UPPSALA
UNIVERSITET

Lecture-5: Distributed Infrastructures (Part-II)

Salman Toor

salman.toor@it.uu.se

UPPSALA
UNIVERSITET

Distributed Computing Infrastructures: Part-I

- Infrastructure as Code (IaC)
  - Contextualization
  - Orchestration
- Kubernetes
- Application-as-a-Package
  - Helm charts

UPPSALA
UNIVERSITET

Distributed Computing Infrastructures: Part-II

- Serverless architecture
  - FaaS
- Continuous Integration and Continuous Delivery (CI/CD)
  - Git Hooks
- Model Serving
  - Model as Code (MaC)
  - Model as Data (MaD)
- Lab-2/Assignment discussion
- Case studies
  - KubeNow
  - Stackn

UPPSALA
UNIVERSITET

Serverless architecture

Serverless architecture is a concept to build and run software and services without having to manage servers. In serverless architecture, the server will manage by provider and not by consumer.

https://blog.runcloud.io/understand-serverless-computing/

https://link.springer.com/chapter/10.1007/978-981-10-5026-8_1

UPPSALA
UNIVERSITET

FaaS: Function as a Service

- FaaS is a realization of the serverless architecture
- FaaS allows to execute code in response to events without the complex infrastructure typically associated with building and launching microservices applications.
- FaaS consists of lightweight functions that subdivide the application's logic and only executed based on generated events.

https://www.ibm.com/cloud/learn/faas

A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms

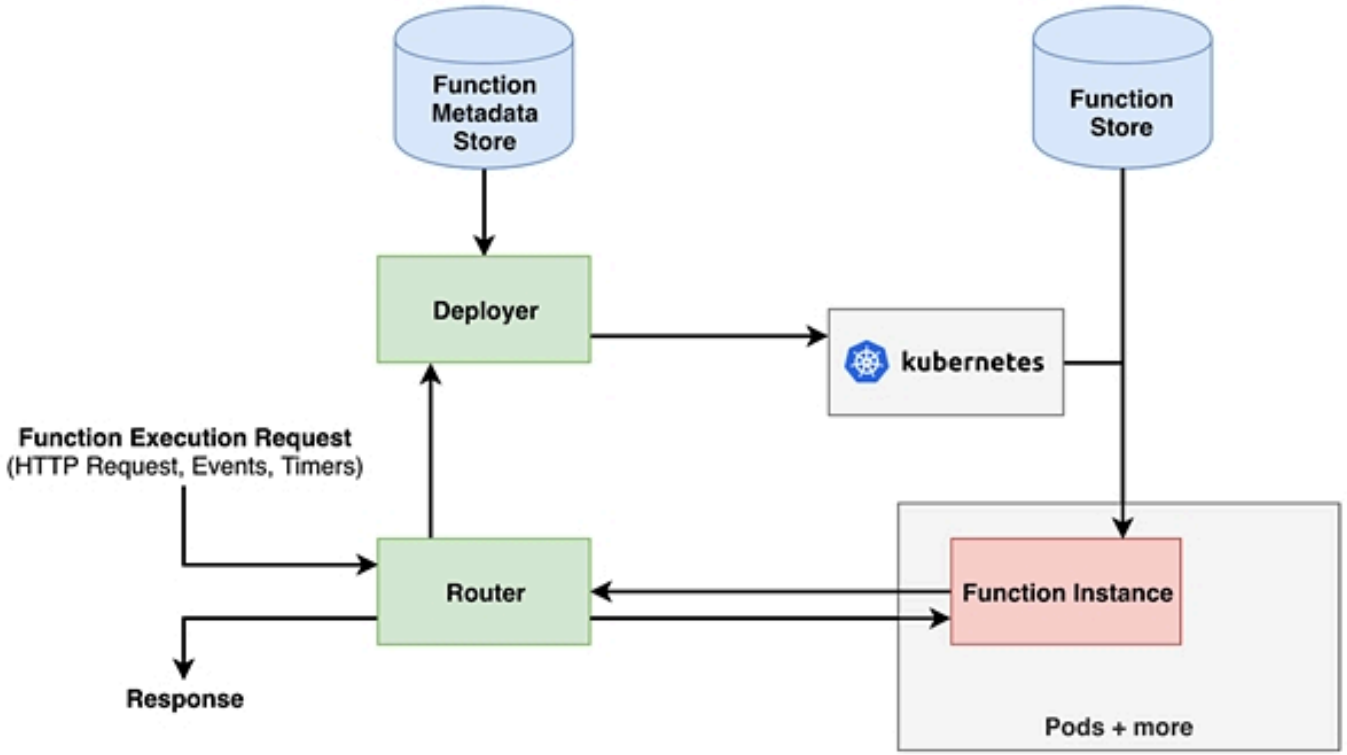
UPPSALA
UNIVERSITET

FaaS: Function as a Service

- Characteristics:
  - Rapid development
  - Event driven execution
  - Cost effective computing environment (Pay-as-you-*run*)
  - No resource selection, configuration and management
  - Better elastic behaviour

UPPSALA UNIVERSITET

FaaS: Function as a Service



The anatomy of the runtime of a FaaS platform.

https://www.infoq.com/articles/serverless-performance-cost/

UPPSALA
UNIVERSITET

FaaS: Function as a Service

- FaaS Platforms
  - Firecracker, AWS Lambda and AWS Fargate (Amazon)
  - Fission (Platform 9)
  - Fn (Oracle)
  - Knative/Kubernetes (Google)
  - OpenFaaS

https://firecracker-microvm.github.io/ https://fission.io/ https://fnproject.io/ https://knative.dev/ https://github.com/openfaas/faas

UPPSALA
UNIVERSITET

Continuous Integration and Continuous Delivery (CI/CD)

- Continuous Integration (CI)
  - A software engineering process of avoiding enormous code conflicts at the end of a development cycle

- Continuous Delivery (CD)
  - A process of ensuring that software is always ready to be deployed
  - Continuous Delivery requires Continuous Integration

Continuous Delivery: https://martinfowler.com/books/continuousDelivery.html

https://link.springer.com/chapter/10.1007/978-3-319-97925-0_33

UPPSALA
UNIVERSITET

Continuous Integration(CI)

- Benefits
  - Systematic development with less number of bugs
  - Building the release is easy
  - Reduce the development time
  - Testing costs are reduced, a CI server can run hundreds of tests in parallel
  - Less time required by the quality assurance team, can focus on overall system improvement than minor application related details.

Continuous Delivery: https://martinfowler.com/books/continuousDelivery.html

https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

UPPSALA
UNIVERSITET

Continuous Delivery(CD)

- Benefits
  - Faster and systematic deployment with checkpointing
  - Reduce the transition time between development and production environments
  - Frequent updates are possible in production pipelines
  - Increase confidence in software releases

Continuous Delivery: https://martinfowler.com/books/continuousDelivery.html

https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

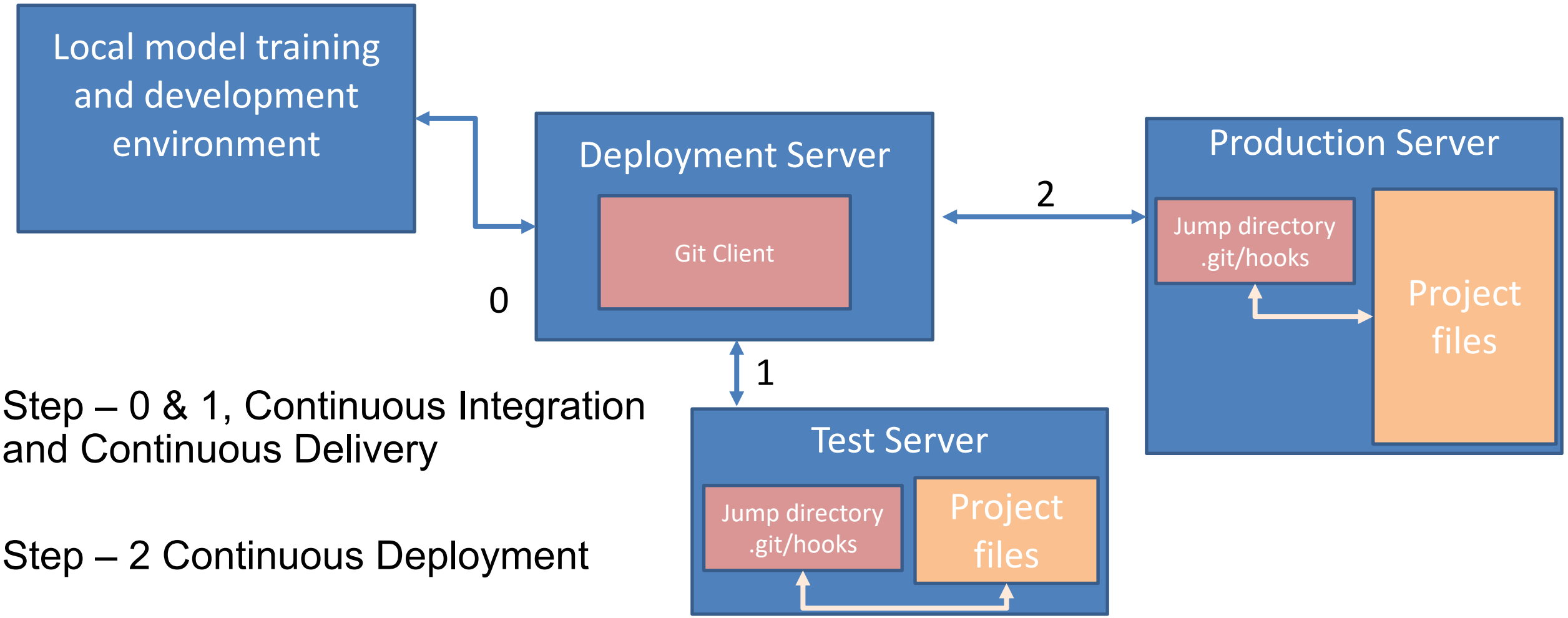
UPPSALA
UNIVERSITET

Git Hooks

- Custom software modules that run automatically every time a particular event occurs in a Git repository
- Use to customize the Git's internal behavior to build reliable and systematic execution pipeline
- *hooks* directory is by default available in every git repo ".git/hooks"
- Some of the samples:
  - *pre-commit.sample*: invoked just before making the commit.
  - *post-receive.sample*: invoked after the remote repository has updated.

UPPSALA UNIVERSITET

Continuous Integration and Continuous Delivery (CI/CD)



Step – 0 & 1, Continuous Integration and Continuous Delivery

Step – 2 Continuous Deployment

https://circleci.com/blog/building-strong-distributed-teams-one-pixel-at-a-time/

UPPSALA
UNIVERSITET

Tools for CI/CD development

- Jenkins: https://jenkins.io/
- CircleCI: https://circleci.com/
- TeamCity: https://www.jetbrains.com/teamcity/
- Bamboo: https://www.atlassian.com/software/bamboo
- GitLab: https://about.gitlab.com/

https://www.katalon.com/resources-center/blog/ci-cd-tools/

UPPSALA
UNIVERSITET

What is model serving?

Model serving is a recently introduced concept to offer machine learning as a service (MLaaS). Mainly designed in following two ways:

1. Dedicate inference serving (service for a particular application)
2. Open inference serving (inference service for different applications within defined settings)

The aim is to simplify the process by offering reliable, easy to use and efficient predictions at scale

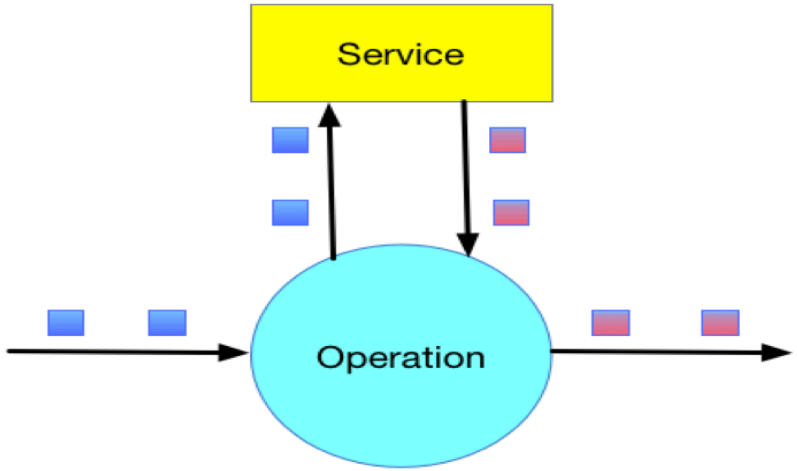
UPPSALA UNIVERSITET

Model serving and stream processing

Model servers simplify the task of deploying machine learning at scale, the same way app servers simplify the task of delivering a web app or API to end users. The rise of model servers, will likely accelerate the adoption of user-facing machine learning in the wild.

Two popular approaches for model server implementation:

- Model as Code (MaC)
- Model as Data (MaD)



https://bugra.github.io/posts/2020/5/25/how-to-serve-model/

https://www.lightbend.com/blog/machine-learning-for-cloud-native-applications-model-serving-as-a-service

UPPSALA
UNIVERSITET

Model Servers

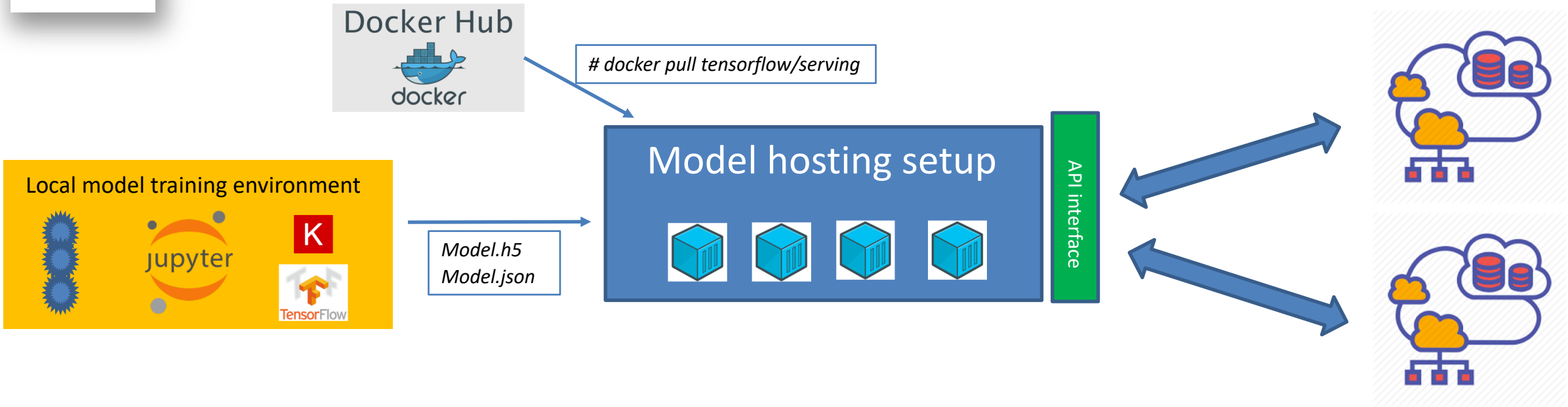
Two popular approaches for model server implementation:

- Model as Code (MaC): Enables model as a function that take input and produce inference
- Model as Data (MaD): Allows use of different runtimes for machine learning and model serving

https://www.lightbend.com/blog/machine-learning-for-cloud-native-applications-model-serving-as-a-service

UPPSALA UNIVERSITET

TensorFlow model serving



1. Save model – both weights and architecture
   – For different versions, “saved_model/<version>”, versions = 1,2 or 3

2. Load TensorFlow-serving container from the Docker Hub
3. Setup the model directory, config file, container parameters and start the container
4. Use both web interface or APIs to get the predictions

https://www.tensorflow.org/tfx/serving/serving_basic
https://hub.docker.com/r/tensorflow/serving

UPPSALA
UNIVERSITET

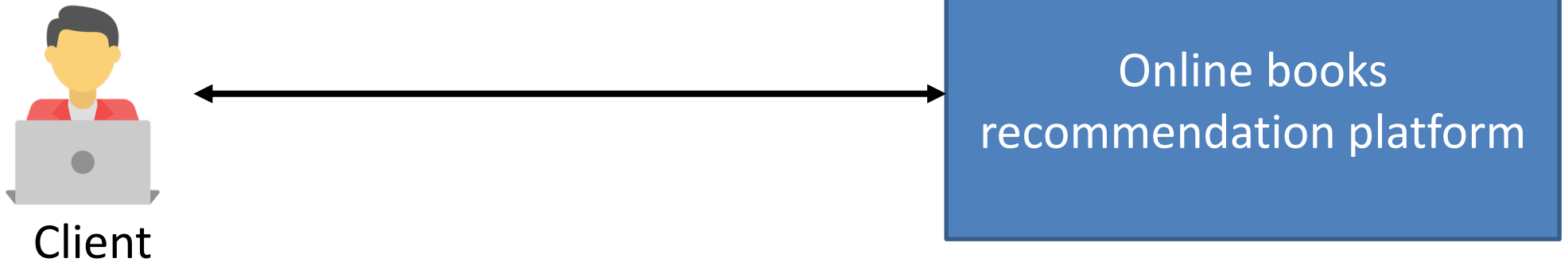
Model serving platforms

- TensorFlow Serving: http://learningsys.org/nips17/assets/papers/paper_1.pdf
- Clipper: https://www.usenix.org/system/files/conference/nsdi17/nsdi17-crankshaw.pdf
- Model Server for Apache MXNet: https://aws.amazon.com/blogs/ai/introducing-model-server-for-apache-mxnet/
- DeepDetect: https://deepdetect.com
- TensorRT: https://developer.nvidia.com/tensorrt

https://medium.com/@vikati/the-rise-of-the-model-servers-9395522b6c58

UPPSALA UNIVERSITET

Example



- Users interaction:
  - Search new books
  - Buy books
  - Get recommendations based on previous searches or purchases
  - Listen to online books
  - Advertisements for non-paying customers

UPPSALA UNIVERSITET

Deployment stage - 1

Can this architecture manages hundreds of requests per minute or second?

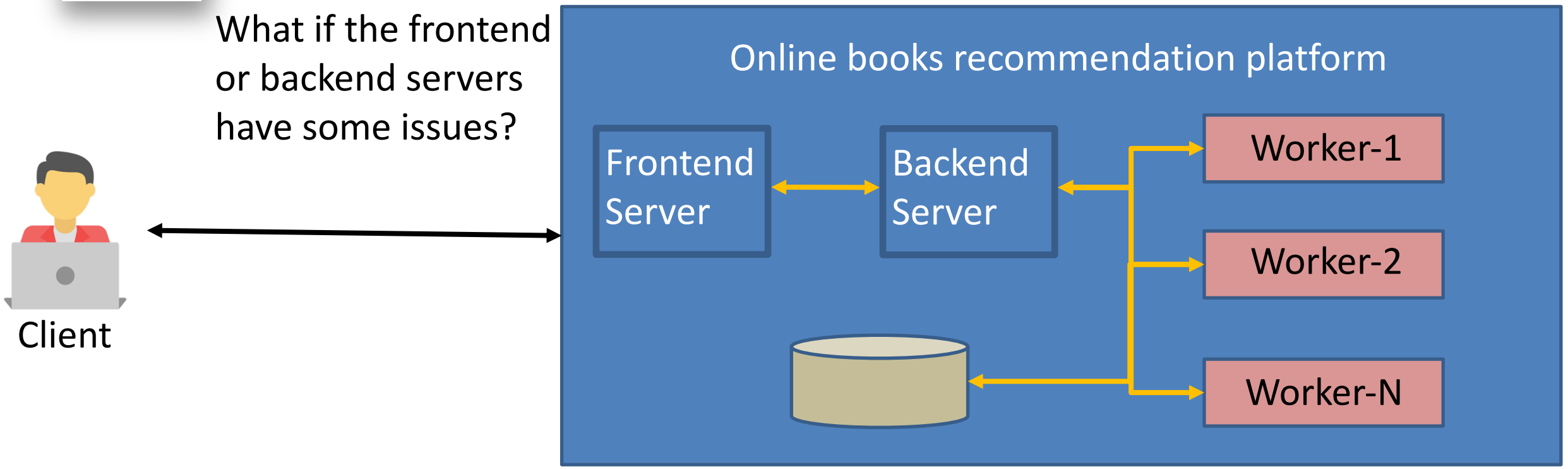
Client

- Internal architecture
  - Frontend Server, manages incoming requests
  - Backend Server, runs the logic, maintains the communication with the internal components of the application

UPPSALA UNIVERSITET

Deployment stage - 1

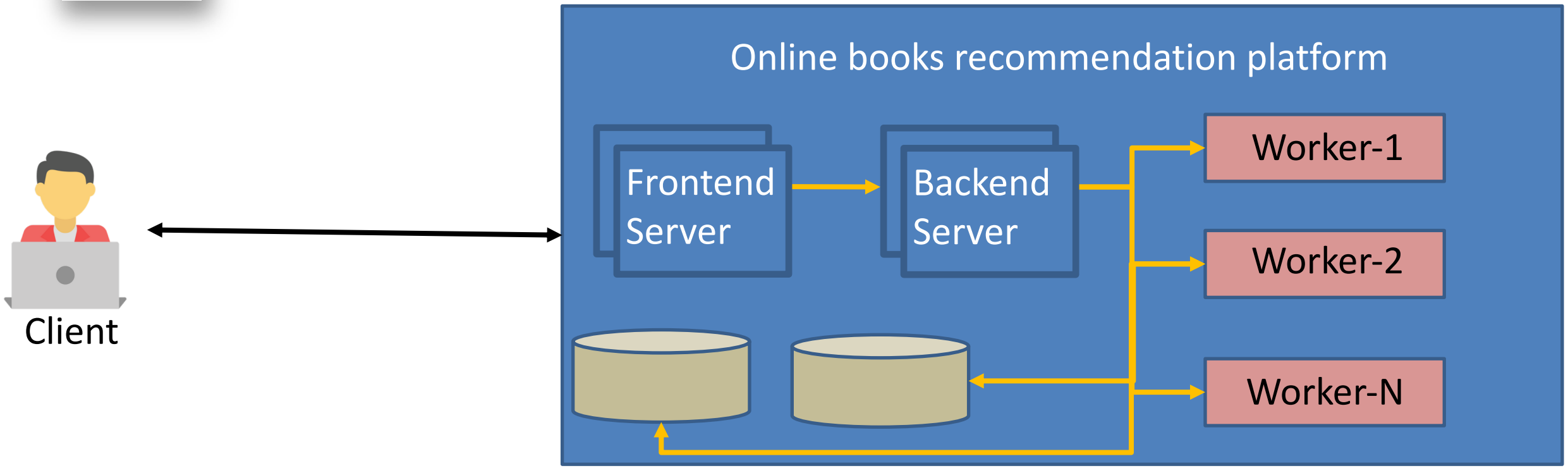
What if the frontend or backend servers have some issues?



- Internal architecture
  - Frontend Server, manages incoming requests
  - Backend Server, distribute the tasks to the available workers
  - Workers, execute the assigned tasks

UPPSALA UNIVERSITET

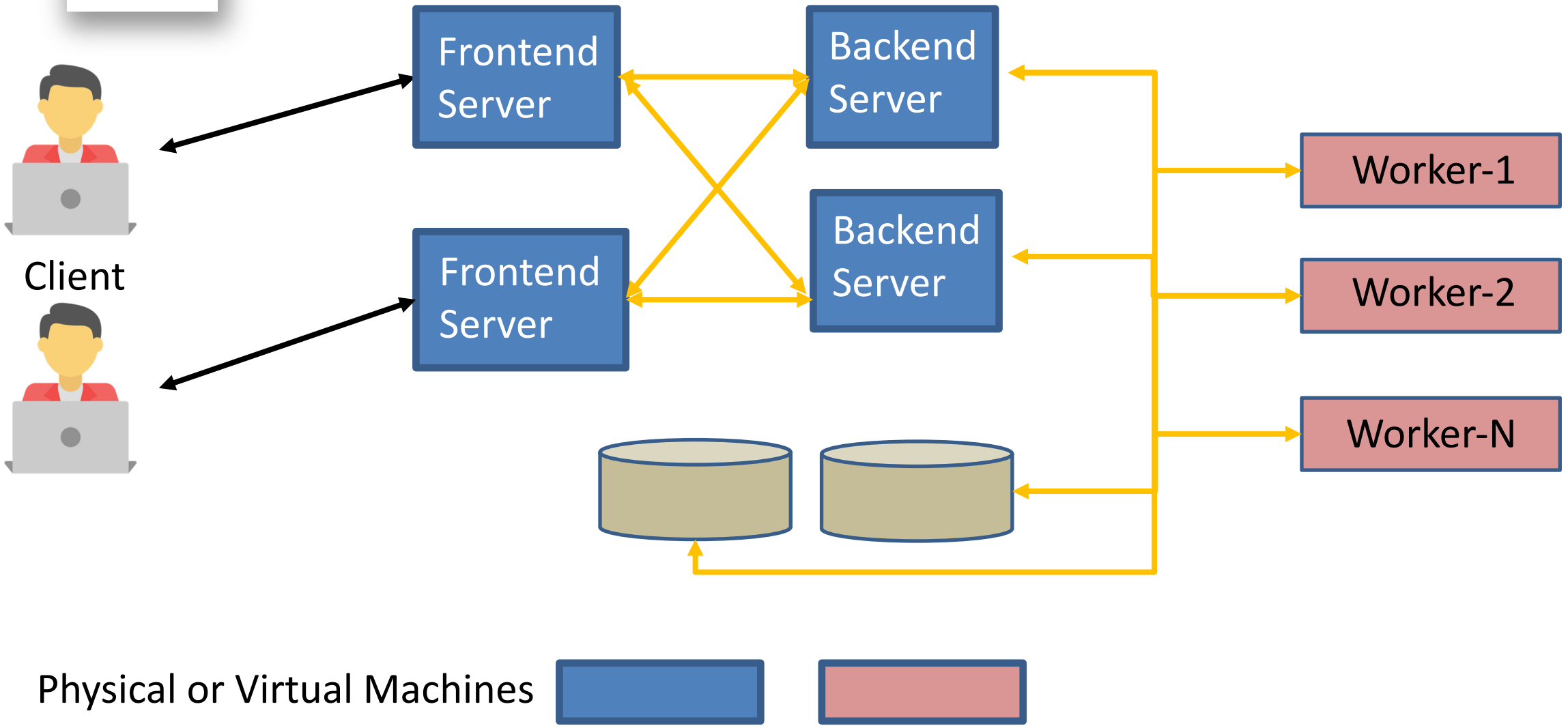
Deployment stage - 1



- Internal architecture
  - Frontend Server, manages incoming requests
  - Backend Server, distribute the tasks to the available workers
  - Workers, execute the assigned tasks

UPPSALA UNIVERSITET

Deployment stage - 2



Physical or Virtual Machines

UPPSALA
UNIVERSITET

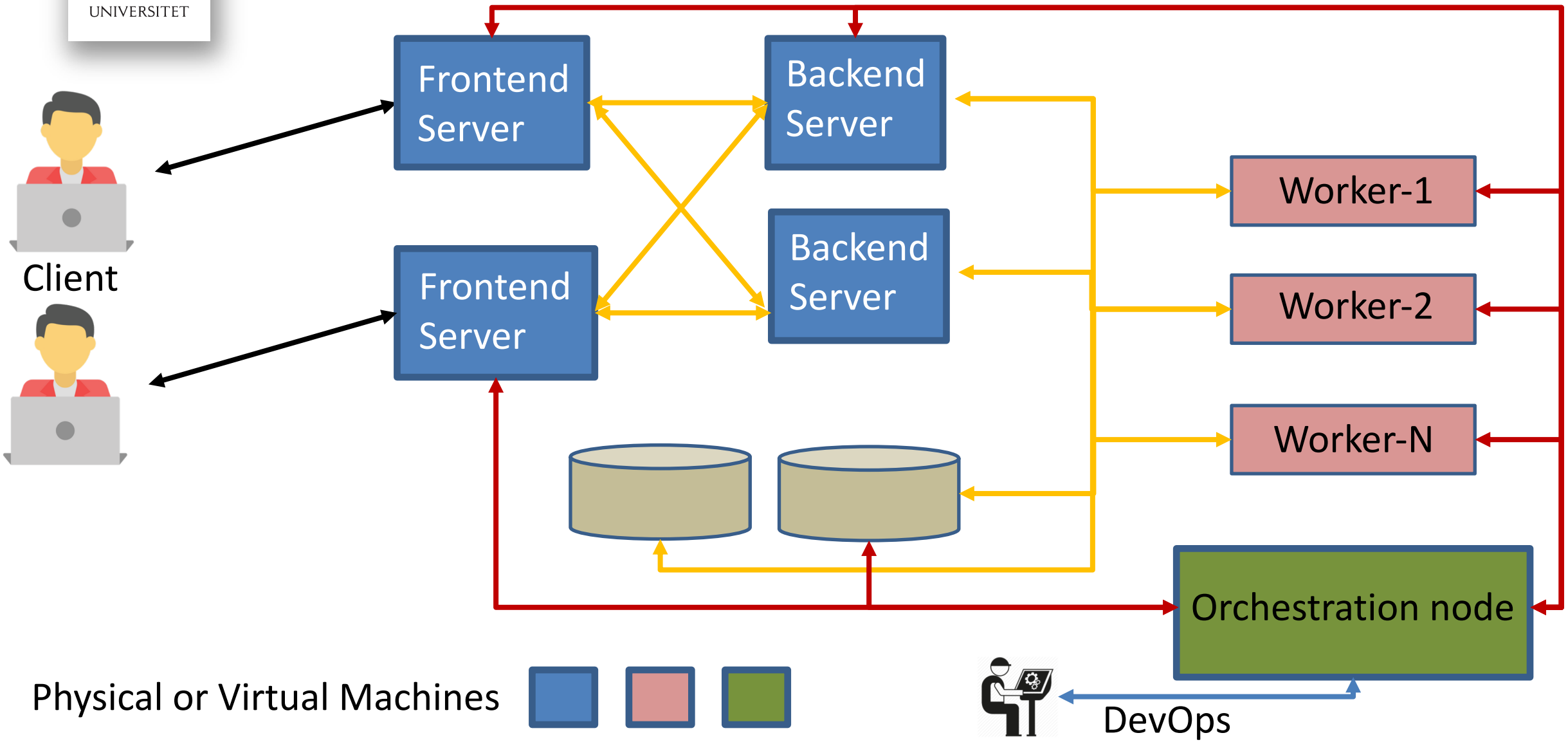
We need fast, systematic and efficient deployment

Cloud platforms are good but we need tools for

- Automation
- Contextualization
- Orchestration

UPPSALA UNIVERSITET

Deployment stage - 2



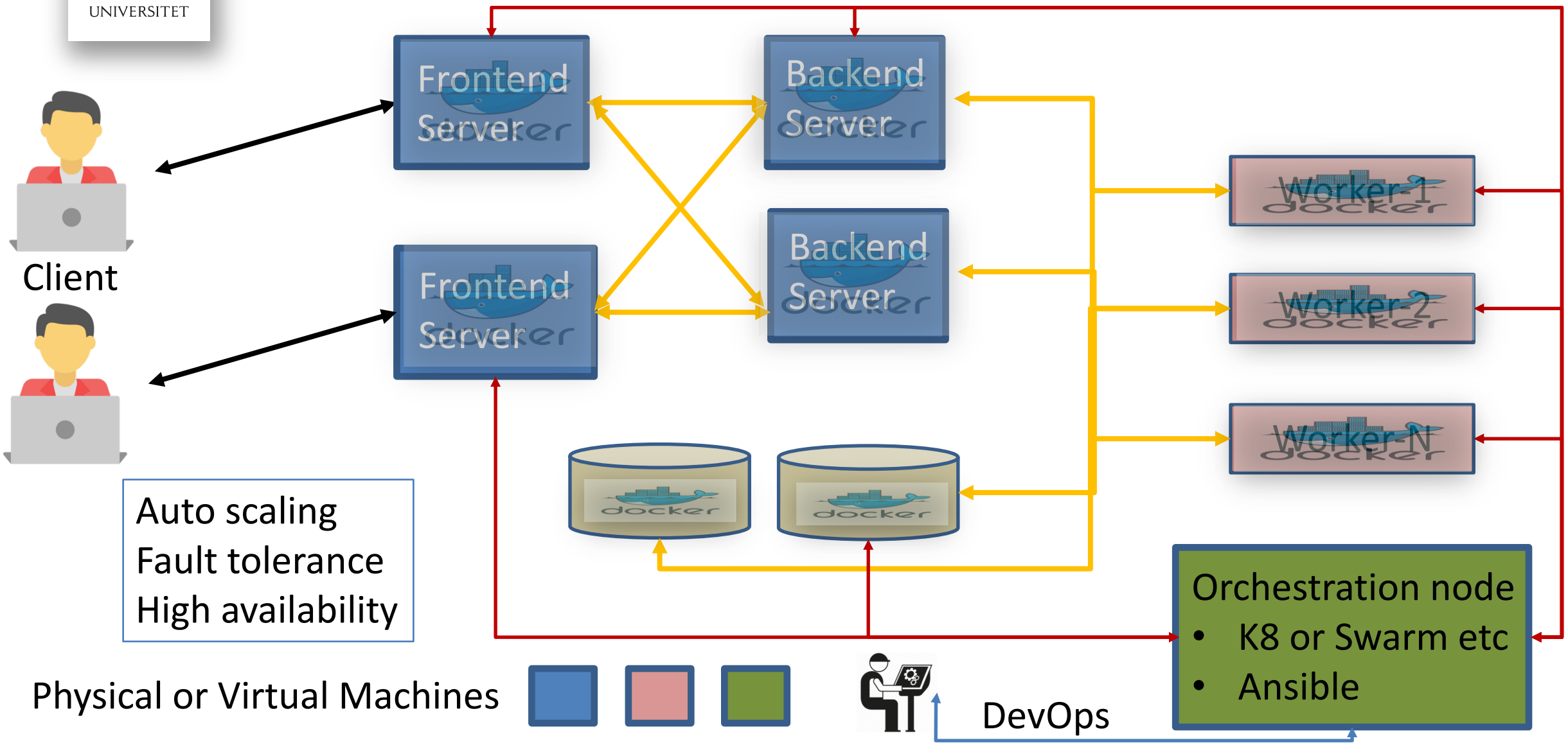
UPPSALA
UNIVERSITET

More improvements to make the platform highly scalable and fault-tolerant

- Avoid tight coupling with the host operating system
- Checkpoint and serialize each step required to make modifications
- Geographically distributed setup for high availability

UPPSALA UNIVERSITET

Deployment stage - 2



UPPSALA UNIVERSITET

Physical or Virtual Machines

Deployment stage - 3

Test Server

Local Model Development environment

Data Scientist

Frontend Server

Frontend Server

Backend Server

Backend Server

Model Deployment
GitLab + Git Hooks

Worker-1

Worker-2

Worker-N

Auto scaling
Fault tolerance
High availability
CI/CD

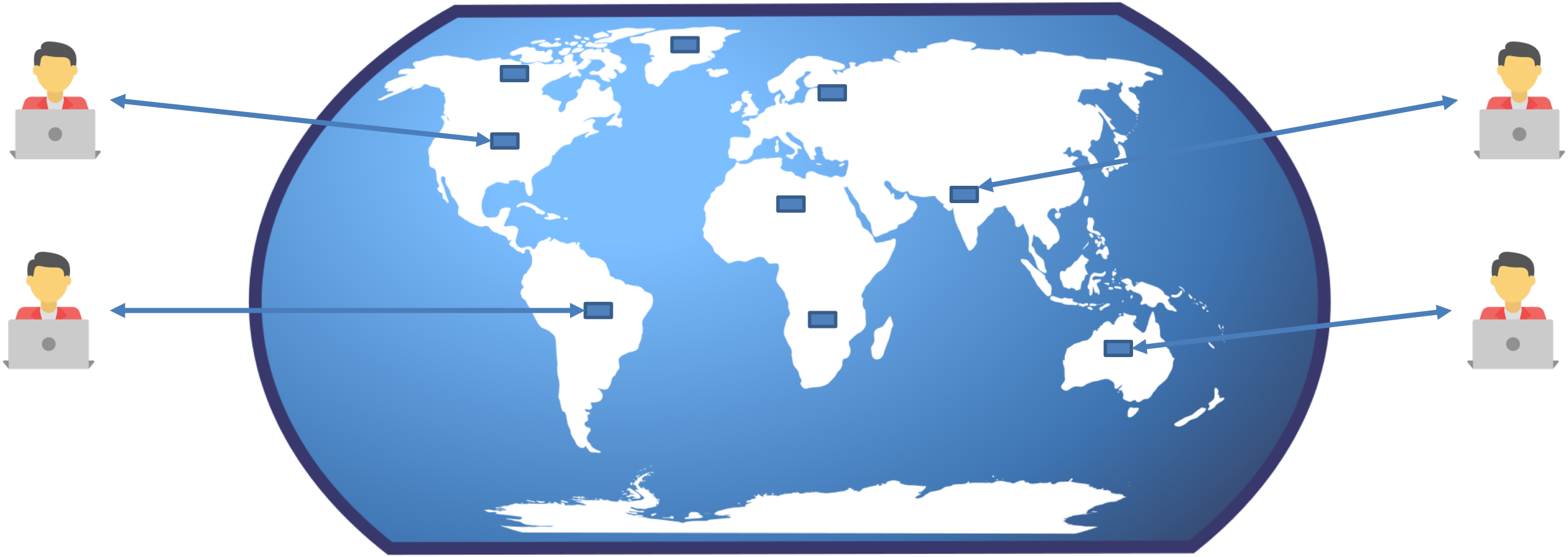
Client

DevOps

Orchestration node
- K8 or Swarm etc
- Ansible

UPPSALA
UNIVERSITET

Deployment stage - 4



Geographically distributed execution environment

UPPSALA
UNIVERSITET

Assignment

- For a simple neural network example, we have used PIMA Indian Diabetes Prediction dataset

  https://data.world/data-society/pima-indians-diabetes-database
  https://towardsdatascience.com/pima-indian-diabetes-prediction-7573698bd5fe

- The objective is to predict based on diagnostic measurements whether a patient has diabetes or not

- Neural network model is used to make the predictions

https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

UPPSALA
UNIVERSITET

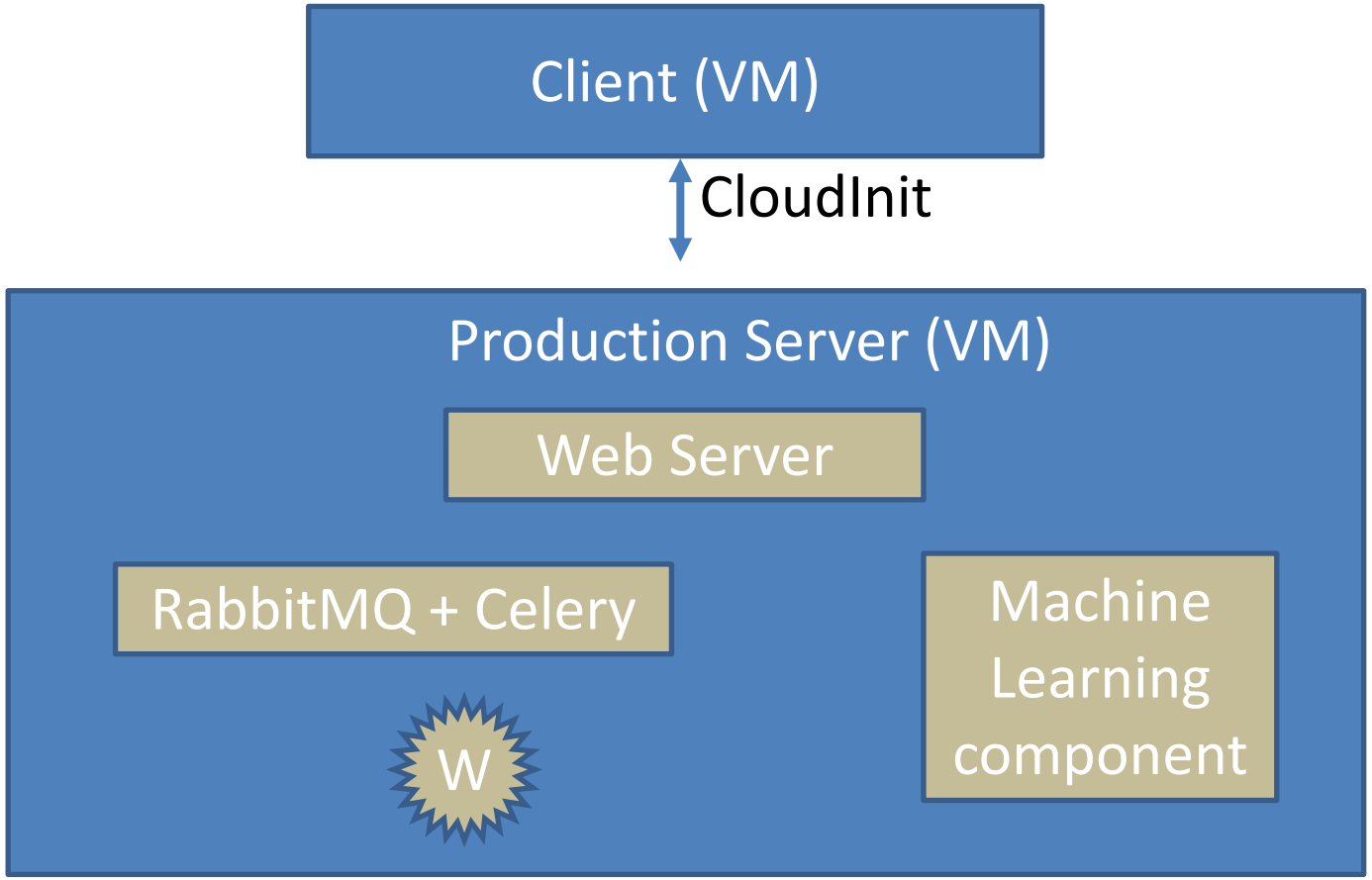
Tasks

1. Dynamic contextualization using CloudInit package
   - Single node deployment
   - Containerized application deployment using docker-compose

2. Multi-node model serving setup
   - Model serving using Ansible playbooks
   - Continuous integration and delivery using Git Hooks

UPPSALA UNIVERSITET

Task-1



Client contacts the cloud via OpenStack APIs and start a server by passing contextualization script.

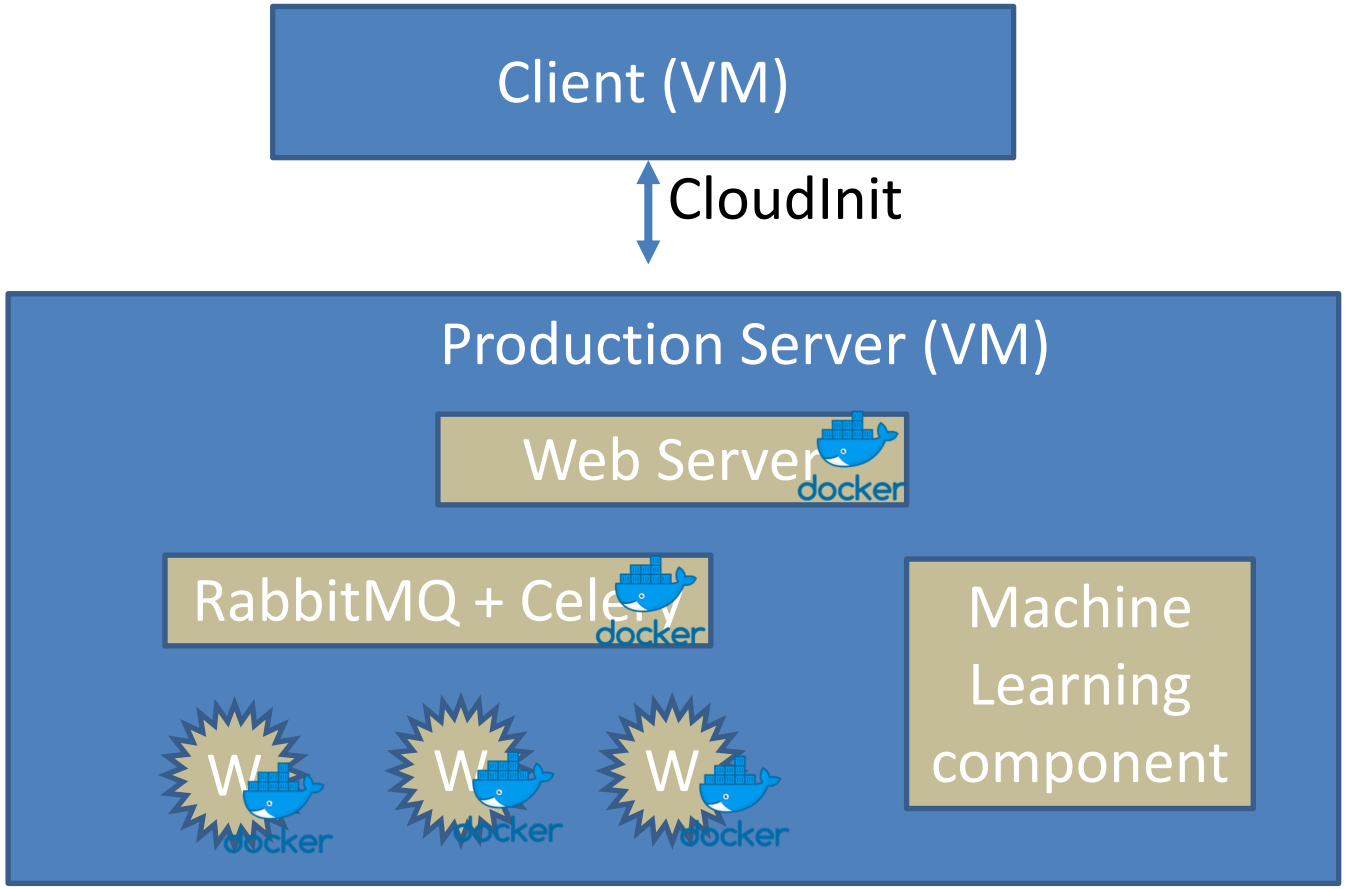
Production server hosts the complete application and provide a web interface to access the results.

Your task is to

1. start and configure a Client VM
2. send an API request from Client to initiate the Production Server
3. access the predictions via the web interface
4. answer the questions related to the Task-1 (available in Git repo)

UPPSALA UNIVERSITET

Task-2



Client contacts the cloud via OpenStack APIs and start a server by passing contextualization script.

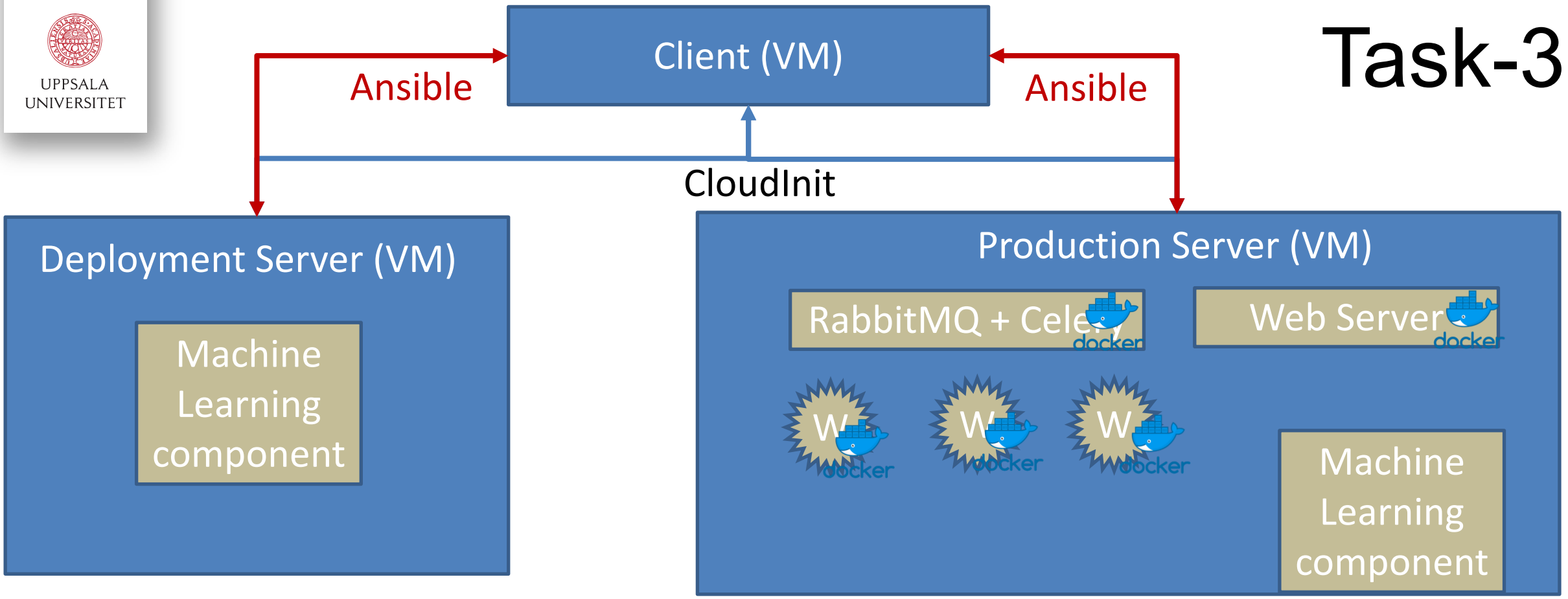
Production server hosts the complete application and provide a web interface to access the results.

Your task is to

1. send an API request from Client to initiate a **new** Production Server
2. access the predictions via the web interface
3. add more workers using docker commands
4. answer the questions related to the Task-2 (available in Git repo)

UPPSALA UNIVERSITET

Task-3

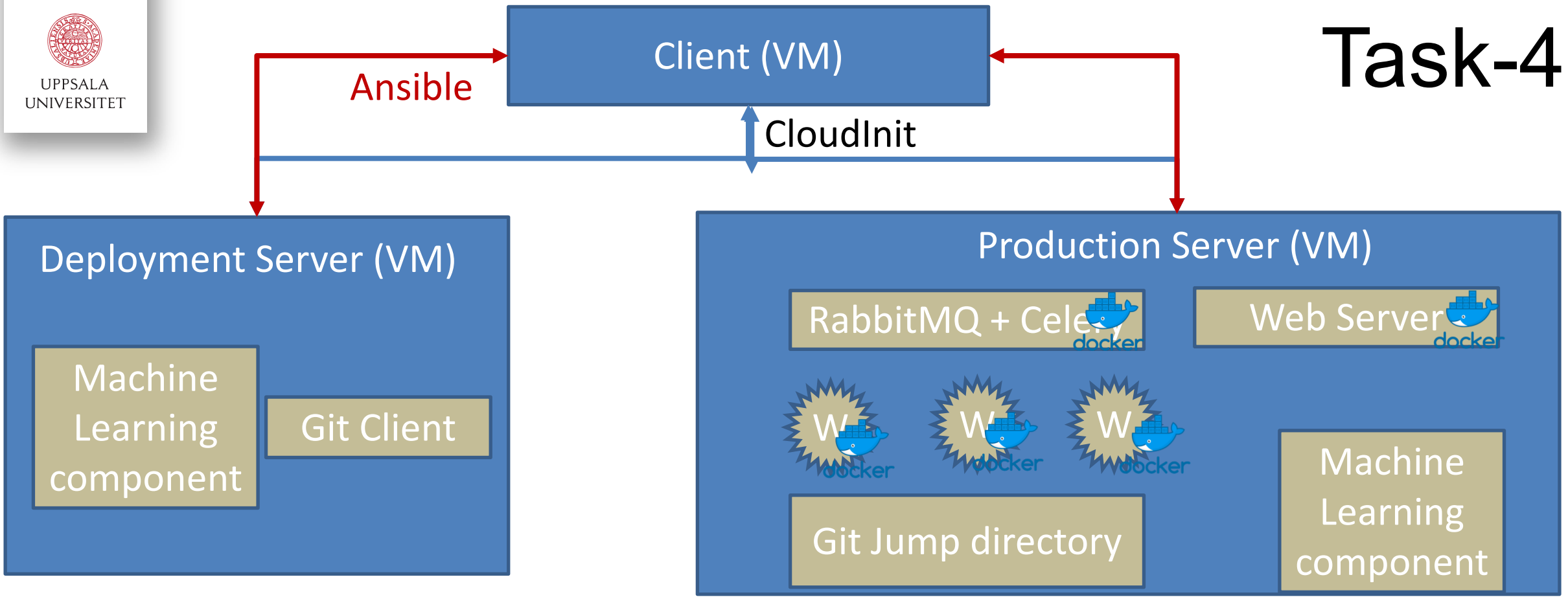


Your task is to

1. send an API request from Client to initiate both Production and Deployment Servers
2. contextualize the servers using Ansible playbook
3. access the predictions via the web interface
4. answer the questions related to the Task-3 (available in Git repo)

UPPSALA UNIVERSITET

Task-4



Your task is to

1. use the environment created in Task-3
2. Setup the Git Hook communication between Deployment and Production Servers
3. Make the changes in the machine learning code and access changes via web interface.
4. Answer the questions related to the Task-4 (available in Git repo)

UPPSALA
UNIVERSITET

Case Study: Stackn

https://github.com/scaleoutsystems/stackn

https://scaleoutsystems.com/stackn

UPPSALA
UNIVERSITET

Stackn

- STACKn is an open source collaborative ML platform
- It provides a highly flexible, cloud-native open toolkit for full-stack data science projects and the entire machine learning lifecycle.
- Key benefits:
  - Continuous Analytics
  - End-to-end workflow orchestration
  - Simple deployment
  - Open source (Apache2)

UPPSALA
UNIVERSITET

Stackn

Demo

UPPSALA
UNIVERSITET

Case Study: KubeNow

https://github.com/kubenow/KubeNow

On-demand virtual research environments using microservices

Capuccini M, Larsson A, Carone M, Novella JA, Sadawi N, Gao J, Toor S, Spjuth O. 2019. On-demand virtual research environments using microservices. *PeerJ Computer Science* 5:e232 https://doi.org/10.7717/peerj-cs.232

UPPSALA
UNIVERSITET

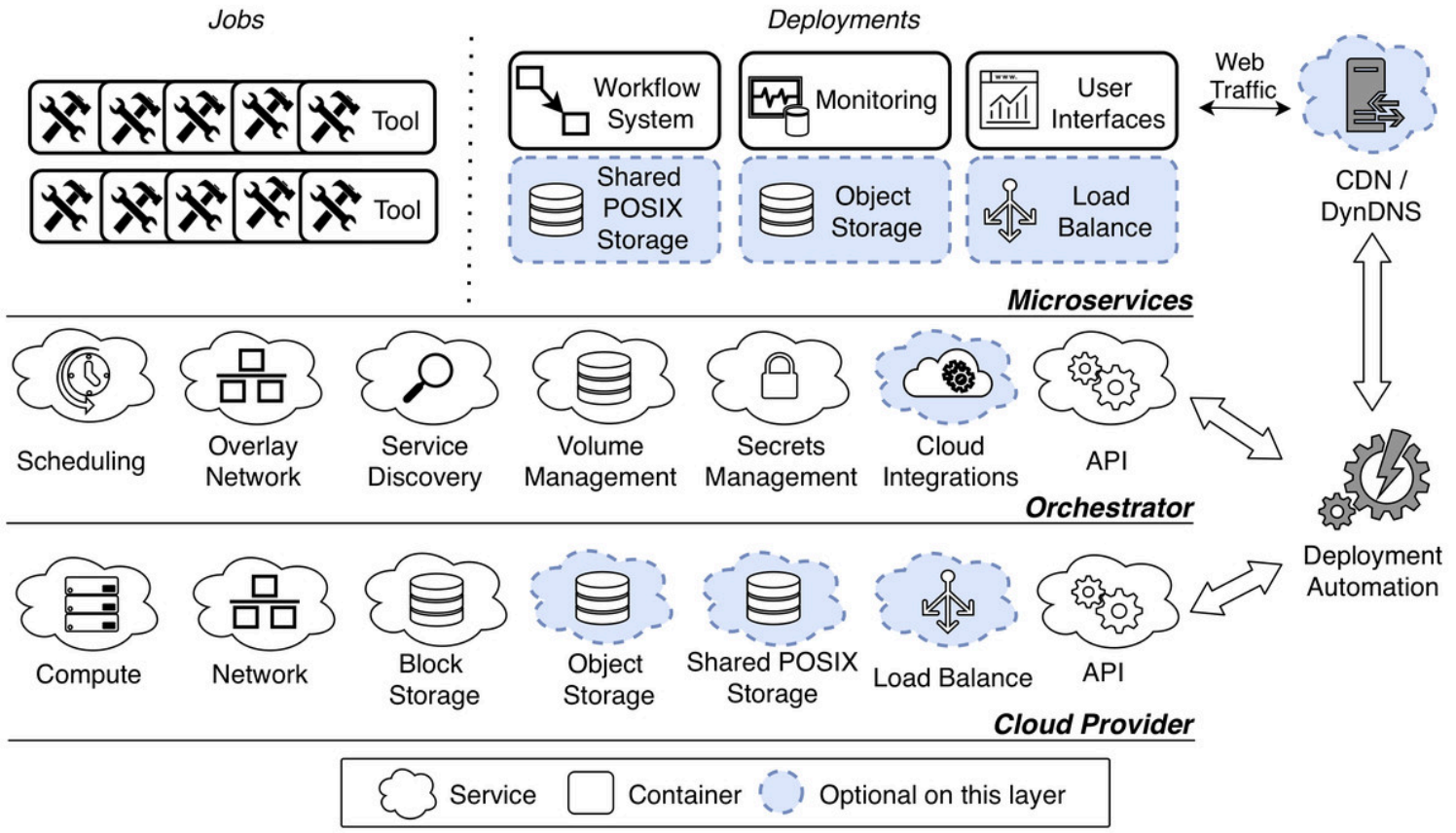
KubeNow

- Infrastructure-as-a-Service (IaaS) does not achieve the degree of flexibility required by the scientific community
- The framework enables scientific applications to run in a distributed orchestration platform as software containers, referred to as on-demand, virtual research environments (VRE)
- KubeNow is vendor-agnostic, enables on-demand VREs on the major cloud providers

UPPSALA UNIVERSITET

KubeNow architecture

- **Deployment automation**
- **Content delivery network and dynamic domain name system**
- **Microservices**
- **Orchestrator**
- **Cloud Provider**



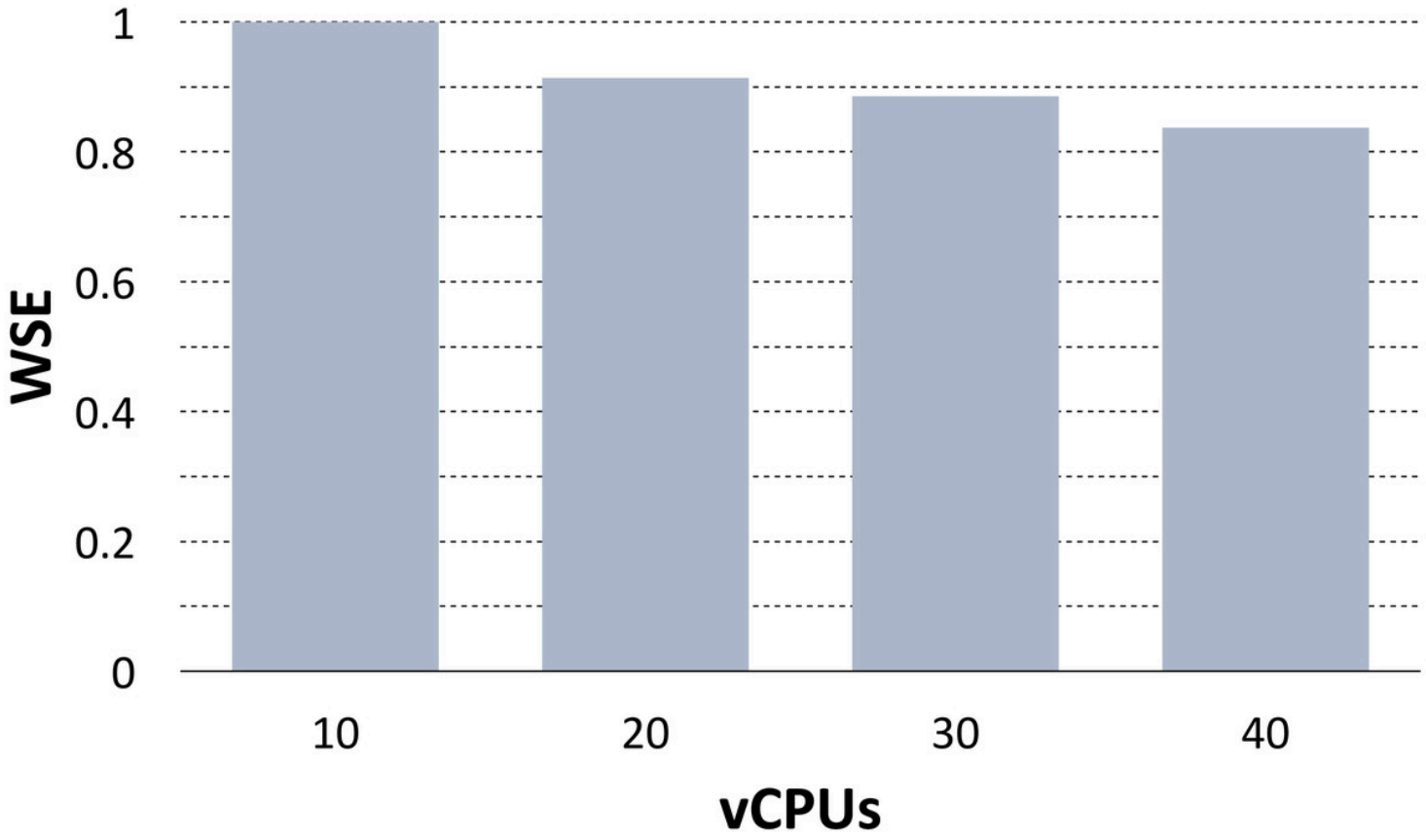
UPPSALA UNIVERSITET

KubeNow, framework evaluation

- Weak Scale Efficiency (WSE)

$$WSE = T10 / TN$$

where T10 was the measured running time on 10 vCPUs and TN was the measured running time on N vCPUs



UPPSALA
UNIVERSITET

KubeNow tools

- **Docker:** the open source de facto standard container engine.
- **Kubernetes:** the orchestration platform that has collected the largest open source community.
- **GlusterFS:** an open-source distributed file system that provides both shared POSIX file spaces and object storage.
- **Traefik:** an open-source HTTP reverse proxy and load balancer.
- **Cloudflare:** a service that provides CDN and DynDNS.
- **Terraform:** an open-source IaC tool that enables provisioning at infrastructure level.
- **Ansible:** an open-source automation tool that enables provisioning of VMs and Kubernetes.

UPPSALA
UNIVERSITET

KubeNow, framework evaluation

- Comparison
  - KubeNow
  - Kubespray