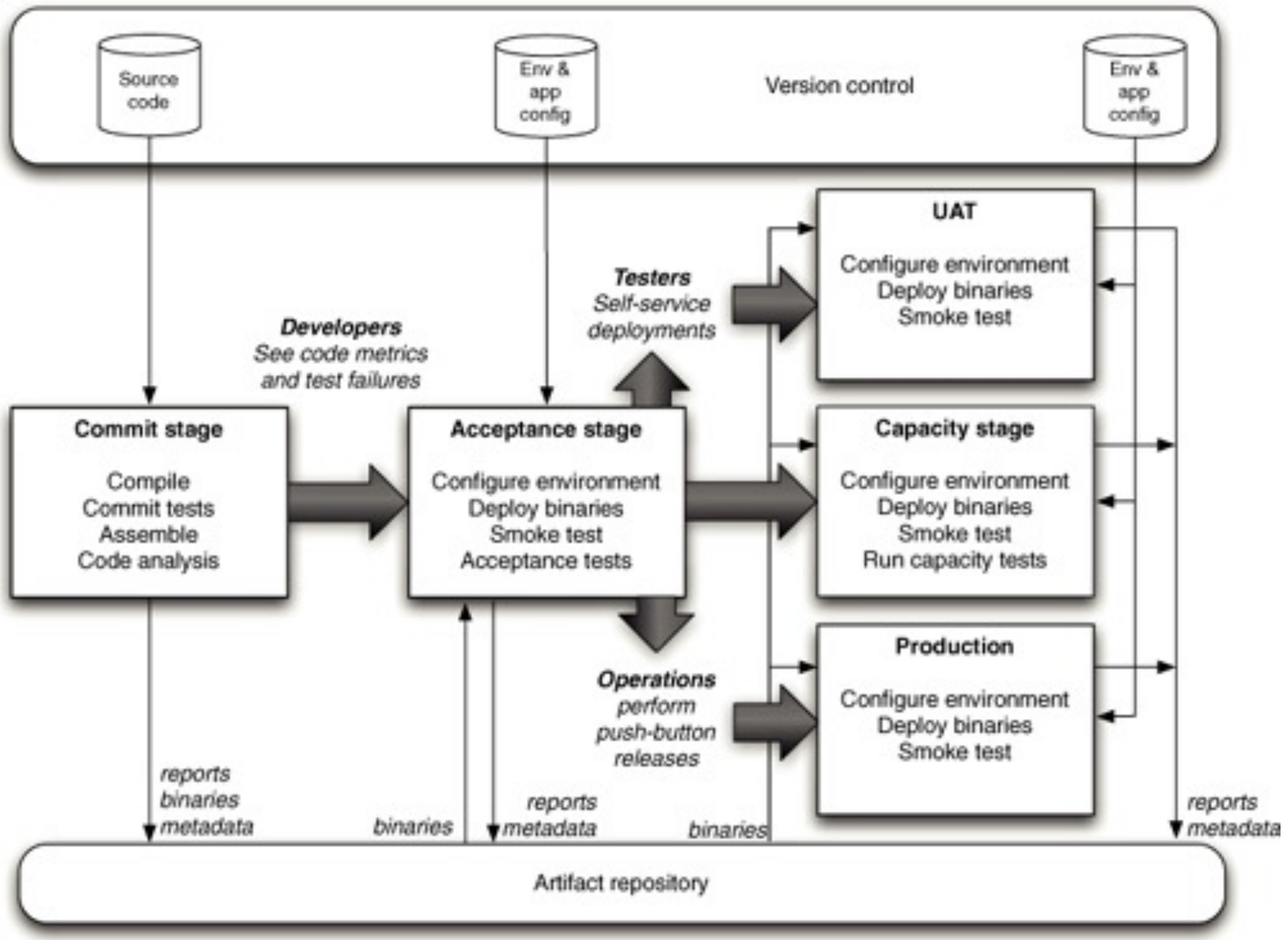
Tampere University

Automation challenges

- ”…provisioning scripts were considered error-prone and, according to developers, they did not work in some environments…”
- ”…automation of the network in was said to be difficult in addition to dealing with legacy system…”
- ”Networks are pretty hard. Some of the databases are pretty hard too because the old relational databases haven’t been designed to be clustered…”

Tampere University

Artefact repository



Huge number or tools available

- https://digital.ai/periodic-table-of-devops-tools
- https://landscape.cncf.io

Tampere University

Alternative approaches for delivery

- Set-up everything when image is created
  - Very static
- Make the container to auto-update
  - You need to know in advance what might change
- ~~Put stuff to shared folder (use volume)~~
- Use configuration tools
  - Work also for full virtual machines and computers

Ansible (https://www.ansible.com)

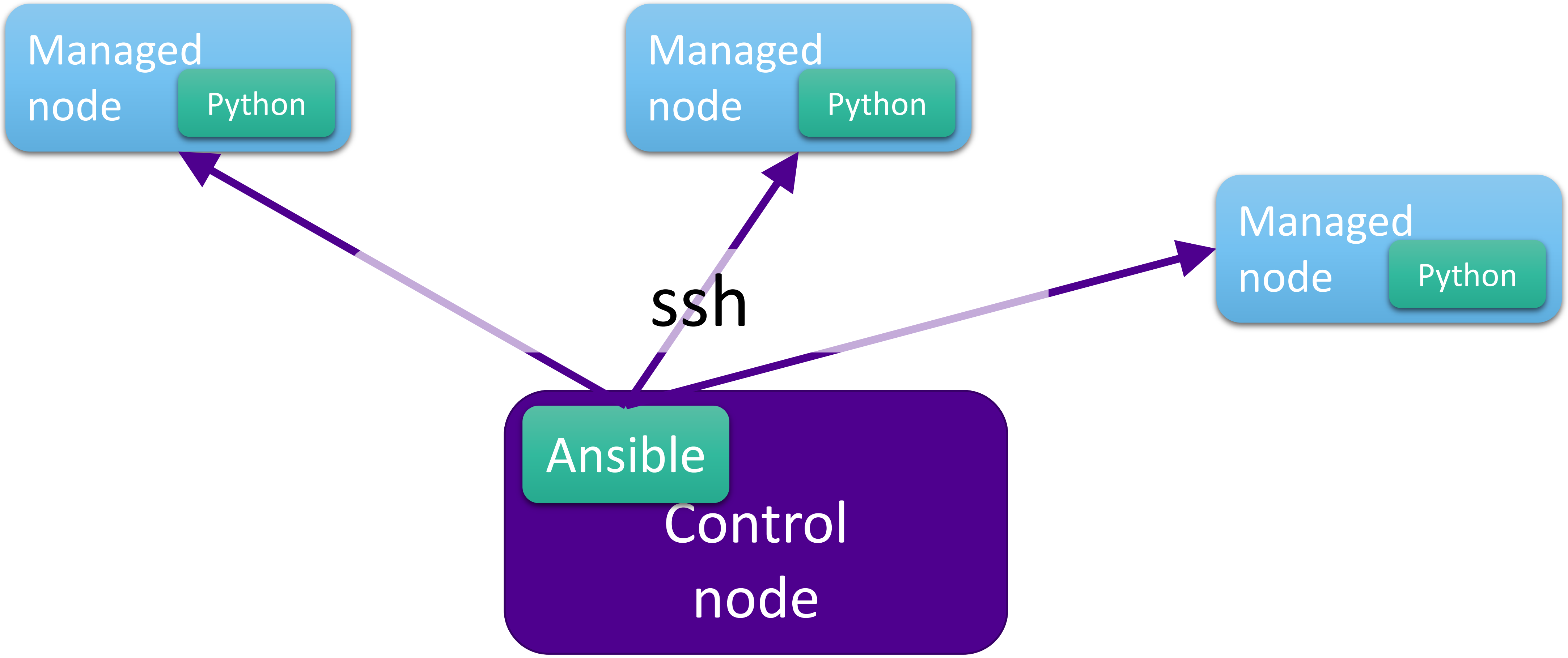
Automation engine for

- Provisioning
- Configuration Management
- App Deployment
- Continuous Delivery
- Security Automation
- Orchestration

uses YAML, in the form of Ansible Playbooks

Tampere University

Architecture



Ansible

- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them.
- These programs are written to be resource models of the desired state of the system.
- Ansible then executes these modules (over SSH by default), and removes them when finished.
- Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.
- Typically, you'll work with your favourite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- A short video:
  - https://www.ansible.com/resources/videos/quick-start-video

Example ansible playbook

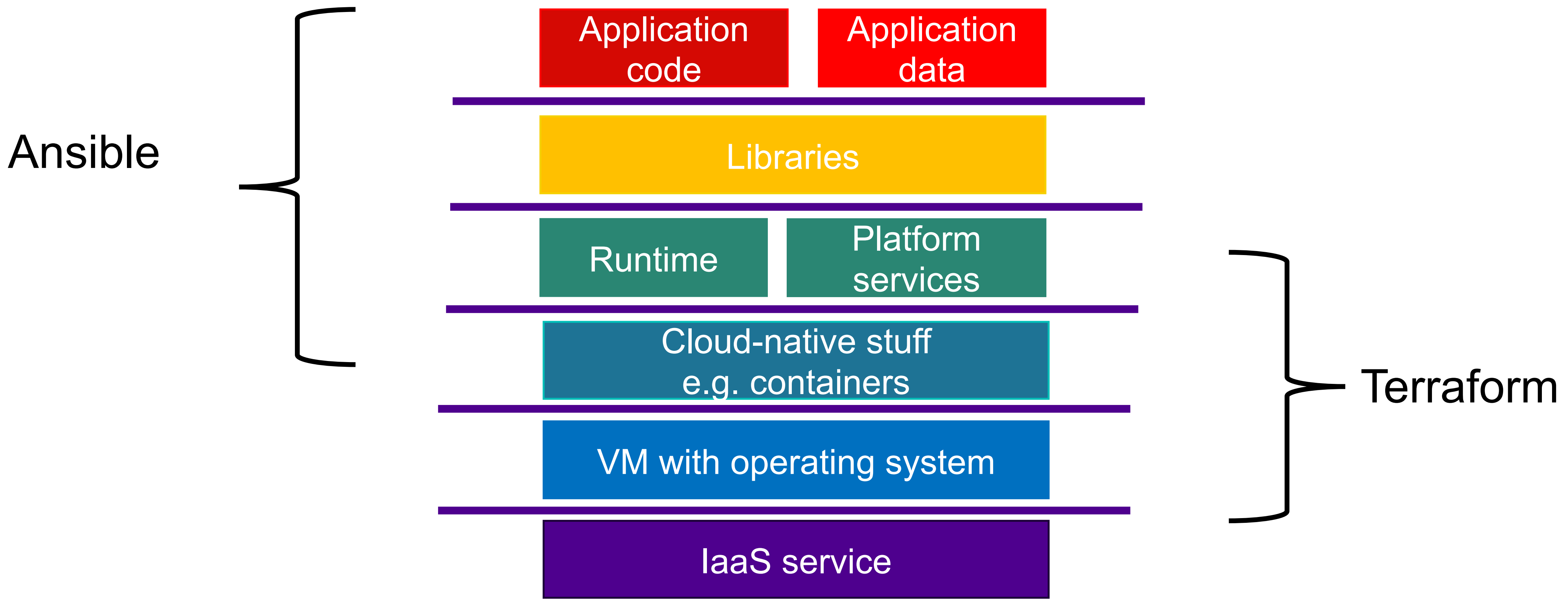
```
---

-hosts: webservers
 vars:
    http_port: 80
    max_clients: 200
 remote_user: root
 tasks:
 - name: ensure apache is at the
         latest version
   yum:
     name: httpd
     state: latest
 - name: write the apache config
         file.
   template:
     src: /srv/httpd.j2
     dest:/etc/httpd.conf
   notify:
     - restart apache
```

```
 - name: ensure apache is
         running
   service:
     name: httpd
     state: started

handlers:
   - name: restart apache
     service:
       name: httpd
       state: restarted
```

Terraform - a tool for infrastructure provisioning



Next exercise

Docker containers as targets

- Since we do not have enough virtual machines, lets use Docker images
- Complicates the exercise,
- but allows you to learn more about Docker

Creating the docker image 1/2

```
FROM debian
USER root
# Copy application itself:
COPY . /home
WORKDIR /home
RUN apt-get update
RUN apt-get install -y nodejs
ENTRYPOINT node server.js
```

Debugging aid

Time consuming init

```
Docker build –t utest
```

Creating the docker image 2/2

```
FROM  utest

RUN apt-get install -y openssh-server
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/’ /etc/ssh/sshd_config
RUN apt-get install -y net-tools
RUN useradd -m -s /bin/bash -G sudo -p $(openssl passwd -1 eee) ssluser

RUN apt-get install -y python3
RUN apt-get install -y sudo

ENV PORT=8894
EXPOSE 22

ENTRYPOINT service ssh start && node server.js
```

Tampere University

SSH support two alternative ways for authentication

Password

- Used in the previous slide
- Not very secure
- You can use, but gives at most 80% of the maximum points

Public/private keypair

- Public key of your computer is installed to the host
- More secure
- If you want 100% of maximum points, you should use this (building of the image need to be changed)

Info

- Short:https://unix.stackexchange.com/questions/210228/add-a-user-without-password-but-with-ssh-and-public-key
- Long: https://www.ssh.com/academy/ssh/key

Example ansible playbook

```
---

-hosts: webservers
 vars:
    http_port: 80
    max_clients: 200
 remote_user: root
 tasks:
 - name: ensure apache is at the
         latest version
   yum:
     name: httpd
     state: latest
 - name: write the apache config
         file.
   template:
     src: /srv/httpd.j2
     dest:/etc/httpd.conf
   notify:
     - restart apache
```

```
- name: ensure apache is
        running
  service:
    name: httpd
    state: started

handlers:
  - name: restart apache
    service:
      name: httpd
      state: restarted
```

Tampere University

Apt vs yum

Operating System	Format	Tool(s)
Debian	.deb	apt, apt-cache, apt-get, dpkg
Ubuntu	.deb	apt, apt-cache, apt-get, dpkg
CentOS	.rpm	yum
Fedora	.rpm	dnf
FreeBSD	Ports, .txz	make, pkg

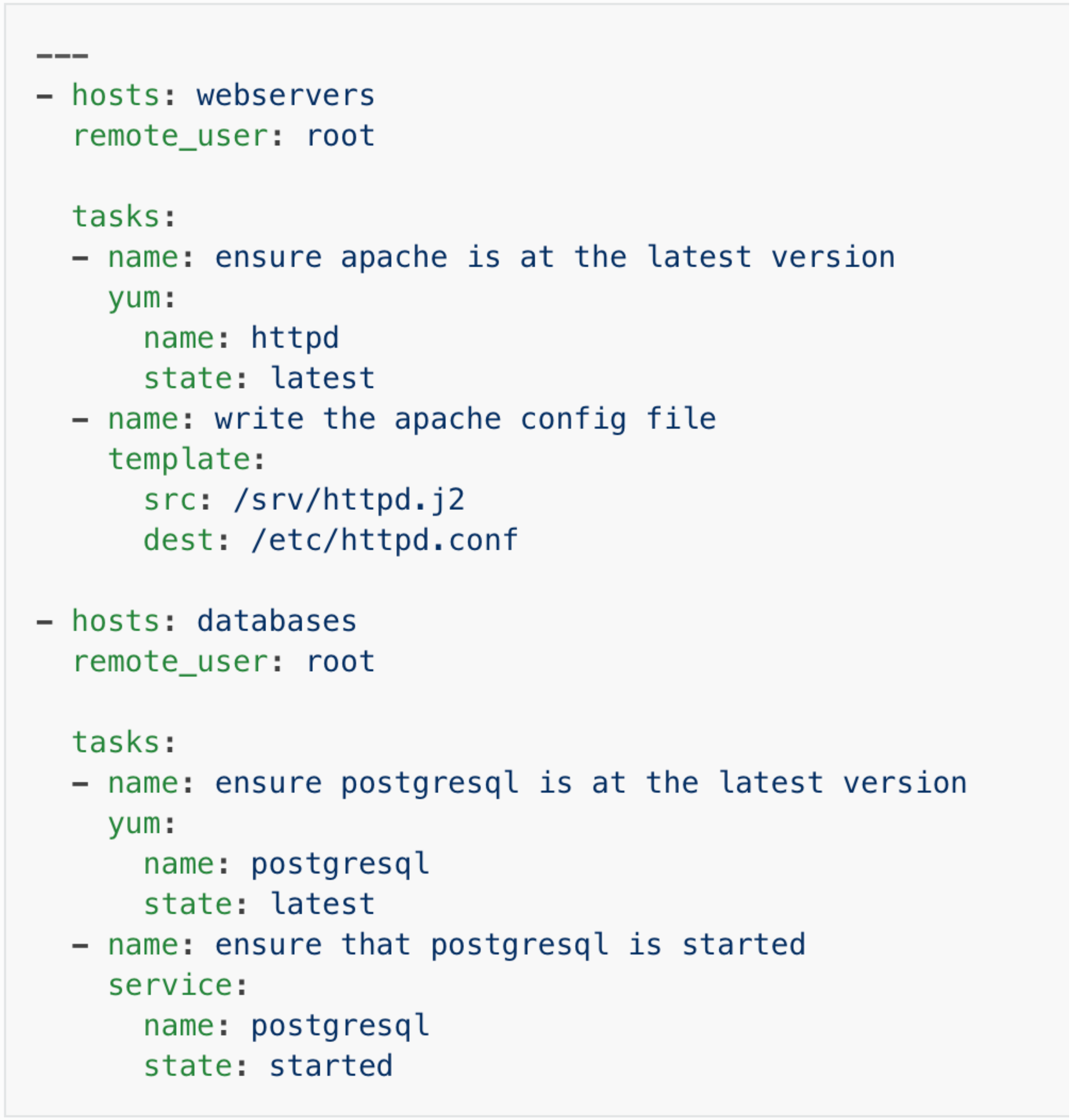
Tampere University

Sidenode: apt vs yum examples

Task	apt (deb)	yum (rpm)	zypper (rpm)
Install from repository	apt-get install pkg-name	yum install pkg-name	zypper install pkg-name
Update package	apt-get install pkg-name	yum update pkg-name	zypper update -t package pkg-name
Remove package	apt-get remove pkg-name	yum erase pkg-name	zypper remove pkg-name
Install from package file	dpkg -i pkg-name	yum localinstall pkg-name	zypper install pkg-name

Tampere University

There can be multiple plays



```yaml
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

Tampere University

The exercise in short

- Read Ansible tutorial to understand how it works. A good starting point is: https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html
- Prepare a docker image that can be used as a target. See details in below.
- Install Ansible in your computer.
- Make simple playbook
  - Check that the image has the latest version of git version management system
  - Queries the uptime (Linux command uptime) of target host
- Return result to Plus

Testing your Ansible

1. Start one container from the image, get its IP-address.
   (in case of password-based authentication you need a manual login after start)
2. Ensure that the IP address is in /etc/ansible/hosts
3. Run the playbook
4. Copy the output (O1)
5. Run the playbook again
6. Copy that output, too (O2)
7. Start a second contained from the image, get its IP-address.
8. Ensure that this IP address is in /etc/ansible/hosts, too.
9. Run the playbook
10. Copy the output (O3)
11. Run the playbook again
12. Copy that output, too (O4)

Submission

- Git link of the code (teacher may want do git clone). Use branch “ansible”.
- The report should have a “report.pdf” with the following contents.
  - All the copied output (O1,O2,O3,O4)
  - Comments on what was easy and what was difficult.