VU CLOUD COMPUTING, 2025W

Assignment 1: **Airport Security System**

Assignment #1: Technologies

Programming Languages

→ **Any language in which you can finish the task**

Basic knowledge of web and related technologies like

→ **HTTP**, **JSON, …**

→ git (collaboration and versioning)

→ curl, or postman, ... (for testing your API)

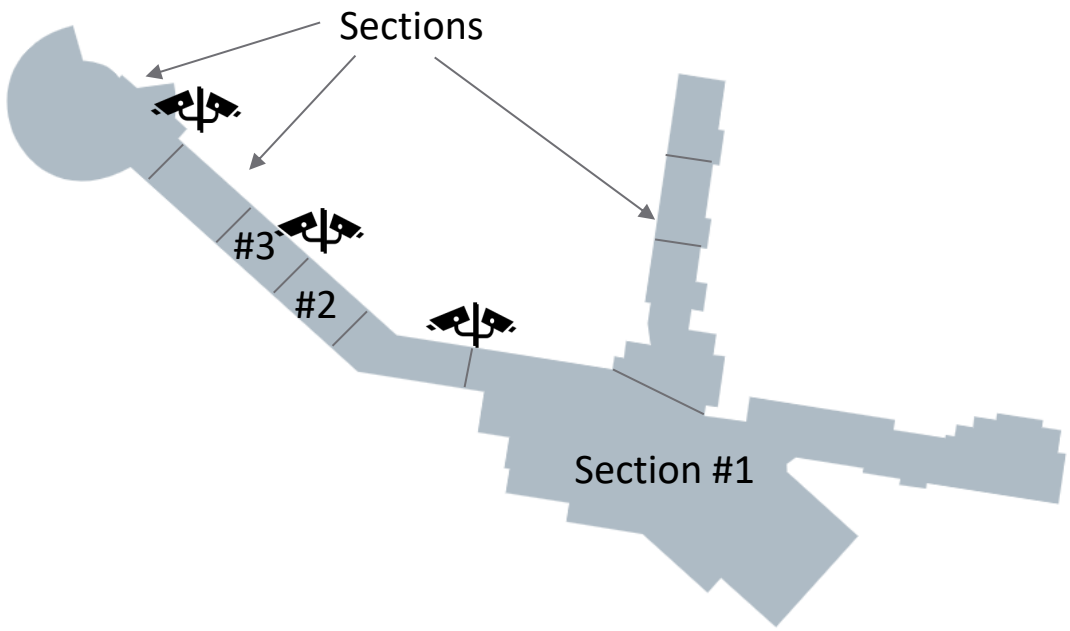
Basic Linux

Basic Google Cloud Platform

Docker, **Kubernetes**

Also: Grafana, Apache jMeter (optional)

ASSIGNMENT #1: AIRPORT SECURITY SYSTEM



Example: airport terminal

Surveillance cameras detect movement and stream images to an endpoint, they need to be processed “on-the-fly"

Each section’s entry or exit is covered

Count the number of persons entering and exiting each **section**

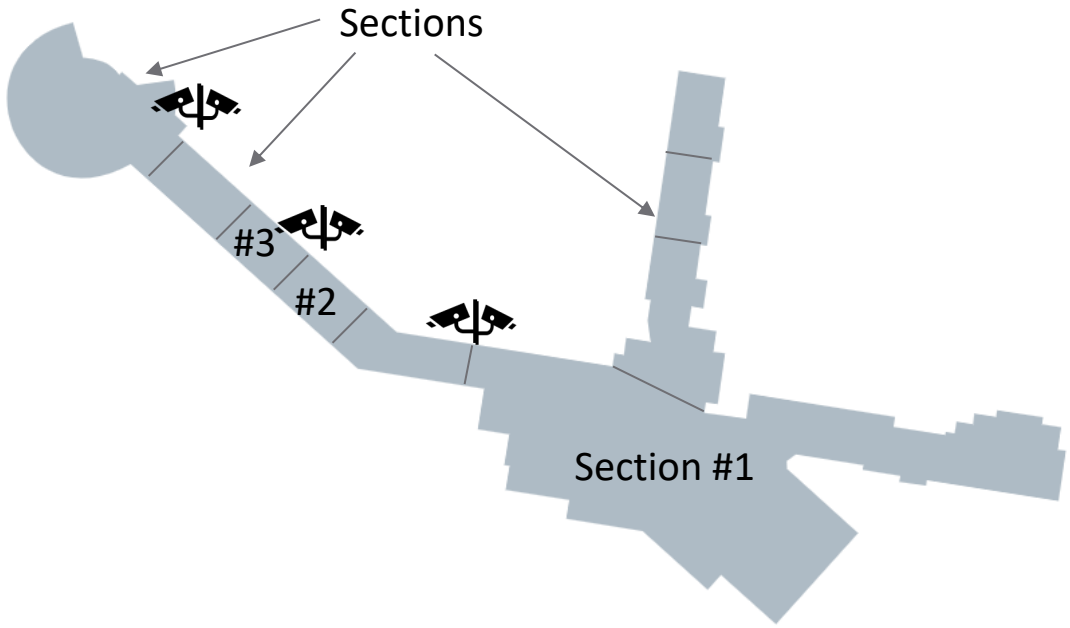
Use recognition software to check if there are some persons of interest in the area and send an **alert**

Q.1: How to address increasing traffic? Traffic spikes?

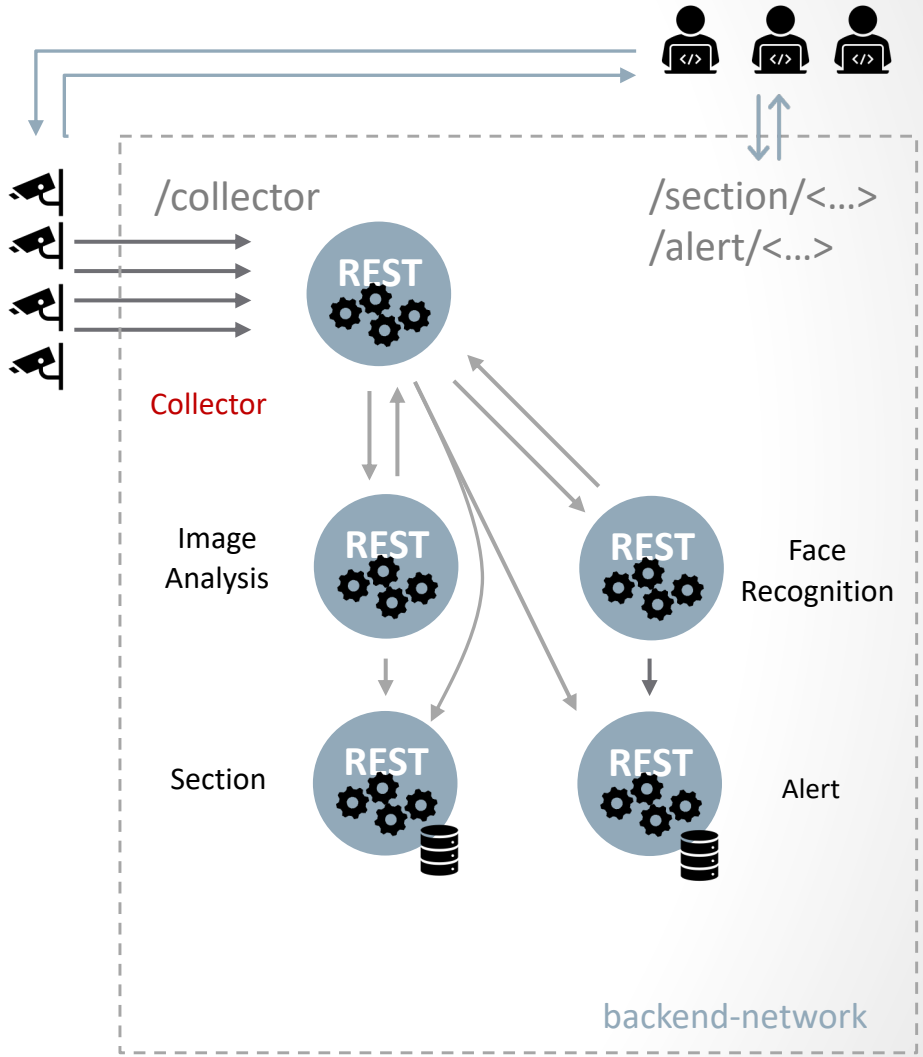
Q.2: More people → busier the services

Q.3: More Section → more cameras → busier the services

ASSIGNMENT #1: AIRPORT SECURITY SYSTEM



Example: airport terminal



ASSIGNMENT #1: SERVICES

1. **Camera**
   A surveillance camera, capturing an image each second and and streams it to target
2. **FaceRecognition Service**
   Compares an image against the list of known persons for possible matches
3. **Section Service(s)**
   Manages and provides statistical data for section(s)
4. **Alert Service:**
   Manages and provides information about detected persons of interest
5. **Image Analysis Service**
   Estimation of age and gender of people on an image
6. **Collector Service** ← Implemented by you
   Receives images from Camera, and forwards them for processing

This is not a full description - read more in Moodle

Assignment #1: Airport Security System

System configuration

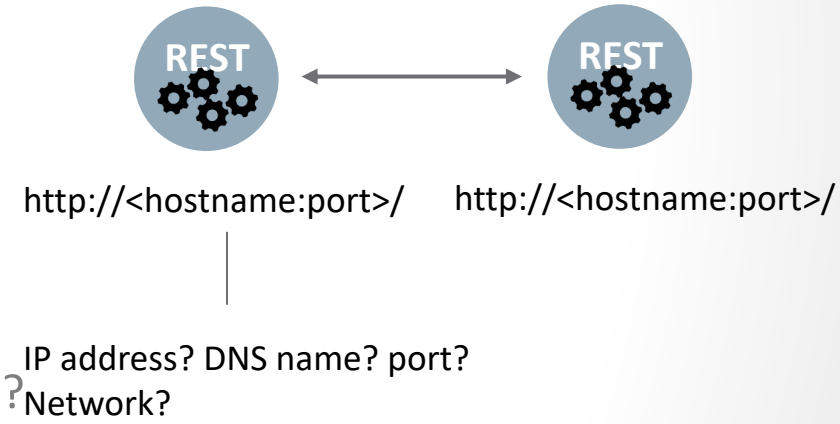
- Remember to describe the configuration of your system (in Kubernetes)

Service discovery

- Use existing Kubernetes objects to discover services

Communication / Data flow

- Your decision but explain what you did



Scalability

- How the system behaves when the number of cameras increase?

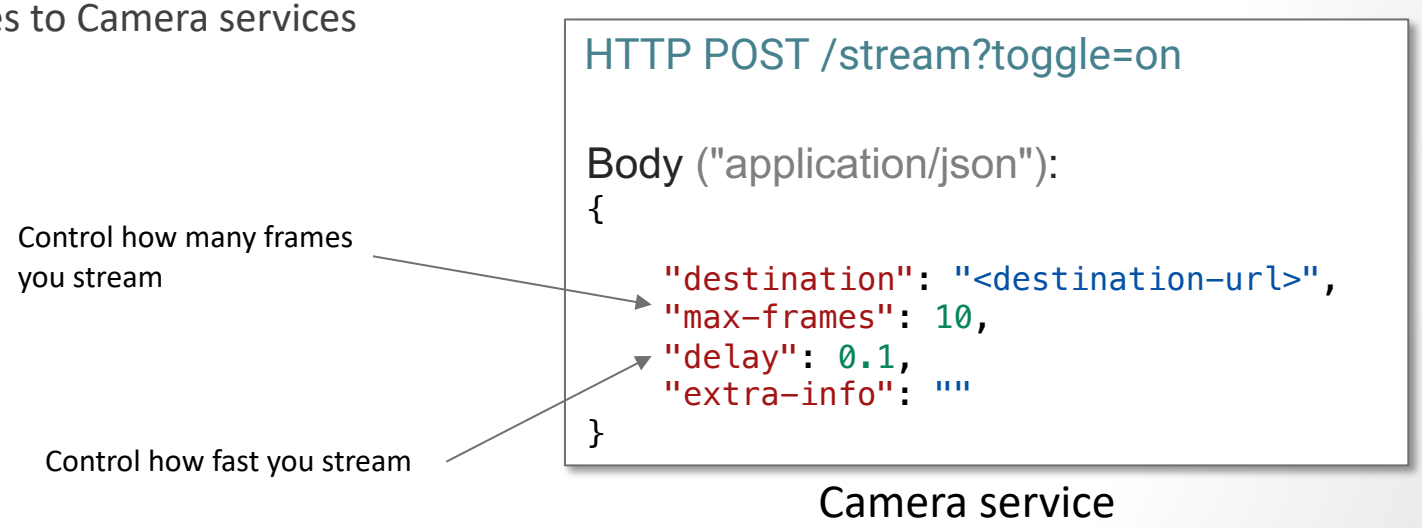
Bottlenecks?

- Find services are the most time consuming,
  and check if communication is causing some performance issues

CLOUD USAGE

You should not spend all your resources on the first two exercises, threfore make sure that:

- **You set limits to your cloud credit usage**
  - https://cloud.google.com/apis/docs/capping-api-usage
- **If you start your services, don't leave them running indefinitelly**
  - You can use max-frames to set limits on how many frame each camera sends
  - You can use the delay to slow down the stream when developing but remove the delay when testing for scalability
- **Remarks:**
  - You can provide environment variables to Camera services



ALREADY IMPLEMENTED SERVICES

You can a started template from Moodle

- **Moodle -> Assignments -> a1-starter-template**

Its easier to get started locally with Compose, you can go the report where docker-compose.yml file is and run:

`docker-compose up` **(This will pull some already implemented services for you)**

*You can put your and other services there and have the whole system running with one command.

Treat already implemented services as a black box and use in containers

- ccuni/camera-service-2025w
- ccuni/image-analysis-service-2025w
- ccuni/face-recognition-service-2025w
- ccuni/section-service-2025w
- ccuni/alert-service-2025w

Submitting the task

Use **Git** → all submissions go to your **git** repository (clone with HTTPS), which should look like:

https://gitta-lab.cs.univie.ac.at/cc25w/a<Matr.Nr.>

*you should have received an email with the exact location of your repository.

- You should "**push**" all your code there, also it is a good practice to push your code frequently, not only when everything is done. You code will only be checked after the deadline.
- You can use online interface, or any other **git** client to submit the task. You can also use git features to ask questions about your code.

Submission

- Source codes to the Gitlab only!
- Documentation (Git and Moodle)
  - You cannot score above 25% without a report
  - Check details in the assignment sheet

Resources, Tutorial

1. Tutorials linked in Moodle platform
   - https://gitta-lab.cs.univie.ac.at/public-examples/cloud-computing-extra-materials
2. Docker - https://docs.docker.com/get-started/
3. Docker - https://github.com/docker/labs/tree/master/beginner/
4. Python Flask https://www.tutorialspoint.com/flask/index.htm
5. Guides for GKE: https://cloud.google.com/kubernetes-engine/docs/quickstart
6. Guides for Cloud Run: https://cloud.google.com/run/docs/quickstarts/build-and-deploy#python

For Java-based approach (typically takes longer to get started than with python)

1. Jersey documentation - https://jersey.java.net/documentation/latest/jaxrs-resources.html
2. IBM RESTful Web Services: The basics - http://www.ibm.com/developerworks/library/ws-restful/
3. REST Web Service Tutorial, with Setup - https://www.tutorialspoint.com/restful/restful_quick_guide.htm
4. REST Web Service Tutorial with Jersey - http://www.vogella.com/tutorials/REST/article.html
5. JAX-RS Endpoint Lifecycle Explained - https://github.com/stoicflame/enunciate/wiki/JAX-RS-Lifecycle
6. JAXB Tutorial - http://www.vogella.com/tutorials/JAXB/article.html
7. Eclipse WTP - http://www.vogella.com/tutorials/EclipseWTP/article.html
8. Gradle - http://www.vogella.com/tutorials/EclipseGradle/article.html