ENVRI FAIR

Cloud computing for developing and operating data management services (DevOps)

Dr. Zhiming Zhao

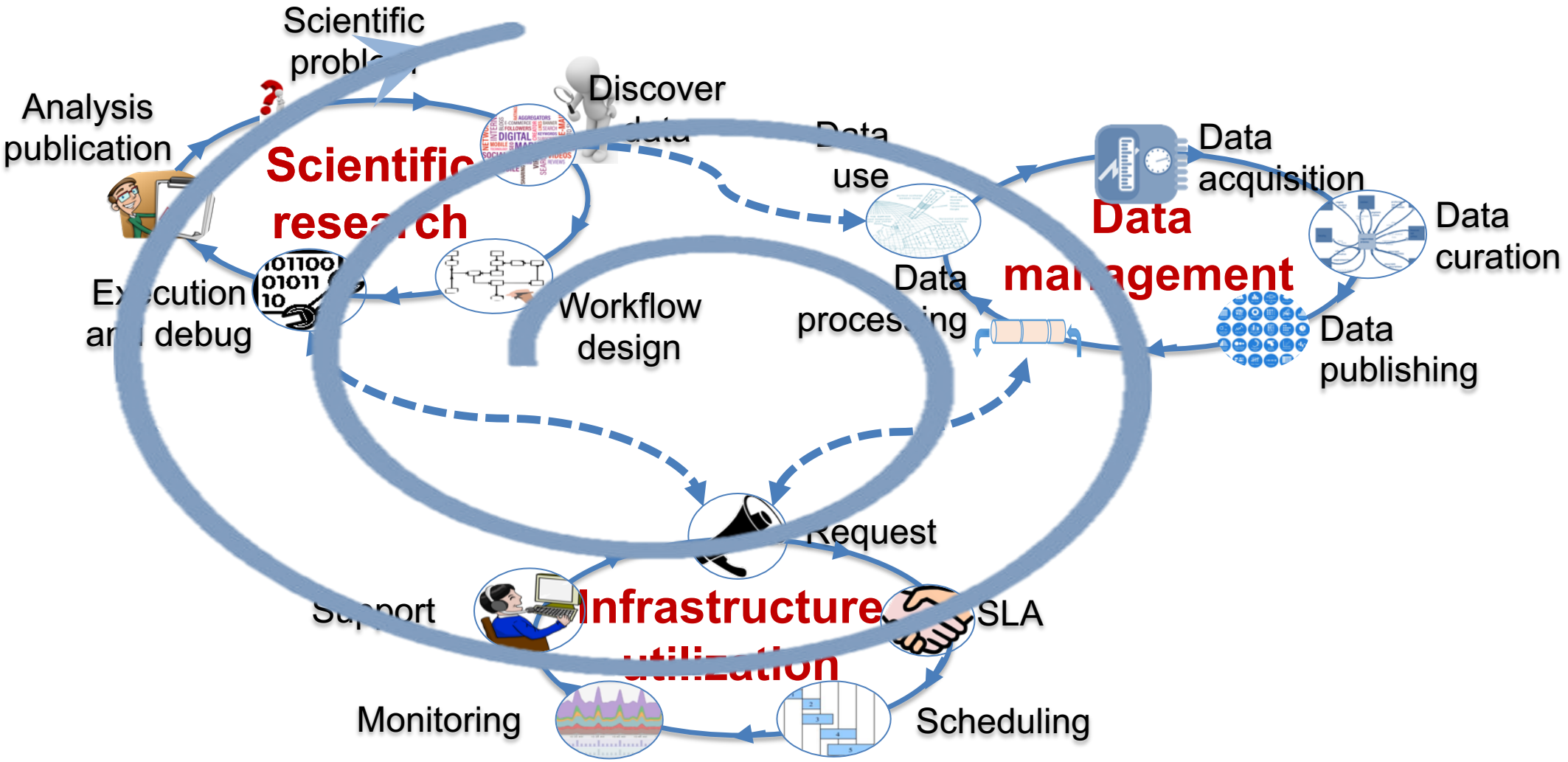
Lab support: dr. Spiros Koulouzis and mr. Riccardo Bianchi

Multiscale Networked Systems, University of Amsterdam, LifeWatch ERIC, vLabs & Innovation Centre

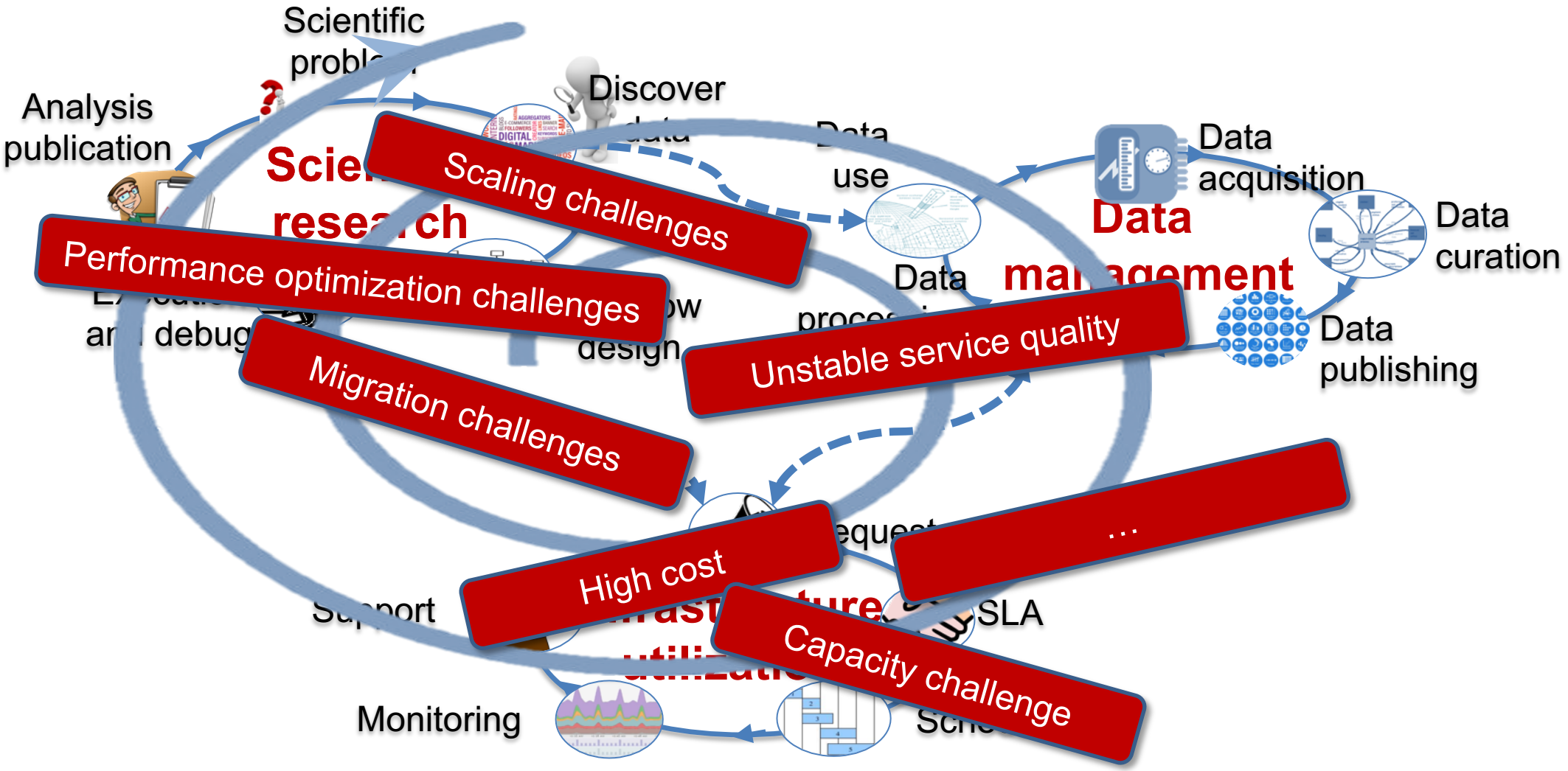
Email: z.zhao@uva.nl

ENVRI-FAIR has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824068

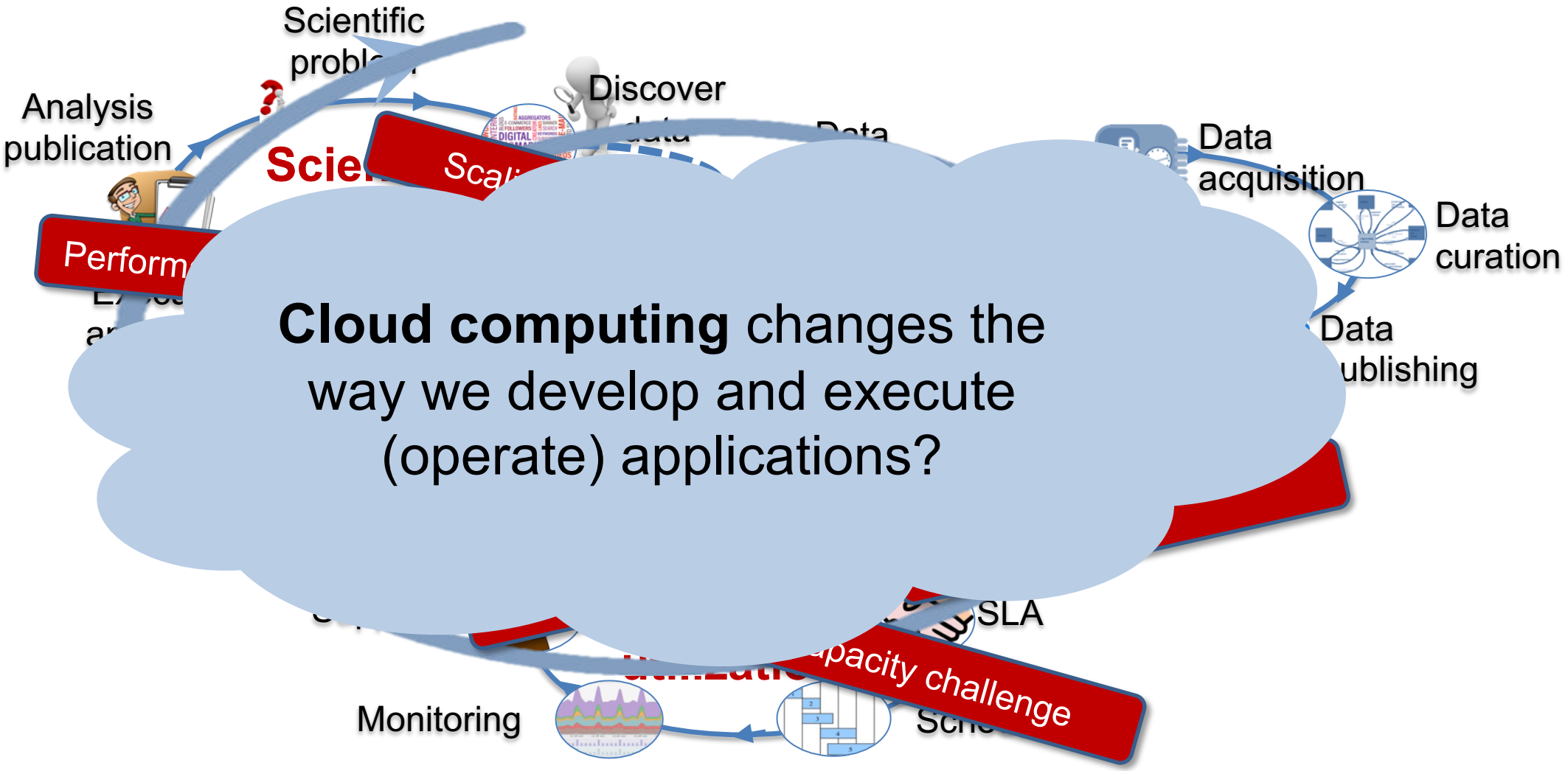
Research activities



Research activities



Research activities



Outline

1. Cloud concepts recap?
2. How to run scientific applications in cloud?
3. How to develop and operate scientific software in cloud?
4. Practical skills via lab assignments
5. Discussion

Outline

1. **Cloud concepts recap**
2. How to run scientific applications in cloud?
3. How to develop and operate scientific software in cloud?
4. Practical skills via lab assignments
5. Discussion

Cloud: resources as services

- **Infrastructure as a service (IaaS)**
  - Computing power
  - Storage
  - Network
- **Platform as a service (PaaS)**
  - Data base
  - Server (e.g., Web server)
- **Software as a service (SaaS)**
  - Word/Excel/PowerPoint etc.
- …

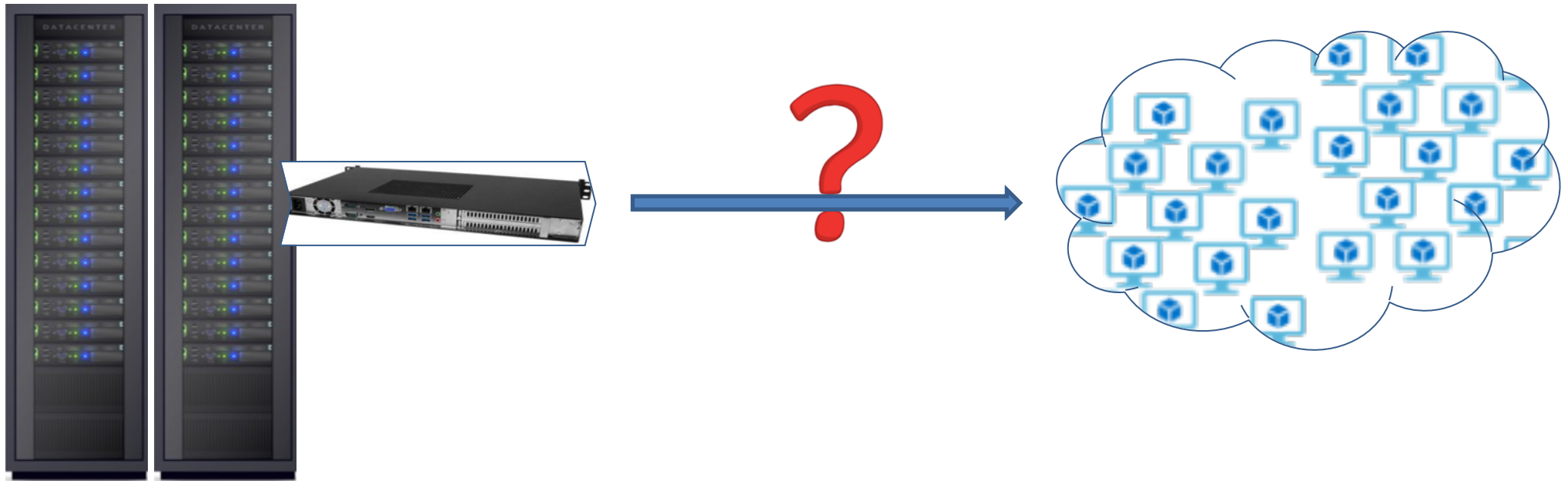
Software as a service (SaaS)
Applications, Office, Terminal emulator, etc.

Platform as a service (PaaS)
Data base, web servers, development tools etc.

Infrastructure as a service (IaaS)
Computing elements, storage, network and server etc.

Key underlying technologies

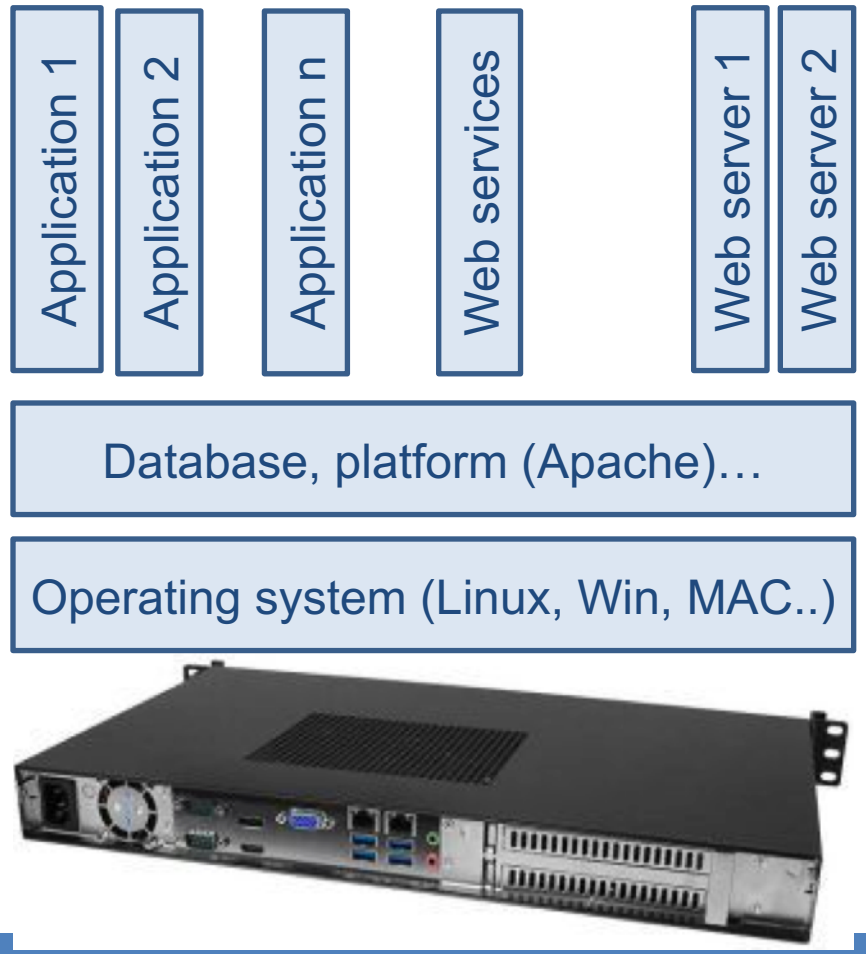
- How can I make a big physical machine as many different virtual machines?
- How do I handle the requests from different users?



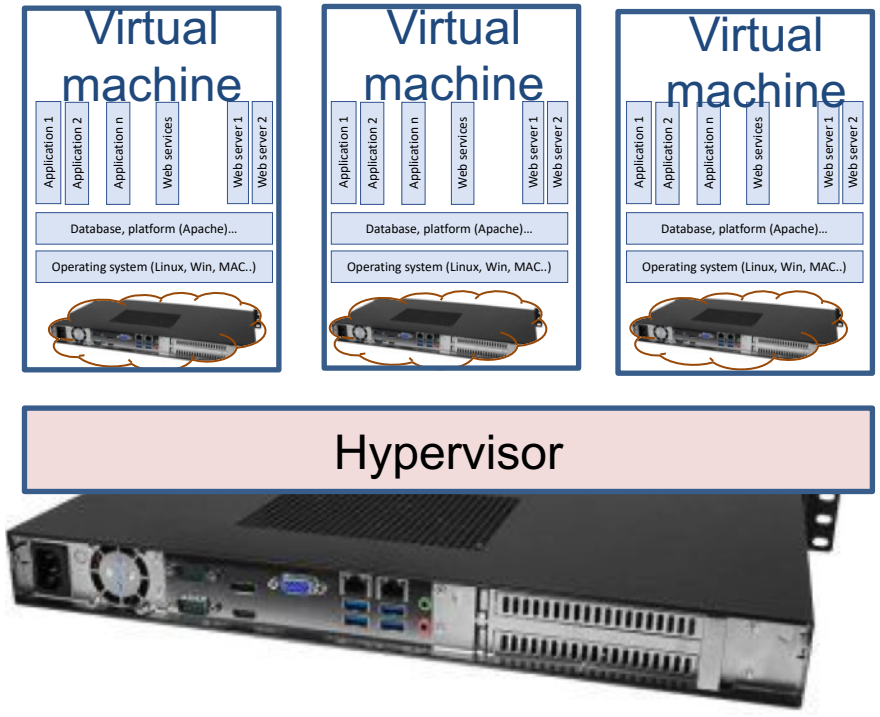
Technology 1: virtualization (Virtual Machine and Hypervisor)

A virtual machine (VM):

- A software can emulate the behavior of a real computer
- Contains hardware abstraction, OS kernel, library, file systems and etc.. The file representation is called VM **image.**



Virtualization



Technology 1: virtualization (Virtual Machine and Hypervisor)

A virtual machine (VM):

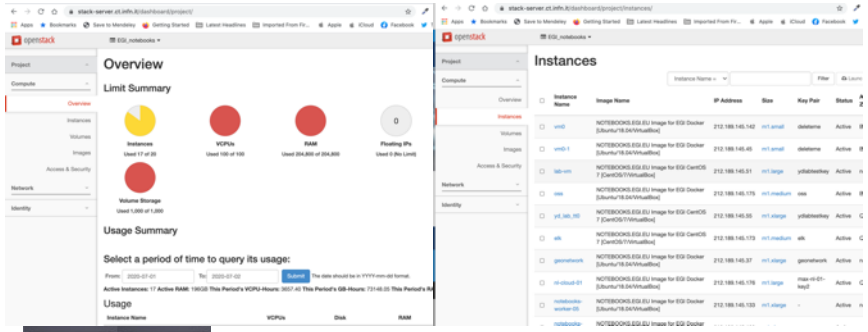
- A software can emulate the behavior of a real computer
- Contains hardware abstraction, OS kernel, library, file systems and etc. The file representation is called VM

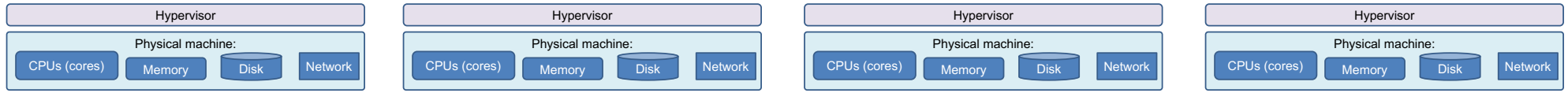
Note:

- ✓ Virtual machines are isolated (VM with different guest OS can run on the same host);
- ✓ **VM images** are files, which are usually in the size of Giga Bytes. Depends on the files included;
- ✓ **VMs** are runtime instances of the VM images in the system;
- ✓ VM images and VMs are dependent on the type of **hypervisor;**
- ✓ One physical machine usually has **only one hypervisor.**

Technology 2: Orchestration (VM provisioning automation)

- Allocate the physical resources for different VM requests
- Provide user interface and API for automating the provisioning of VM requests
- Allow administrators/users to check the current status of the resources, and manipulate them
- Example: **OpenStack, CloudStack, vRealize, Puppet, cloudformation (AWS) …**





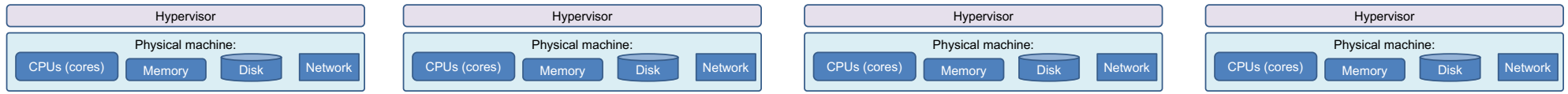
Technology 2: Orchestration (VM provisioning automation)

- Allocate the physical resources for different VM requests
- Provide user interface and API for automating the provisioning of VM requests
- Allow administrators/users to check the current status of the resources, and manipulate

Note:

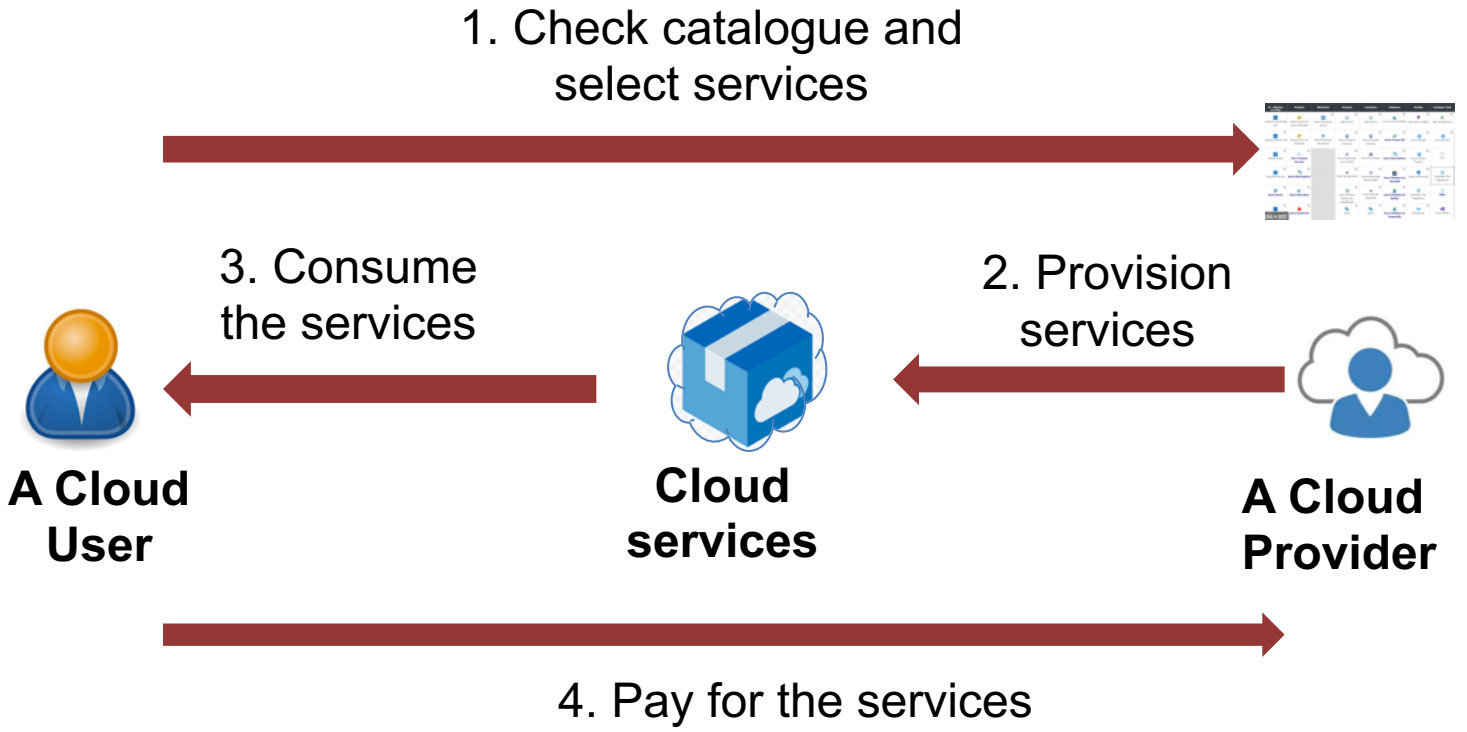
✓ An Orchestration system is interacting with the hypervisor, but independent from the hypervisor

✓ An Orchestration system provides interface for both users and administrators

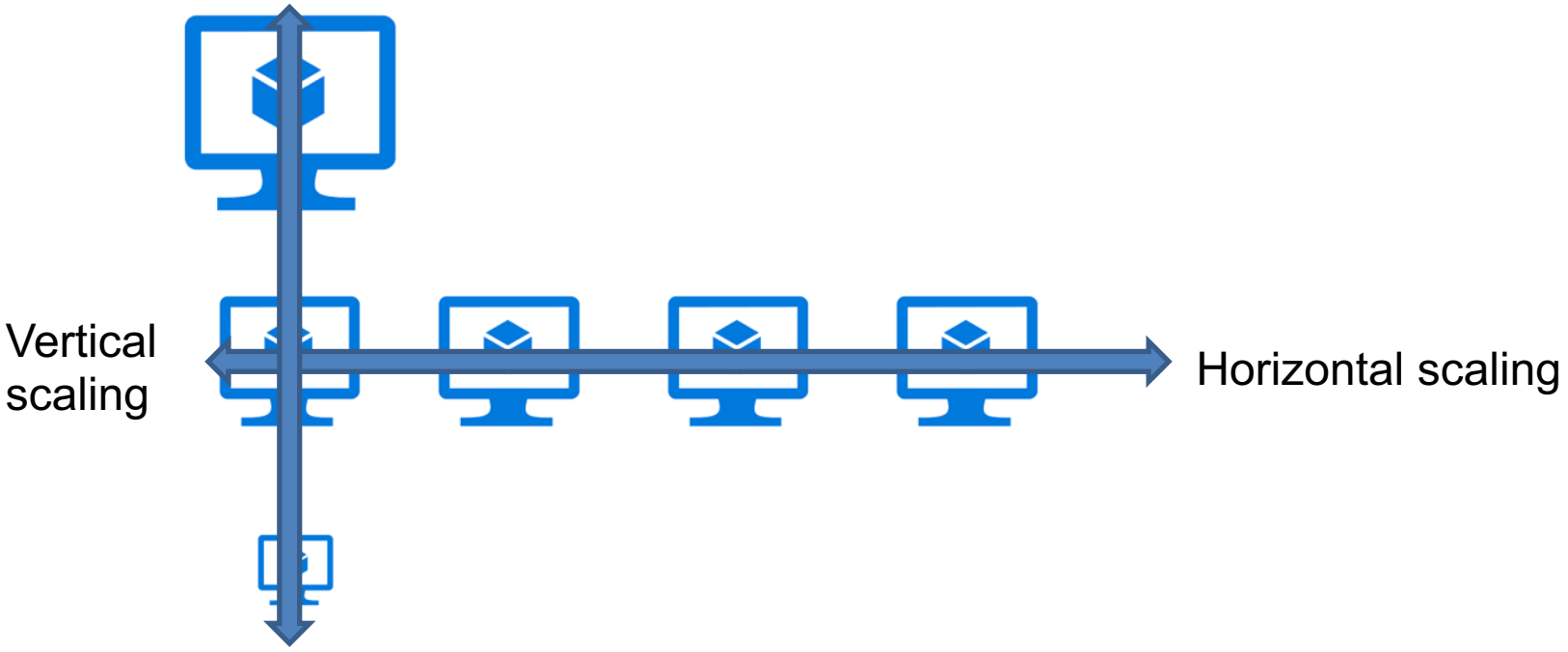


Cloud: computing resources (VMs) are online goods

- Providers sell (offer) computing capacity, storage, network and software as **goods (services)**
- **Users** can *select and customize* services, and the **provider** can *provision* them for *utilization* via **network** (internet)
- **Providers** charge the cost based on the **duration and quality** that customers use



Cloud is elastic

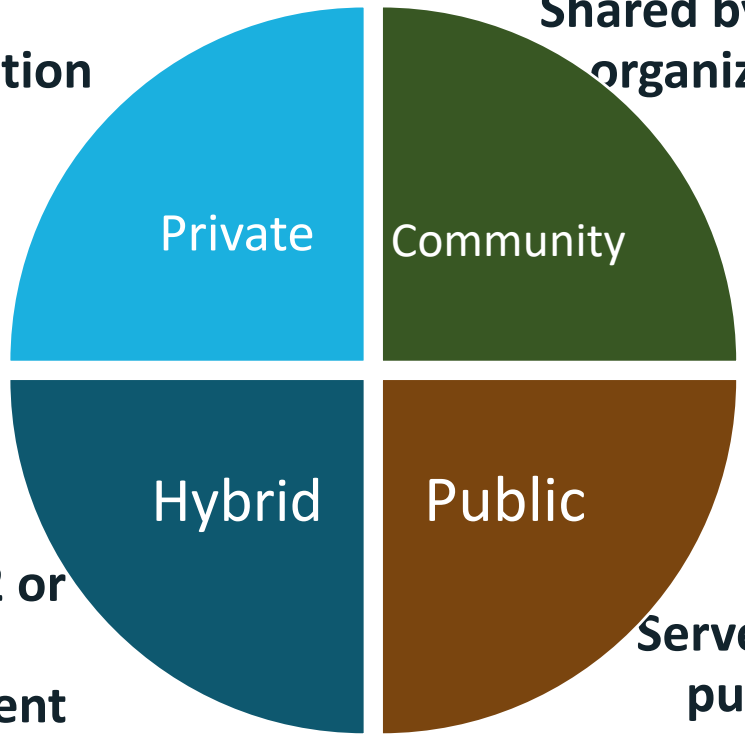


Different types of cloud

Single organization

Private used data center, e.g. for single institute

Shared by several organizations



Mix of 2 or more different types.

Serve public use

See more examples from RIs

Why secure hybrid multi-cloud?

- Highly distributed data sources
- Data security and access policies
- "**Cloud-first** with a **secure hybrid multi-cloud** service offering"[1]– EU Cloud strategy whitepaper, 2019

Discussion

- **Discussion**: What types of Cloud services have you ever used?
- Menti.com code: 3667332

Outline

1. Cloud concepts recap
2. **How to run scientific applications in cloud?**
    1. **How to select cloud services?**
    2. **How to run a scientific application in cloud?**
3. How to develop and operate scientific software in cloud?
4. Skills via lab assignments
5. Discussion

Options for runing scientific application in cloud

Typical options:

- Infrastructure as a service (e.g., VM, Storage, Network)
- Platform as a service (e.g., Database, big data cluster)
- Software as a service (e.g., Jupyter Hub)

Many new services…

- Serverless (Lambda or Function)
- Blockchain as service
- DevOps
- ….

A scientific application

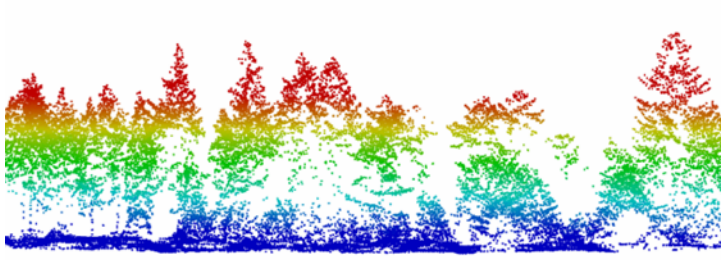
- Types of scientific application
  - Modelling and simulation,
  - big data analytics,
  - machine learning,
  - sensor data processing,
  - data base
  - Workflow of different tasks
  - …

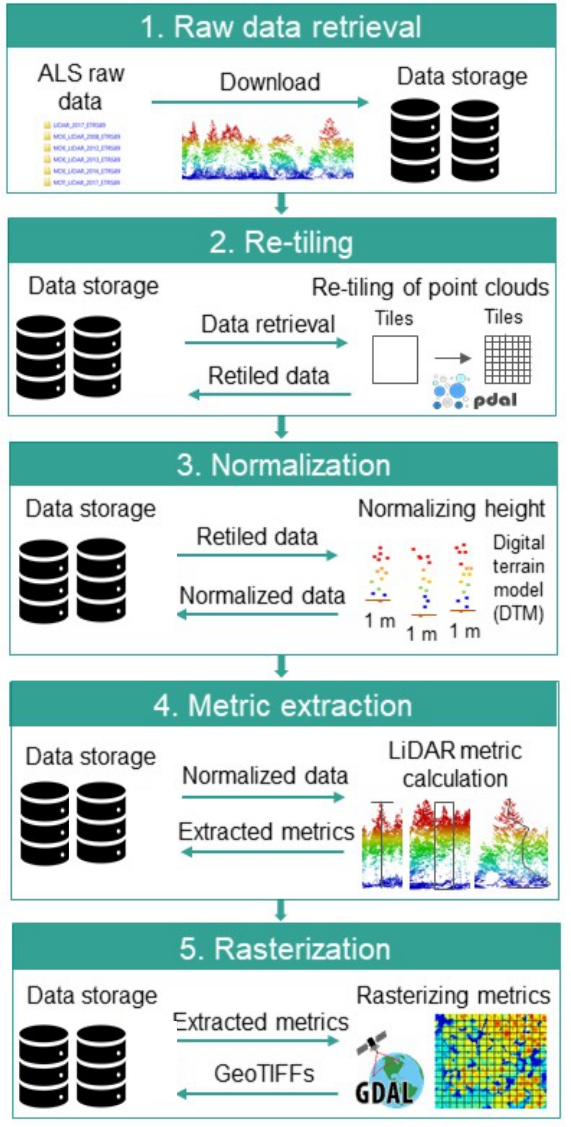
Discussion: What types of scientific application are you developing?
Menti.com code: 3667332

An example

- A use case from LifeWatch: processing high-resolution Light Detection and Ranging (LiDAR) measurements
- A researcher developed basic analysis code in python: point cloud processing for Airborne Laser Scan (ALS) raw data
  - Retiling → normalize height → Lidar metric calculation → Rasterizing
- Python code developed in Jupyter, only with data from NL
- Executed on our local infrastructure

Processing LiDAR point clouds (ALS raw data)



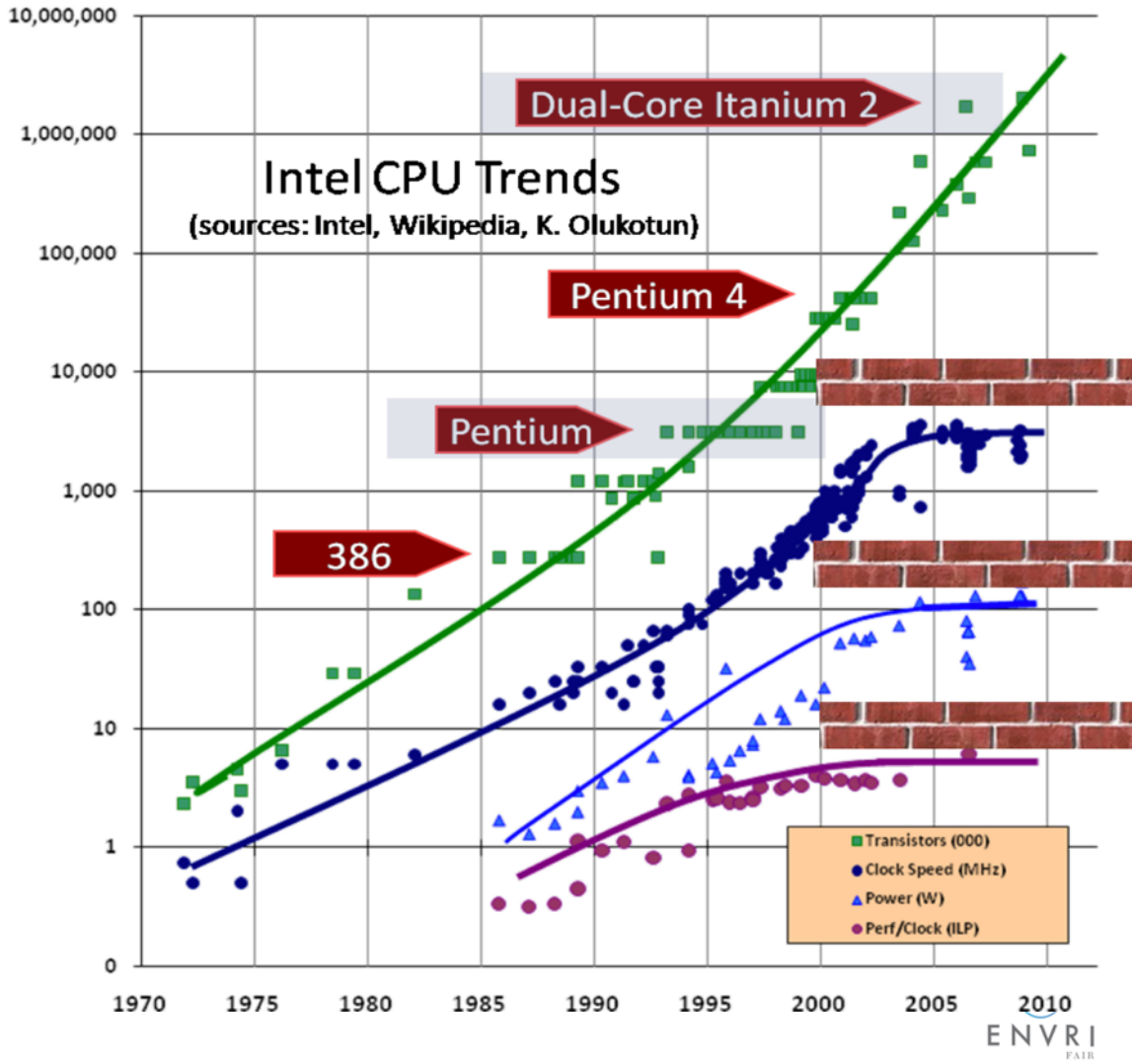


How to scale the application?

- **For larger data set**, for instance
  - Large Volume data set (e.g., Process ALS data from the entire country or EU scale)
  - and Multi data sources (e.g., species information from GBIF)
- **For more intensive computing tasks**, for instace
  - simulating high resolution models or training high quality deep neural networks (e.g., combing species distribution or climate information)
- For **combing new features**, or **models**

Why need remote computers?

- The increase of the **process speed** (clock frequency) will reach certain limits, which can not infinitely improve the speed of your program
- A developer has to think of different ways to improve the performance by parallelizing the scientific code, typically:
  - **Multi threads** (typically, in a single computer, with shared memory, e.g., multi core, many core)
  - **Message passing** (in different computers, distributed memory, e.g., cluster)

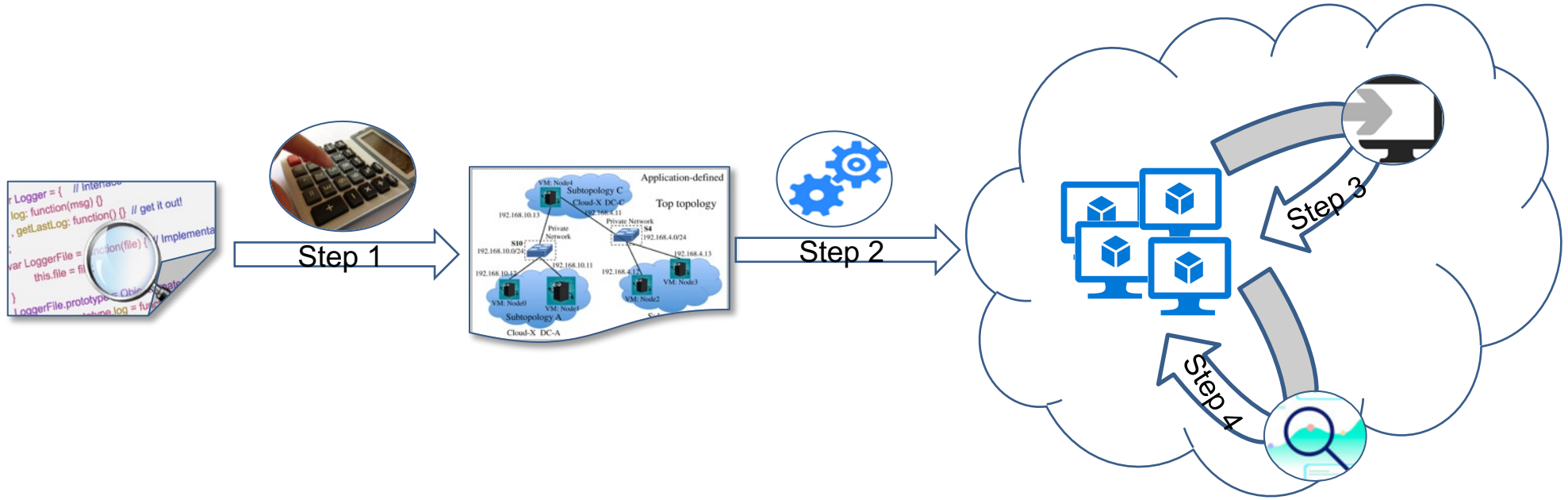


Run your application in a remote infrastructure

- Send, build and execute code to the remote infrastructure
  - E.g., running parallel code in a remote computing cluster or super computer.
- Advantages
  - Special hardware
  - Known architecture and capacity in advance
- Disadvantages
  - Shared environment
  - Can have long waiting list

Using Infrastructure services

1. Virtual infrastructure planning
2. Virtual infrastructure provisioning
3. Platform and software deployment
4. Application execution and monitoring



Analyze your application first

- Application characterstics
  - What is the program model of the application?
    - Standalone program:
      - Sequential code
      - Parallel code: Multi threads, MPI, or CUDA/GPU
    - Workflow of tasks
  - Communication characterstics: driven by data (files) or events (small messages)
  - Computing tasks with short time duration or persistant online applications (e.g., web pages)

Related to the **CPU** capacity you need

Related to the **IO/Network** capacity you need

Related to **duration** you need

Planning the cloud service (e.g., type, capacity, topology, price) based on expect performance requirements, data policies, application characterstics and budget.

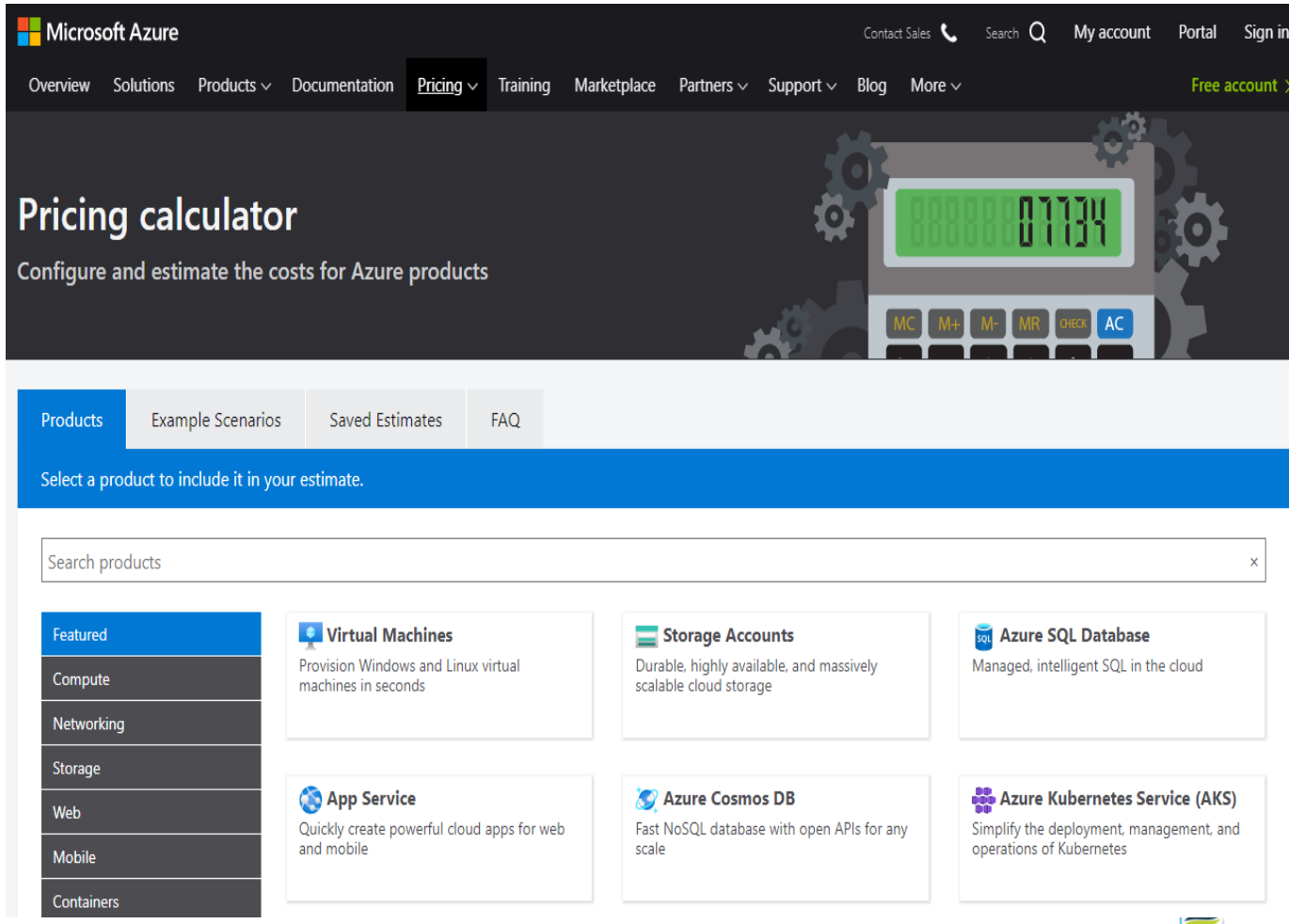
A close look at the scientific program

- Computer architecture and programming model
- Characterstics of using computing resources
  - (*They depend on the specific application*)

	Core(s) (CPU or GPU)	Memory	I/O	Network
Modelling and simulation	+++ / distributed	++ / distributed	+	+++
Big data analytics	++ / distributed	+++	+++	+++
Deep neural network	+++	+++	+++	+
Sensor data processing	+	+	+	+++
Data base	++	+++	+++	+
Workflow	distributed	distributed	distributed	+++
…				

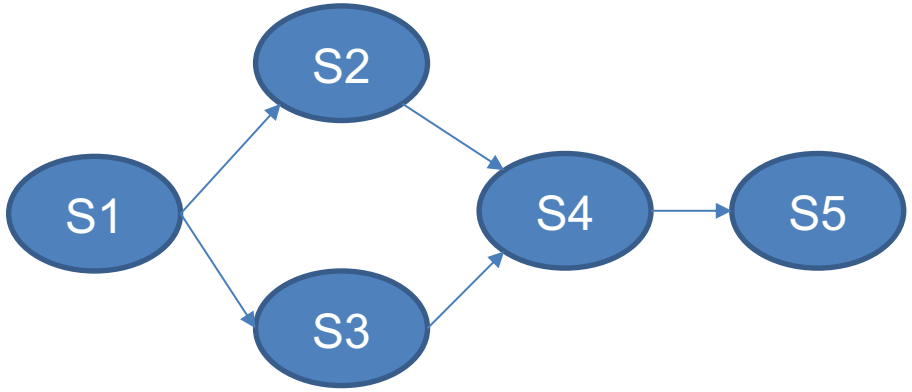
Azure pricing calculator: an example in public cloud

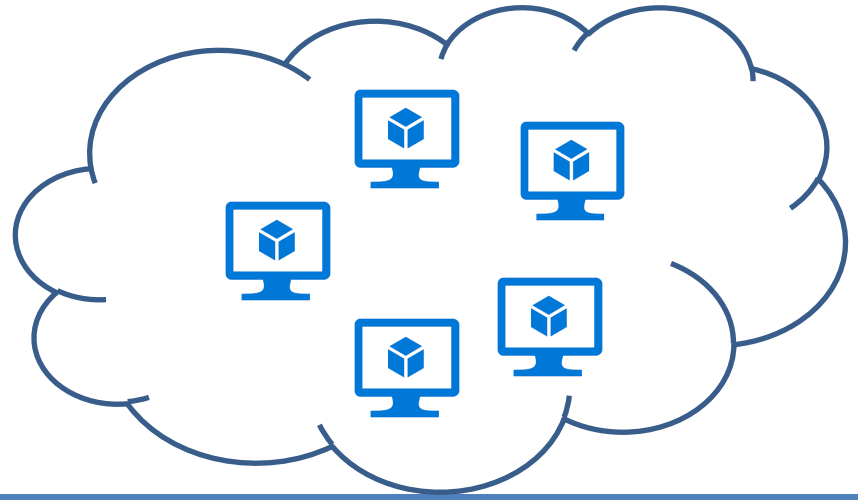
- Virtual machine types
  - Operating system, General purpose or special ones (normally in special data centers)
- Virtual machine capacity
  - Number of core (cpu),
  - Size of memory,
  - Size of disk,
  - Network bandwidth,
  - Input/Output throughput
- Virtual machine price
  - Price/hour



For workflow of services

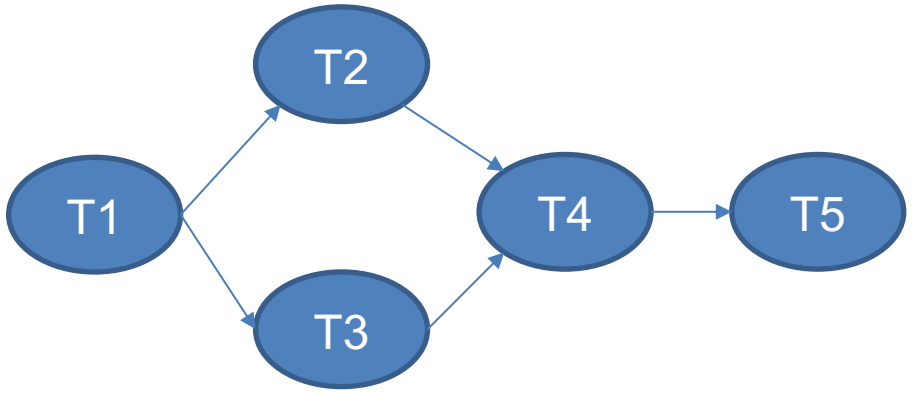
- If the tasks are web services (need to be persistantly online)
  - Select virtual machines for individual web service, or for group of services based on their performance characterstics (following similar approach for a single application)
  - Or set up a container cluster based on a set of virtual machines, and deploy services as containers on the cluster
  - Note: if you want to enable auto-scaling, extra capacity needs to reserved

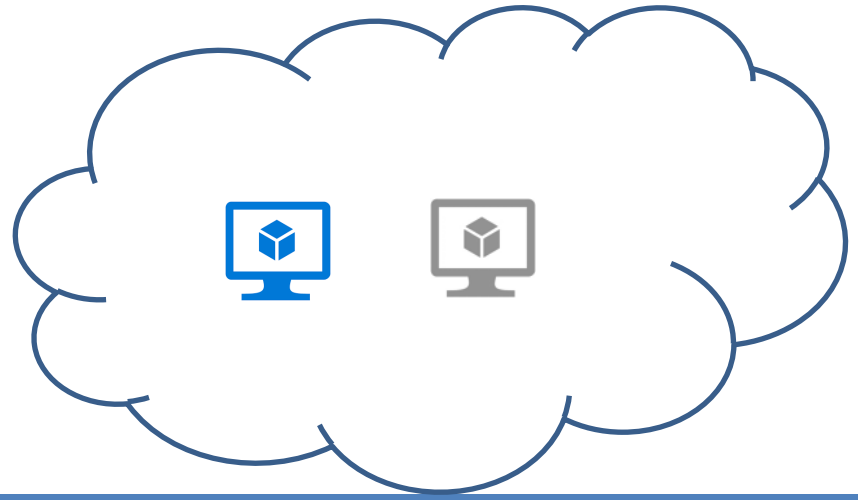




For workflow of computing tasks

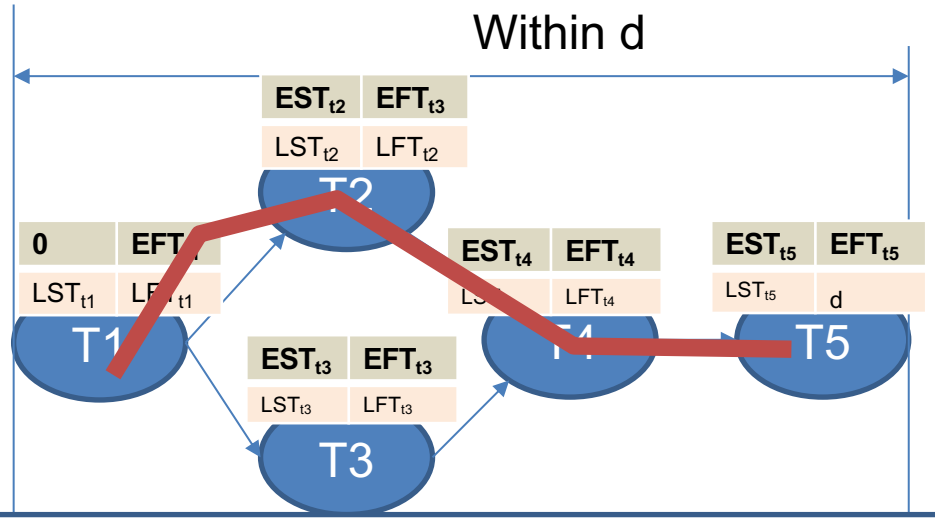
- If the tasks have short duration, but dependency on input/ouput
  - **On demanding way**: reserve some VMs in the pool, and select certain size of VMs from the pool based on workflow scheduling; when VM is running out, you just add other instances in the pool.
    - For T1-T5, a new task starts after the previous one finishes
    - It means a new task can reuse the VM of the previous one. In this example, you may only need a single VM or just two.

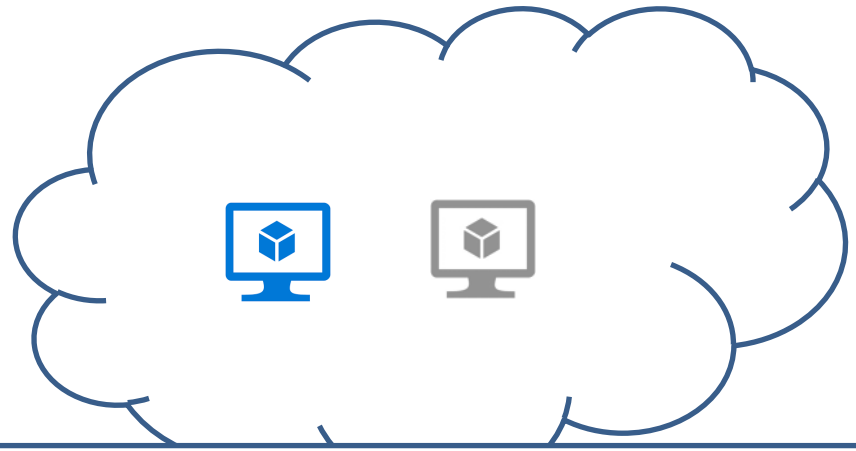




For workflow with critical time constraints

- If the tasks have short duration, but dependency on input/ouput
  - **Quality critical applications**: if you need the workflow (part or entire) finish within reuired time (also called deadline).
    - One way is to use critical path-based approach; by estimating the time cost for each task (earliest/latest starting/finishing time: EST, EFT, LST, LFT. The EST of t1 is 0, LFT of t5 is d)
    - Select a VM for all tasks in the critical path (all tasks EST=LST and EFT=LFT)
    - The execution time of a task is estimated based on measurement or performance model





Wang, J., Taal, A., Martin, P., Hu, Y., Zhou, H., Pang, J., de Laat, C., **Zhao, Z.**: Planning virtual infrastructures for time critical applications with multiple deadline constraints. Future Generation Computer Systems. 75, 365–375 (2017). https://doi.org/10.1016/j.future.2017.02.001, [https://zenodo.org/record/3361591].

Provisioning: Infrastructure as a code

- Manually select virtual machines from the user portal offered by the cloud provider
- Or describe the virtual machines and their topologies using a description file
  - Using TOSCA, CAML… (https://doi.org/10.1145/3150227)
- Providers normally offer API to execute the description
  - During ENVRIplus, we jointly developed a tool called Dynamic Real-time Infrastructure Planner (DRIP) together with the EU SWITCH project.

Deployment

- The VMs got from Cloud providers are often for general purpose; you can customize the OS version, and hardware configurations
- The software platforms needed by your scientific application, e.g., python, java, etc., must be installed by yourself
- Do it manfully
  - Remotely login the system (as you will try during the lab)
  - Install them using relevant commands on Linux or other systems
- Or automated
  - Compose the installation orders as a playbook, and automate it using the tool like ansible

Deploy a distributed application in a remote environment is time consuming!

Deploying the different containers

- VM images are complete self-contain, but are usually very large;
- Application virtualization are not generic for all languages
- Container technologies are getting popular in cloud computing.

Virtual machine images

- ✓ Application
- ✓ Platform and libraries
- ✓ File systems
- ✓ Operating systems (full)
- ✓ Hardware configuration and abstraction

- ✓ **Directly deploy above hypervisor**
- ✓ *No other installation needs*

Container (Docker) images

- ✓ Application
- ✓ Platform and libraries
- ✓ File systems
- ✓ Operating systems libraries (without kernel)

- ✓ **Require a container (e.g., docker) engine**
- ✓ **Require operating system kernel from the host environment**
- ✓ *(Virtual)* ***Hardware*** *configuration*

Application packages (JAR, WAR)

- ✓ Application
- ✓ Platform and libraries

- ✓ **Require special environment (e.g., java runtime environment)**
- ✓ *Require* ***full*** *operating system kernel from the host environment*
- ✓ *(Virtual)* ***Hardware*** *configuration*

ENVRI FAIR

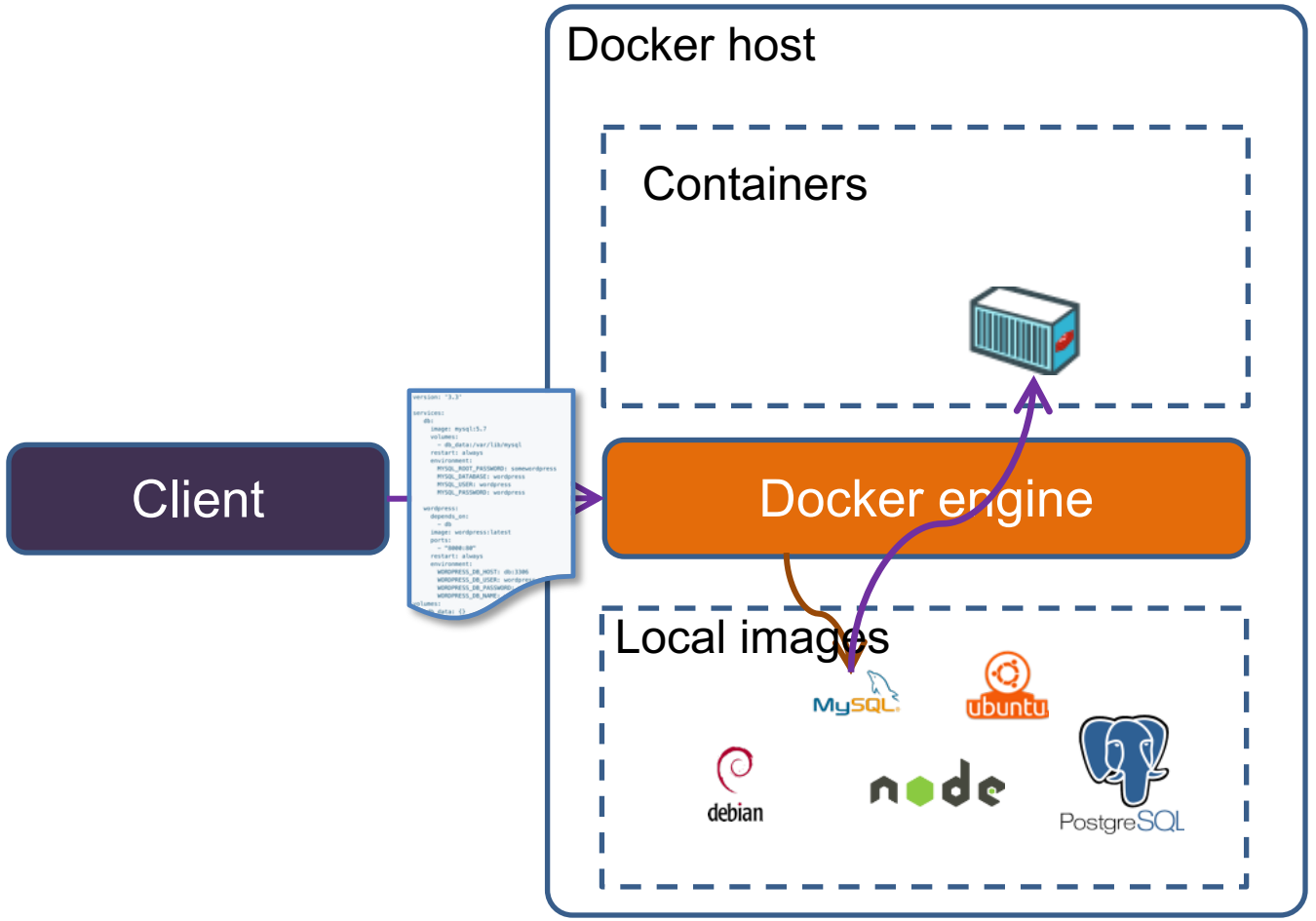
How does a do

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD:
       WORDPRESS_DB_NAME:
volumes:
    db_data: {}
```

Client

Docker Hub

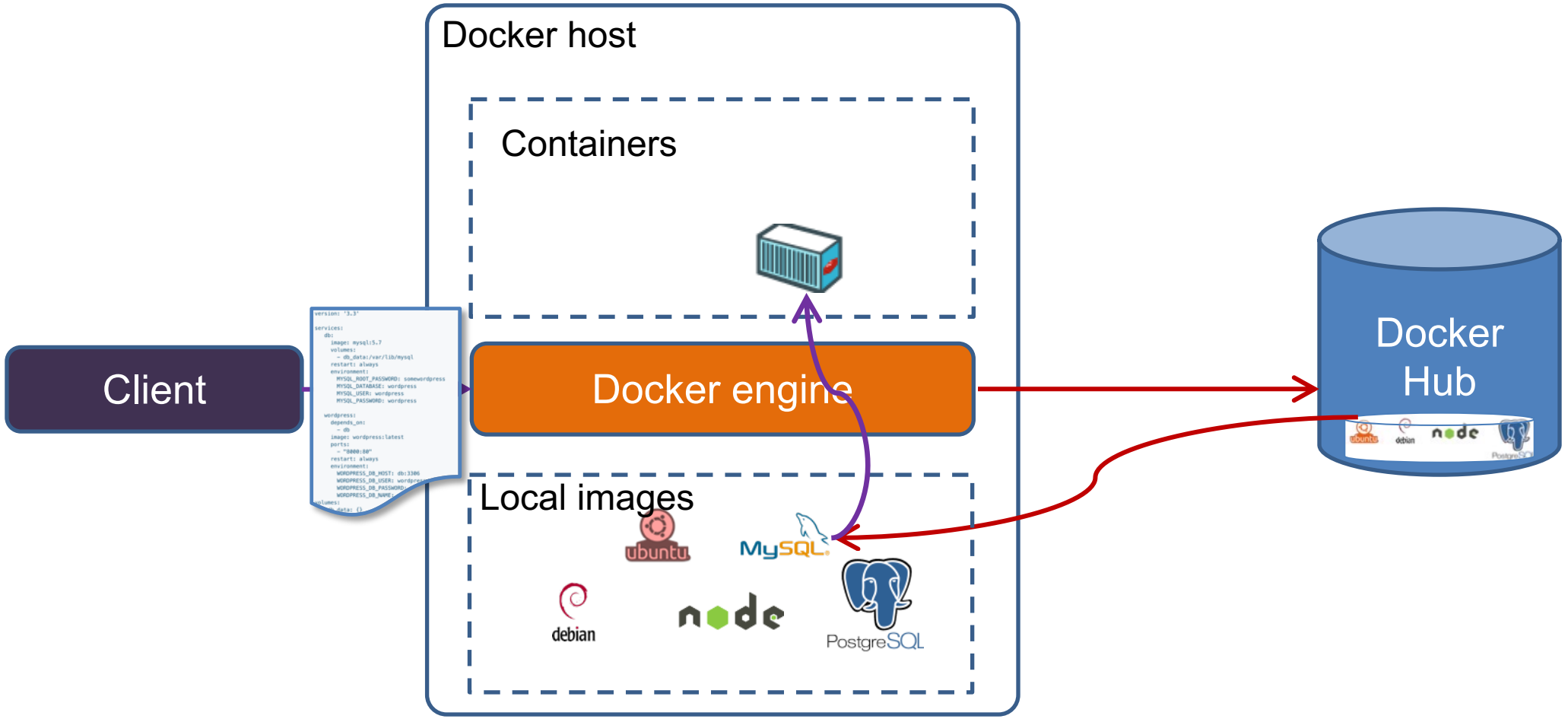
If the docker image exists locally



How does a do

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD:
       WORDPRESS_DB_NAME:
volumes:
    db_data: {}
```

If the docker image does not exist locally



Docker orchestration over a cluster

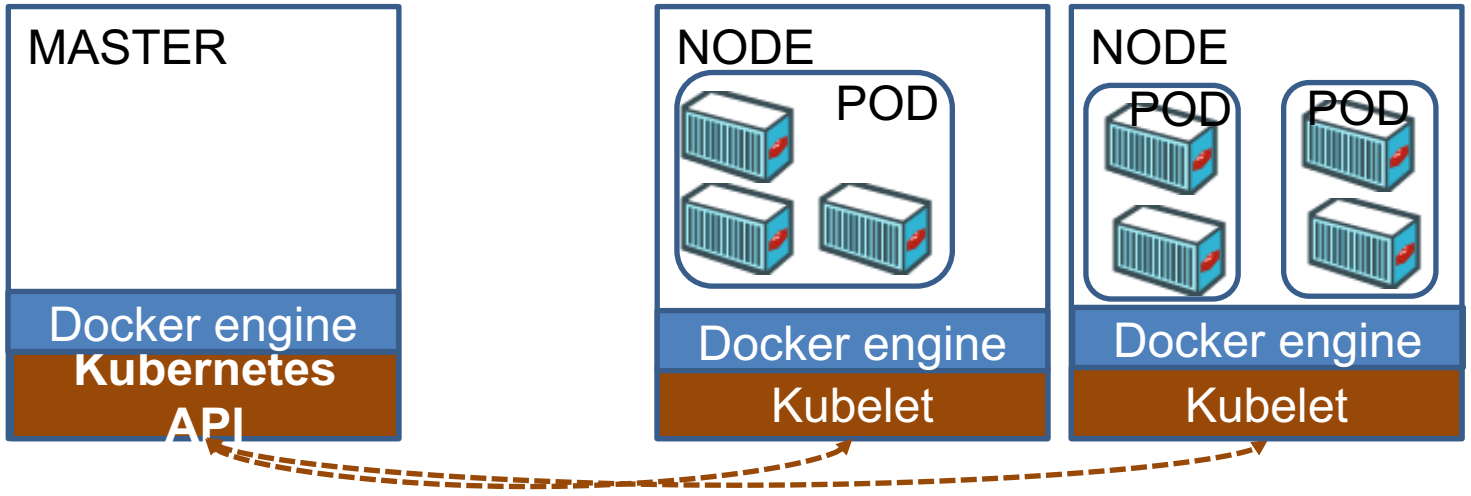
- SWARM: using compose file
- MESOS: using marathon
- Kubernets: google

kubernetes

MESOS

Kubernetes cluster and POD

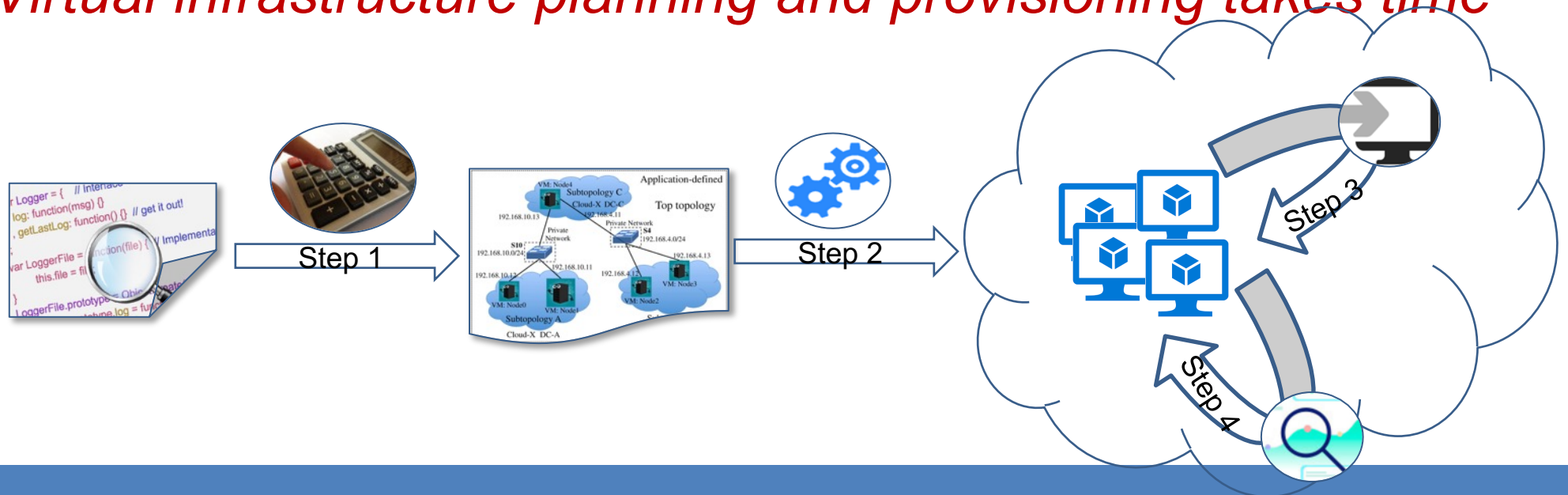
- Smallest unit in Kubernetes, A structural abstraction of a group Containers
  - Some Containers are dependent, and need to be deployed in a single host, or work together. Share IP address or port space.
  - Can also be on container per POD
- Containers in a POD share storage/network



https://kubernetes.io/docs/setup/pick-right-solution/

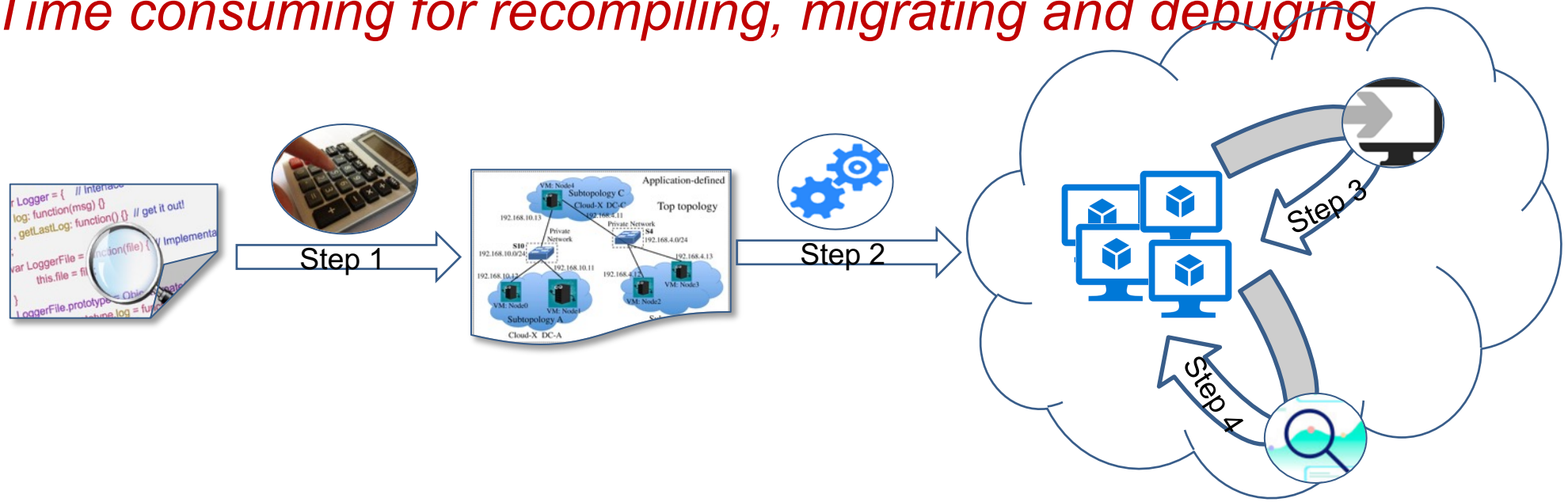
Run scientific code in cloud- Option 1

1. Virtual infrastructure planning
2. Virtual infrastructure provisioning
3. Containerize (Dockerize) the scientific code
4. Deploy the containers on a docker cluster (Kubernetes cluster)
5. *Virtual infrastructure planning and provisioning takes time*



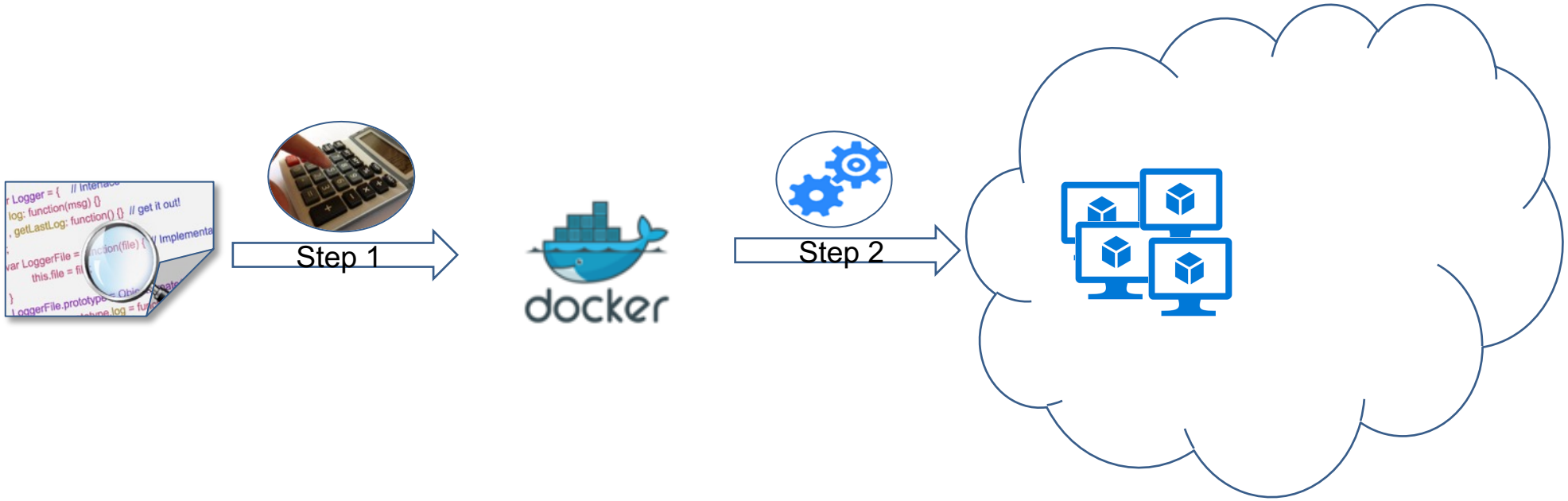
Run scientific code in cloud- Option 2

1. Virtual infrastructure planning
2. Virtual infrastructure provisioning
3. *Deploy and compile the scientific code on the virtual infrastructure*
4. *Execute the code*
5. *Time consuming for recompiling, migrating and debuging*



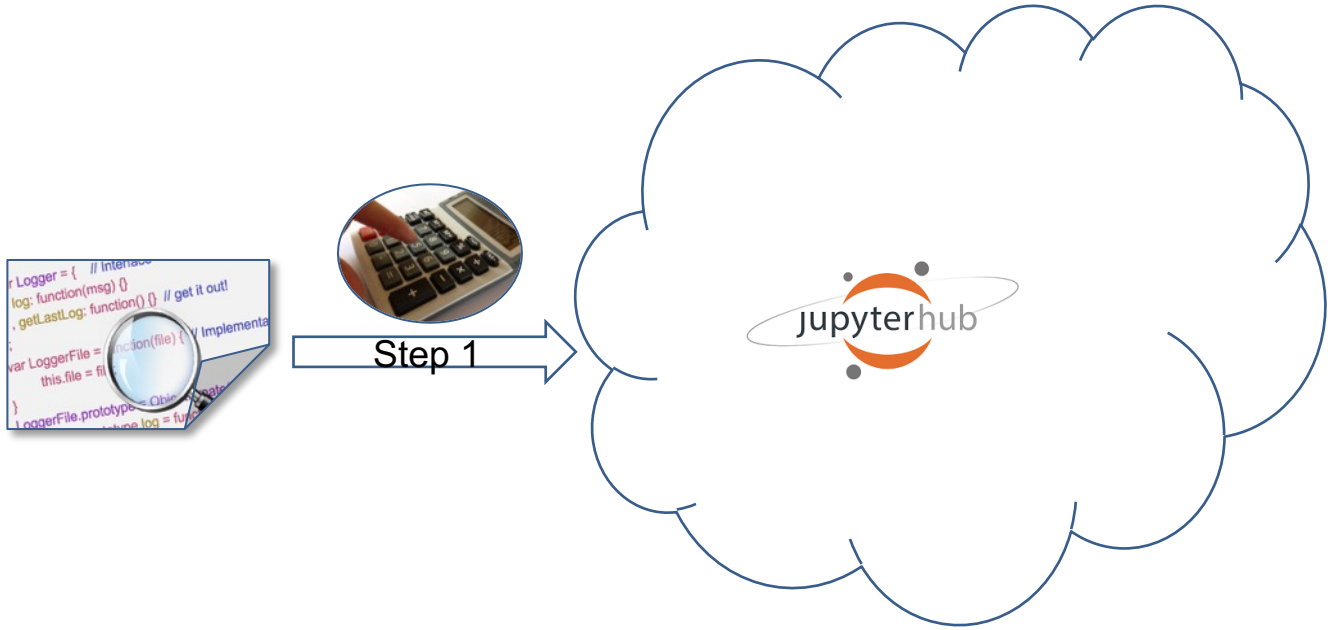
Run scientific code in cloud- Option 3

1. Containerize (Dockerize) the scientific code
2. Deploy the containers on a docker cluster (e.g., Kubernetes cluster) from the provider.
3. *Not all providers provide this service.*



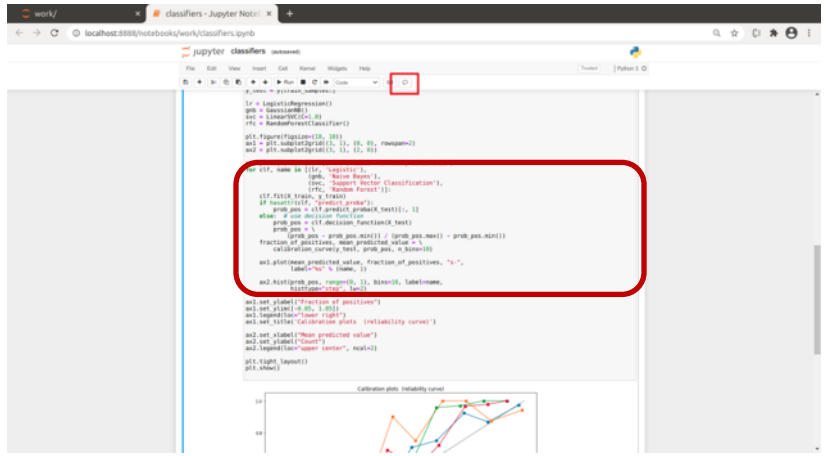
Run scientific code in cloud- Option 4

1. Directly use the jupyter environment (e.g., Jupyter Hub) from the provider.
2. *You can only share the notebook, might be difficult to be included other workflows*



From scientific code to docker

- Scientific code is often developed in Jupyter
- Can we containerize the code, or a specific segment (Cell) as a RESTful service and encapsulate it as a container?

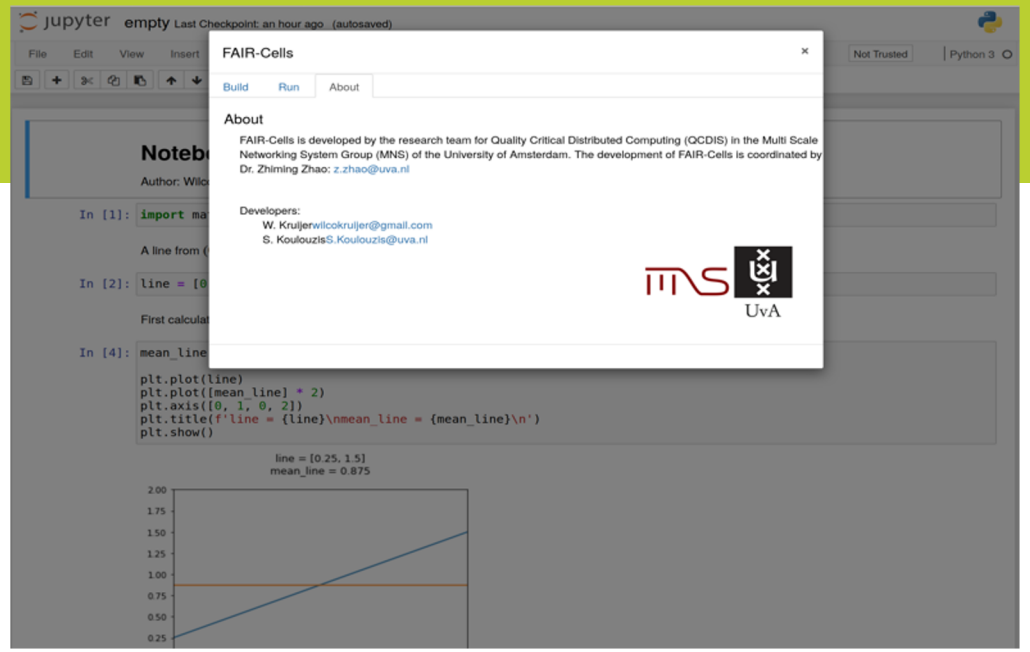


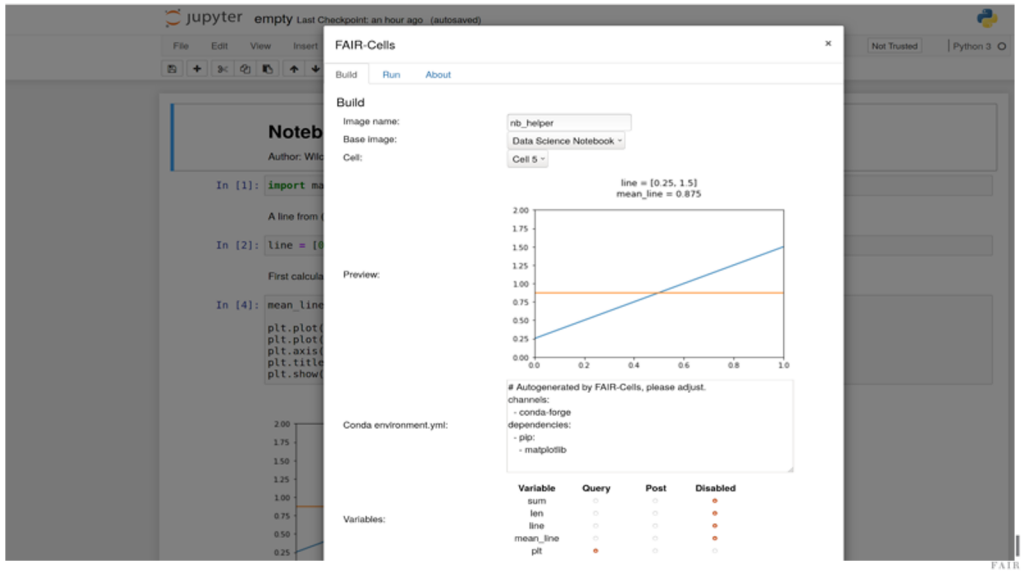
FAIR-CELL: a Jupyter extension

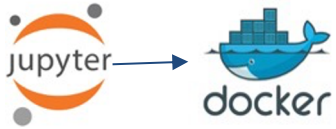
- Install on a jupyter environment as an extension
- It allows you to interactively containerize any code from your notebook
- It is developed by the LifeWatch Virtual Lab Innovation Center (VLIC) in the context of ENVRI-FAIR project and the EOSC early adopt program (EOSC EAP)
- Ope source: https://pypi.org/project/FAIR-Cells/
- Live demo: http://shorturl.at/koDH0

FAIR-Cells

- FAIR-Cells is a Jupyter Notebook extension that allows users to interactively encapsulate a Jupyter cell as RESTful service wrapped as a Dockerfile
- The Dockerfile can be then build as a container and deployed and scaled using a container container orchestrator e.g. kubernetes
- Currently, FAIR-Cells supports python but we want to include more kernels such as R

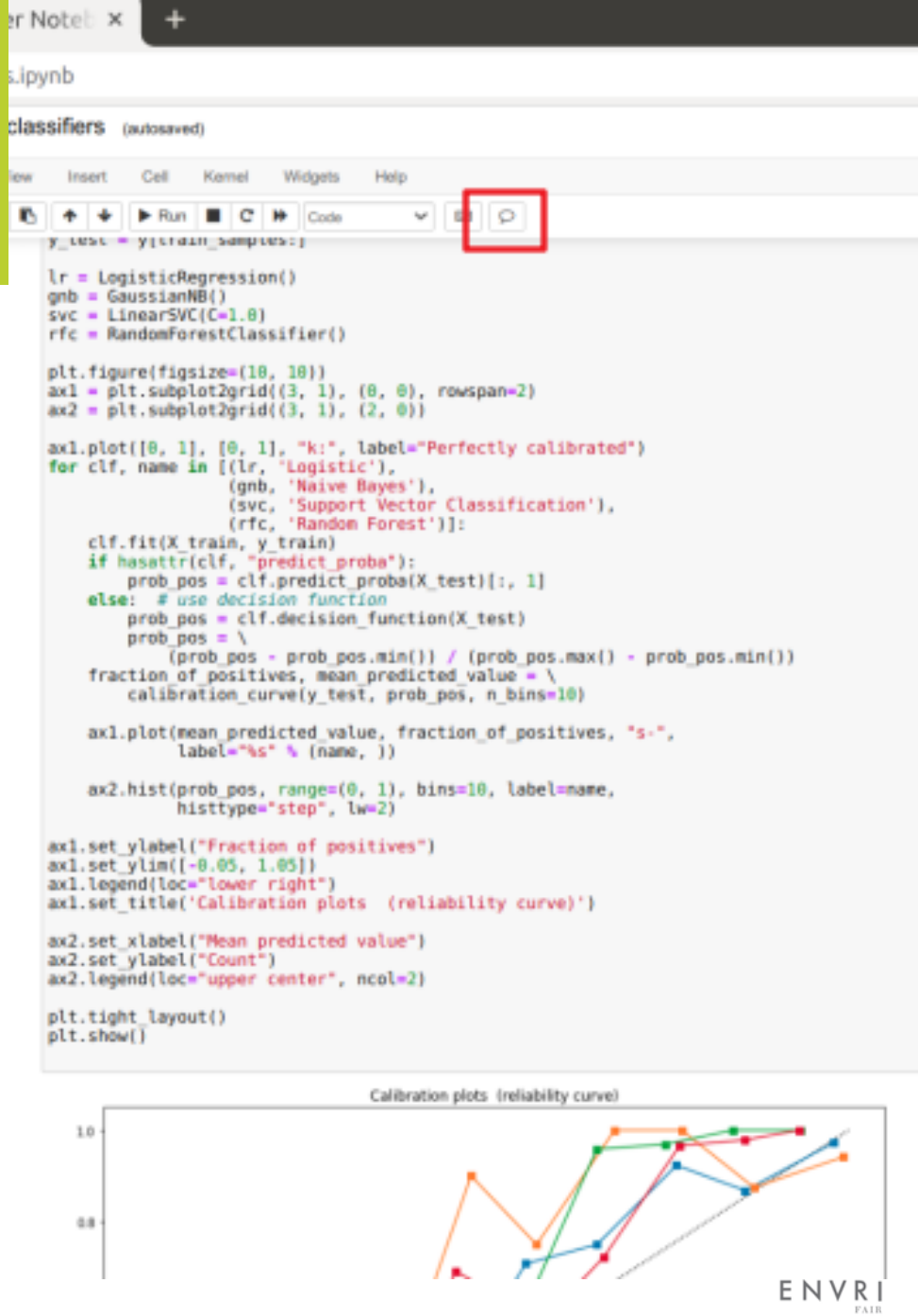






Lab practice

1. **Tutorial 1: http://shorturl.at/bjMUV**
   1. Jupyter notebook + scientific code
   2. How does the containerization work?
   3. Create a test docker of the code
   4. Store the test docker to the dockhub
2. **Tutorial 2: http://shorturl.at/dnBF9**
   1. Install kubernetes
   2. Deploy and execute the test docker in the kubernetes (K8s) environment

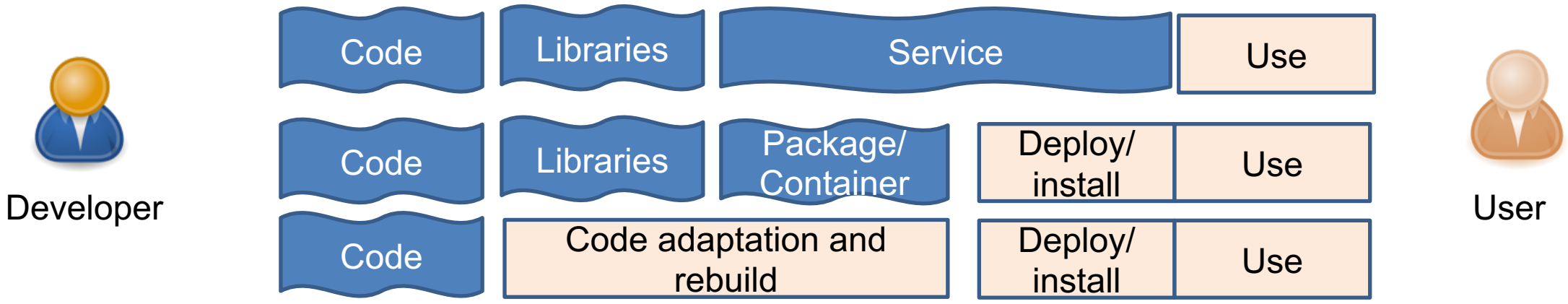


Outline

1. Cloud concepts recap
2. How to run scientific applications in cloud?
3. **How to develop and operate scientific software in cloud?**
    1. **Why do we need agile and continuously integration/deployment (CI/CD)?**
    2. **What is DevOps?**
    3. **How to set up an open source framework for CI/CD?**
4. Skills via lab assignments
5. Discussion

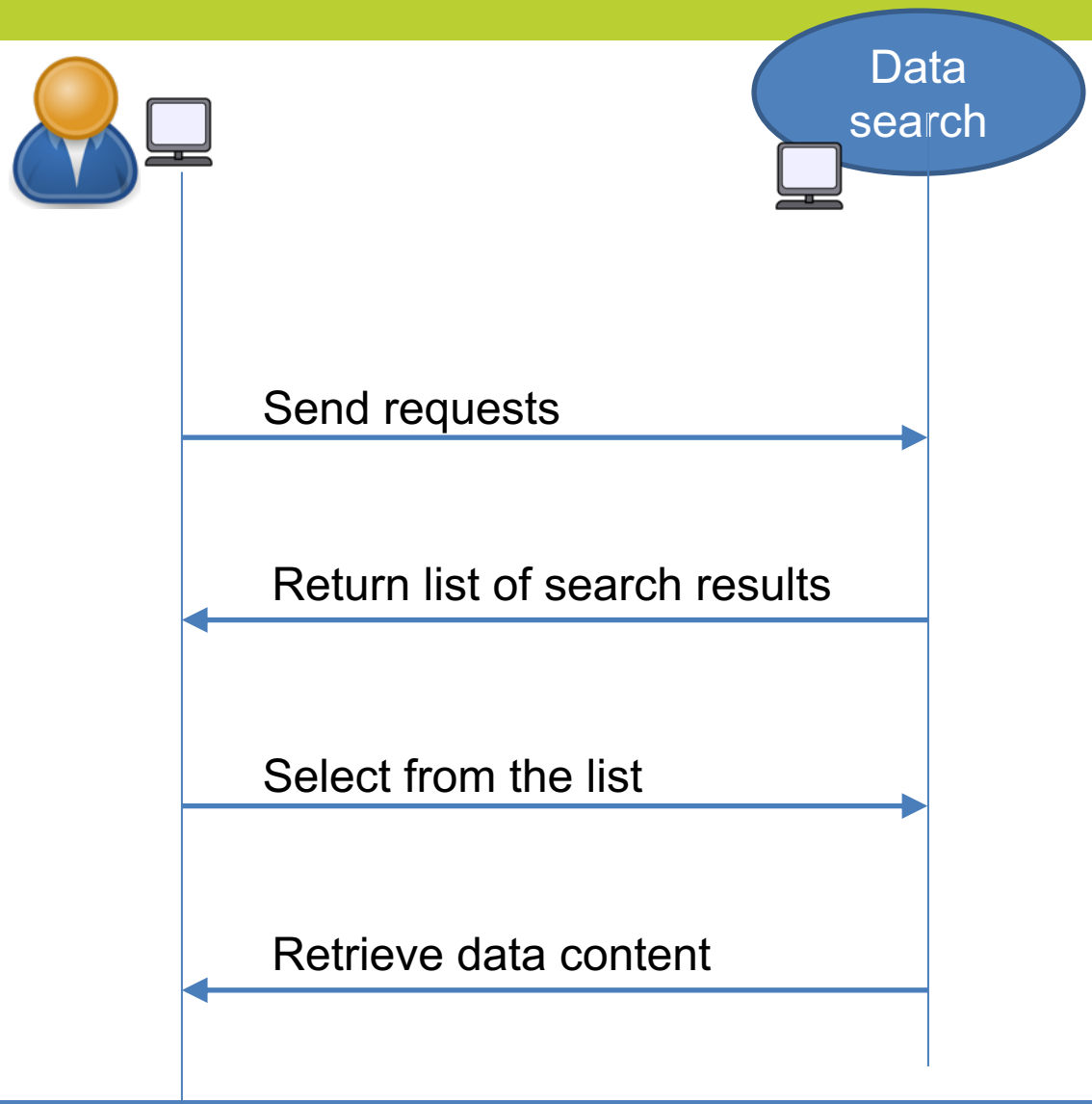
Software value delivery: from code to service

- Scientific value delivered at:
  - Code level
  - Container level
  - Service level



Example web services

- Software function can be invoked via network
- Standardised protocol
  - HTTP, or others
  - Encoding/decoding of data, e.g., XML, JSON

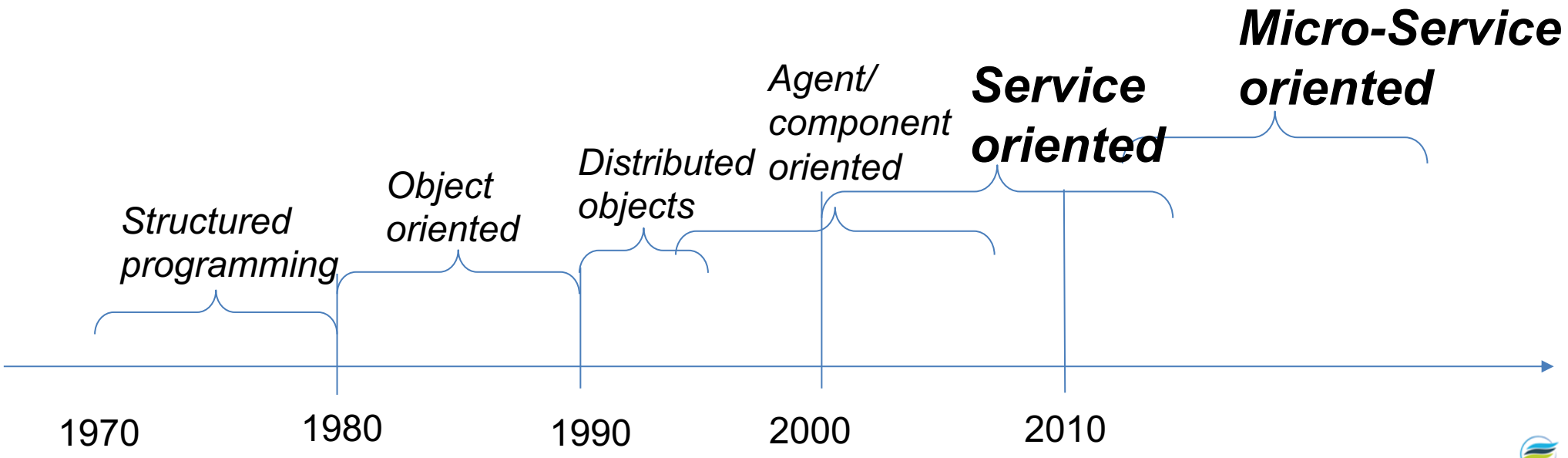


Services in data management and cloud

- Used in many different contexts
  - An organization offers long term **data archive services**
  - Run an application on **cloud infrastructure services**
  - A research infrastructure offers **support services** for solving technical problems from users
  - A scientific workflow invokes online **a web service** for processing satellite images
  - …

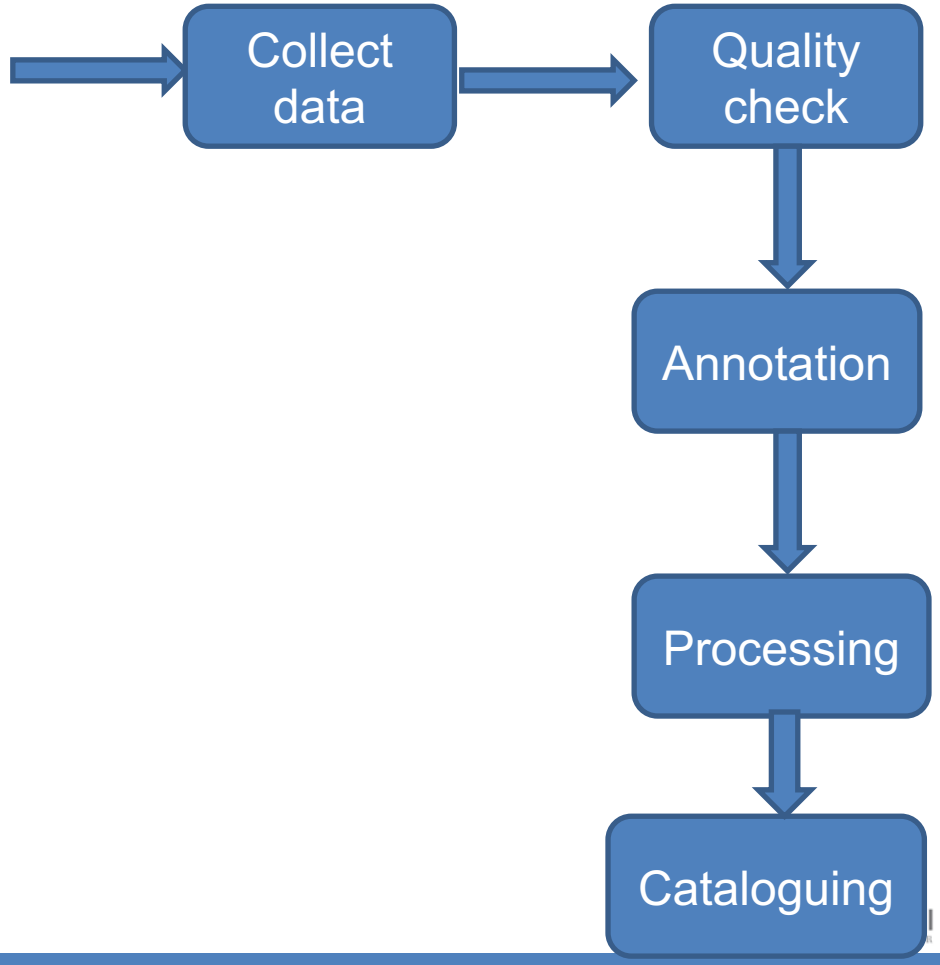
From the evolution of programming paradigms

- A **programming paradigm** is a **fundamental style** of computer programming that deals with how solutions to problems should be **formulated** in a **programming language**.



From the business process automation

- A **business process** is a collection of linked tasks for achieving specific business objectives.
- Example: the lifecycle of data management: from raw data to published data products



Business process automation

- **Automated business processes can**
  - Be more **flexible** in refining the business processes and their logic
  - Improve **efficiency** by reducing overhead caused by manual operations
  - Be more **competitive** in delivering the business value, e.g., integration optimized processes from collaborating companies
- **Services orientation** as a design paradigm
  - Decomposes business (application) logic into **suitable units** that can be designed to solve immediate **problems** while remaining **agnostic** to the **possible future greater problems**.

From the business value

- **Services** (also known as "**intangible goods**") include attention, advice, access, experience, and affective labor. [*from wikipedia*]
- E.g., Cloud services, GPS services, ..
- ....

Companies:

"a great business should focus on **serving**, not selling"

Business and ICT aspects of services

Agile, incremental, iterative

Software implementation of services: e.g., Service Oriented Architecture (SOA), web services, micro services, business process modelling (BPM), etc.

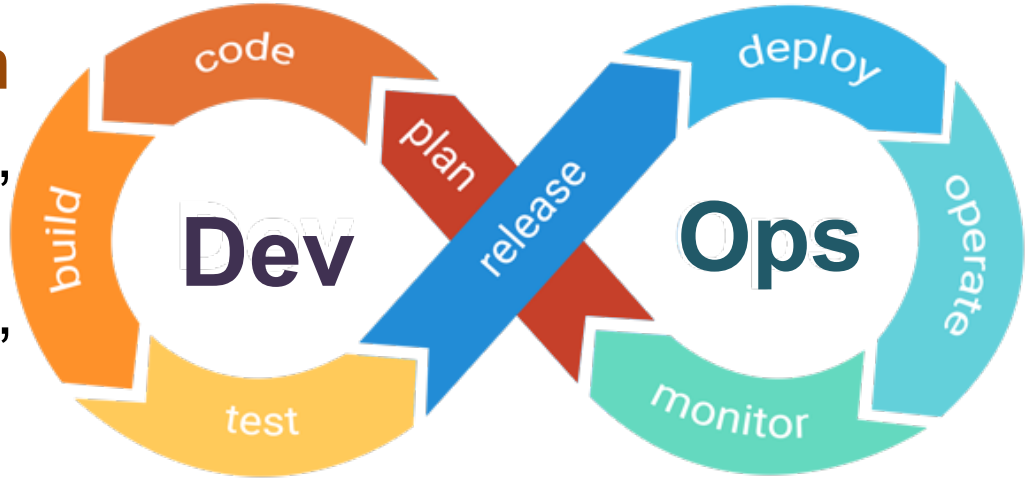
Continuous (integration, testing, deployment and operation)

Business value of services requires high quality continuous operation: e.g., response time of data search portal, availability of services when number of user increases …,

Business and ICT aspects of services

Software implementation of services: e.g., Service Oriented Architecture (SOA), web services, micro services, business process modelling (BPM), etc.

Agile, incremental, iterative

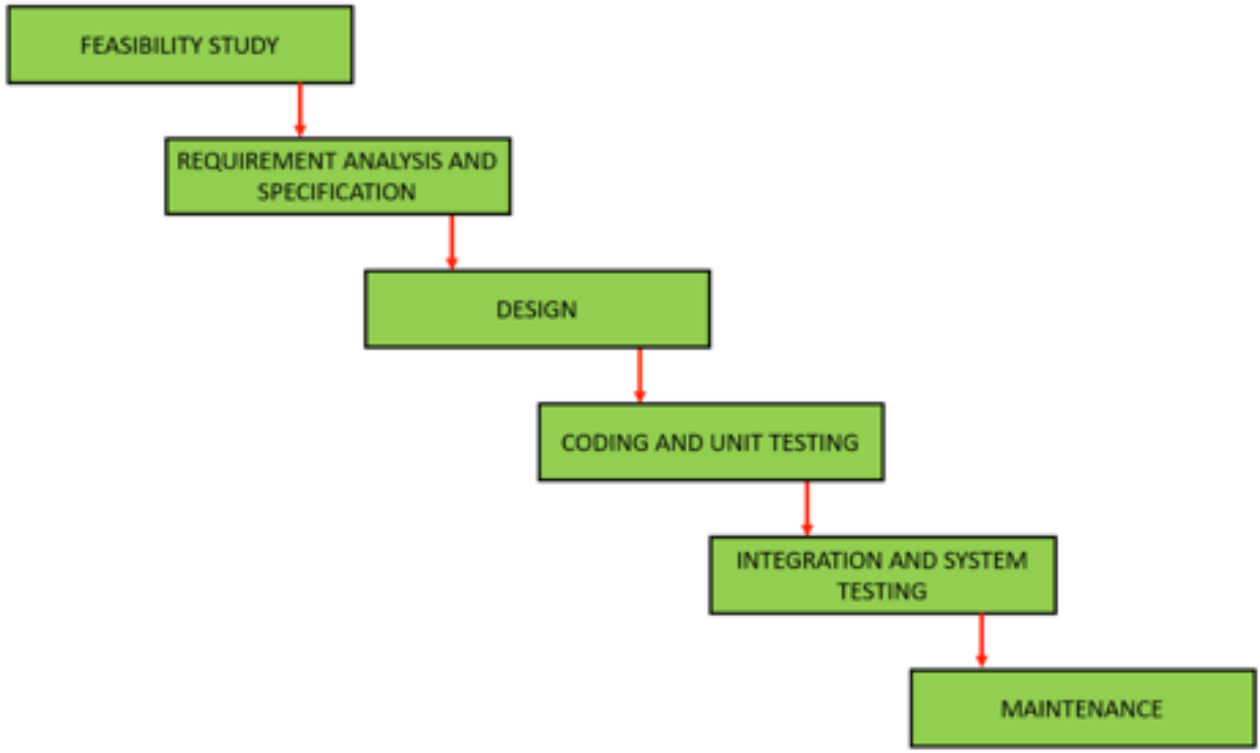


Continuous (integration, testing, deployment and operation)

Business value of services requires high quality continuous operation: e.g., response time of data search portal, availability of services when number of user increases …,

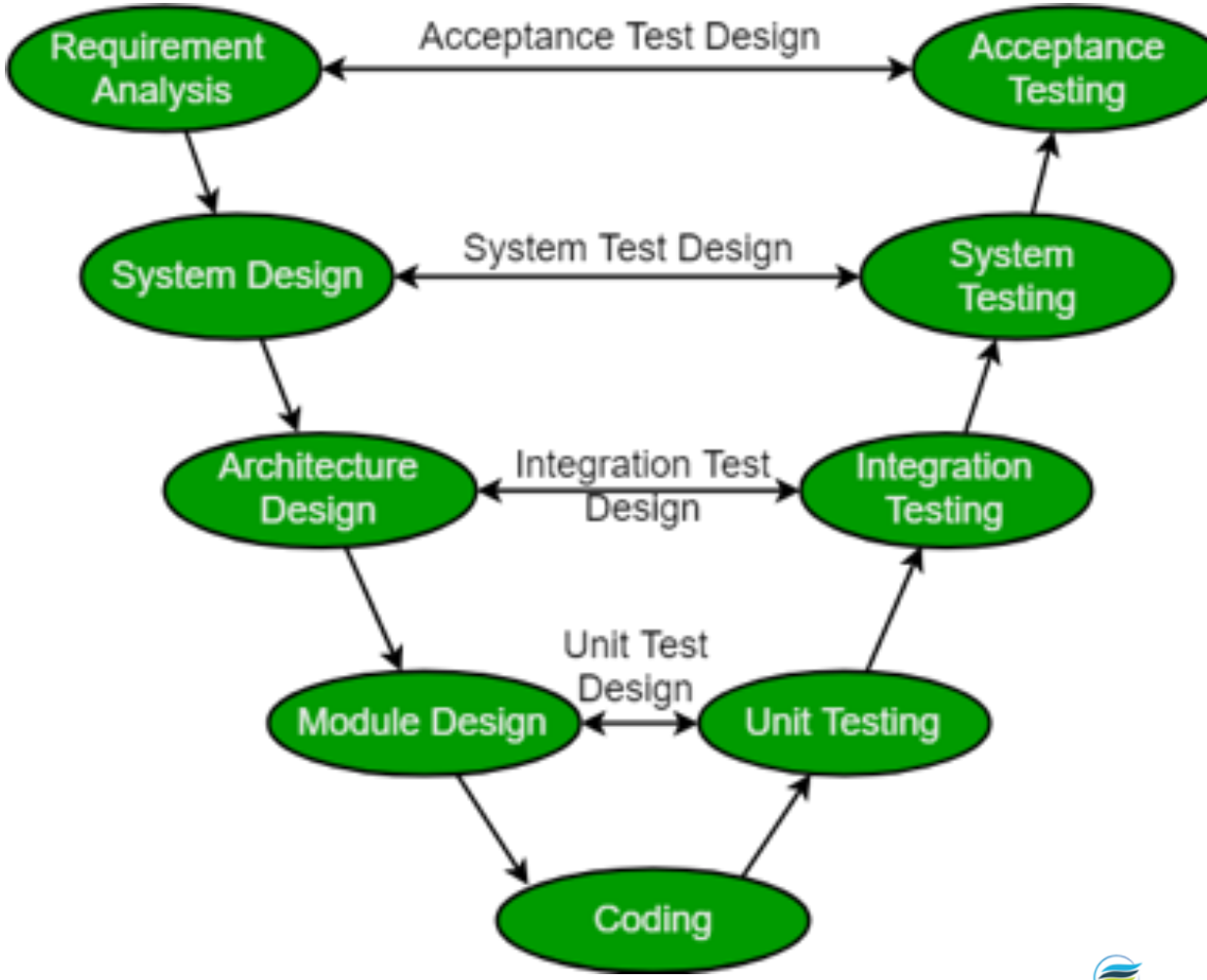
Waterfall model

- Waterfall model, 1970
- Sqeuntial phases: one step (phase) depends on the output of the previous one
- Require detailed description of requirements and design documents
- High risks if requirements are not complete or changing.



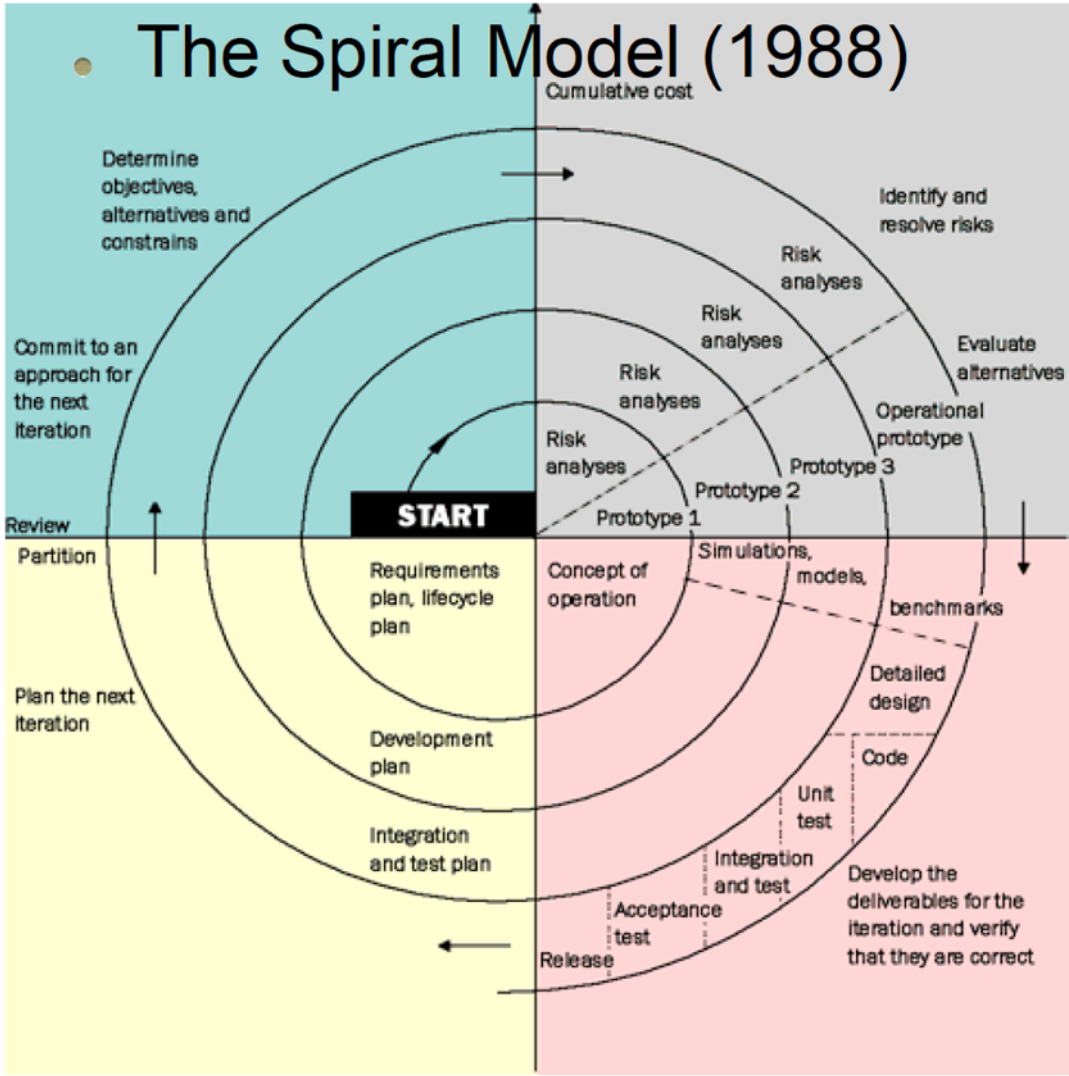
V-Shaped model

- V-Shaped model (1991)
- Reduce risks by early planning of the testing phases:
  - Acceptance, system, integration and unit
- Did not fundamentally change the problem of waterfallmodel: difficult to handle changing requirements. Require detailed planning.



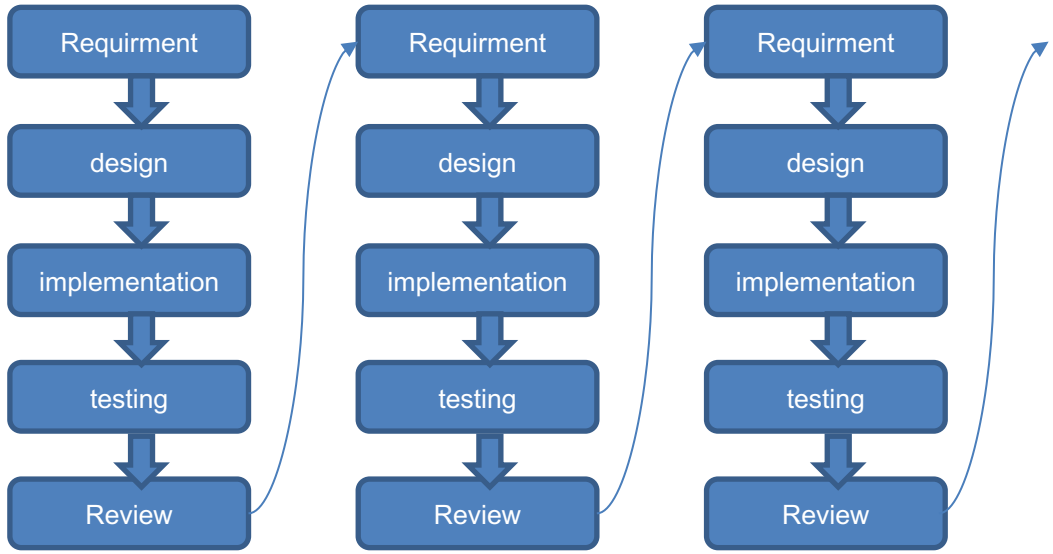
Spiral model

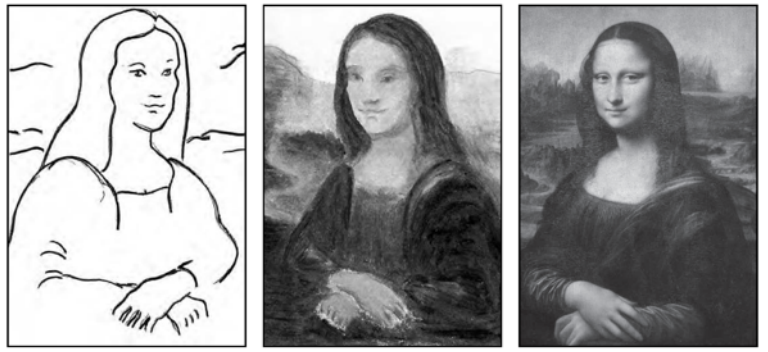
- Spiral model, 1988
- Steps (phases) based, but provide opportunities for multi feedback
- Explicit risks management
- Complex. High time cost and expertise required for planning and doing risk analysis.



Iterative model

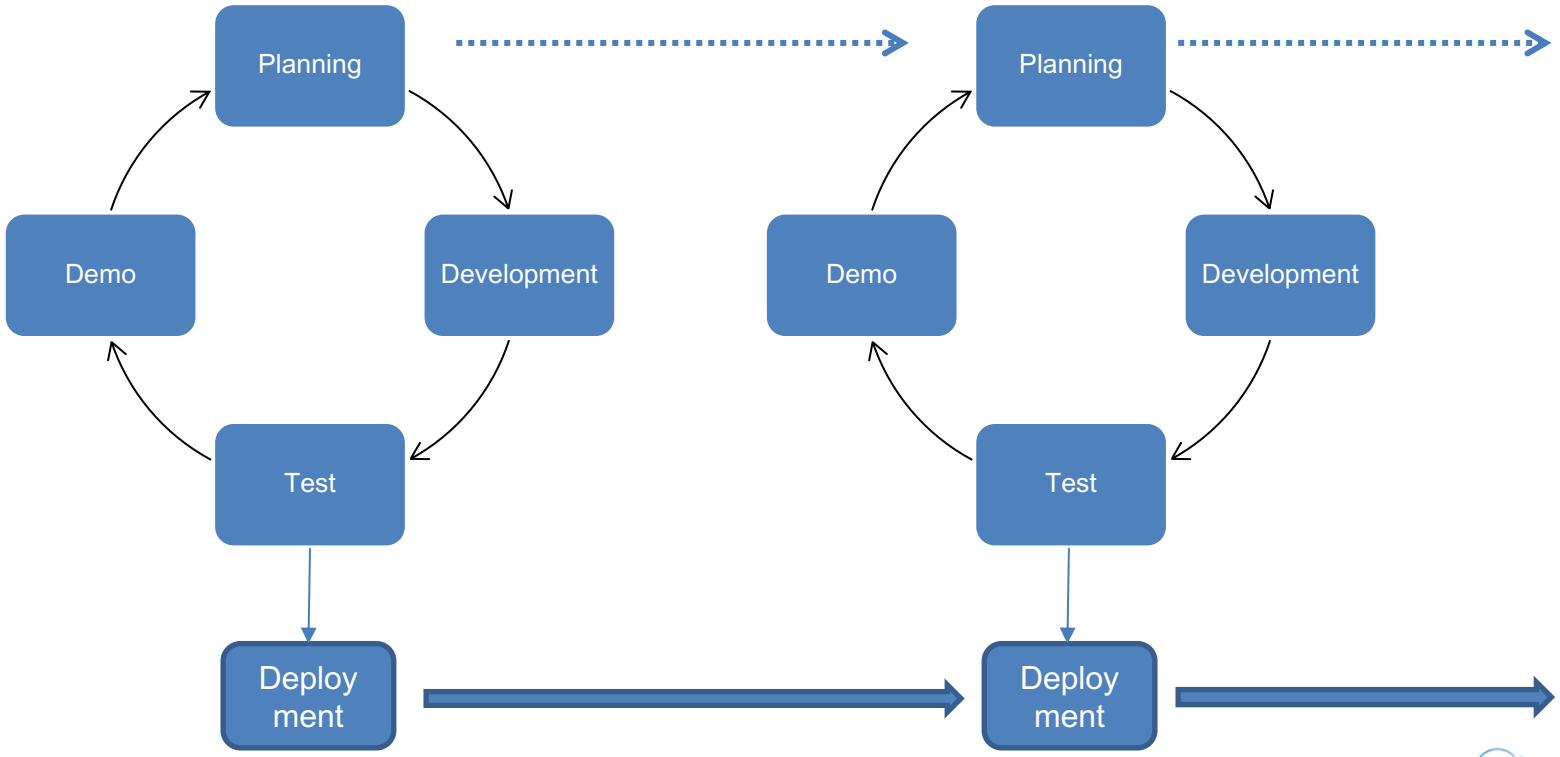
- Iterative model, 1995
- Lifecycle phases are decoupled from software engineering activities
- Each iteration mitigates the risks of current activities
- Require high customer engagement, and efficient communication and effective planning.



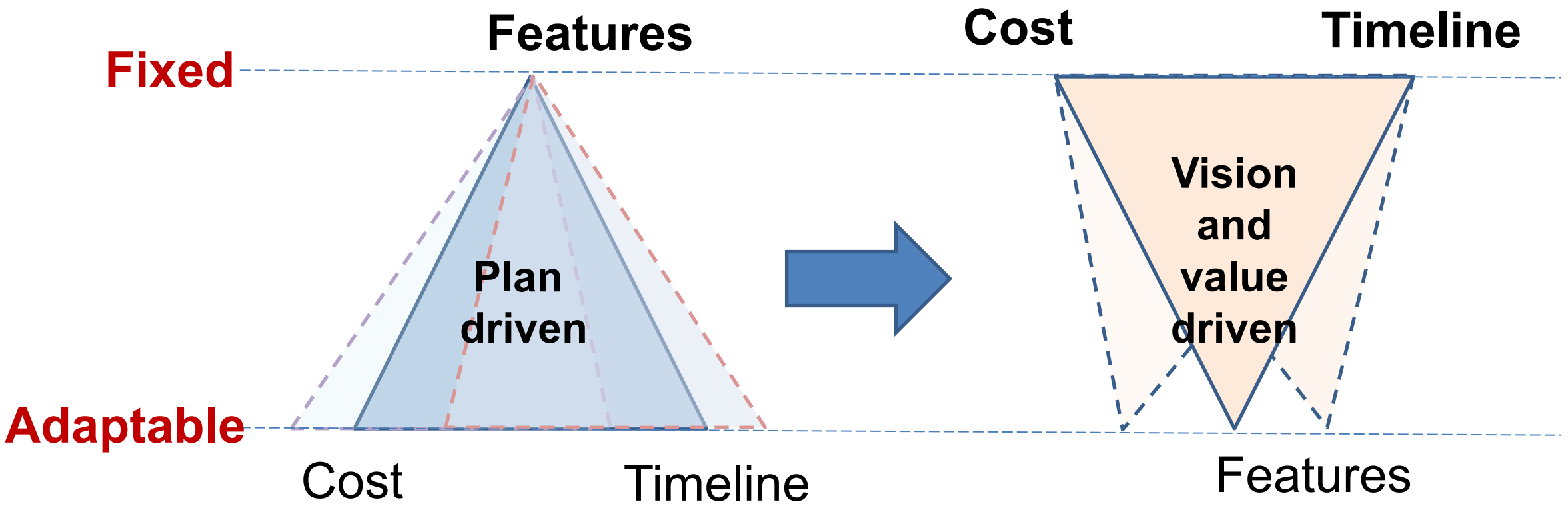


Agile model

- Improved from incremental model
- Planning based on new requirements, priority, production roadmap, …
- Rapid cycle
- Require special skills of the development team

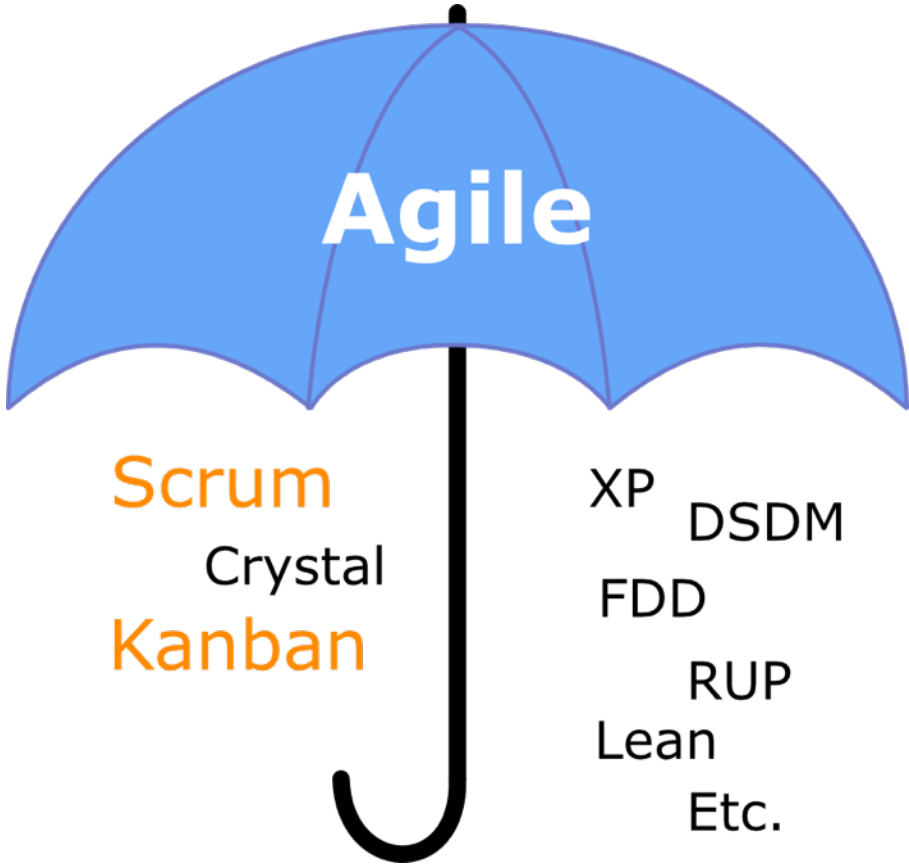


Think it in agile way?



Tools under the umbrella of Agile

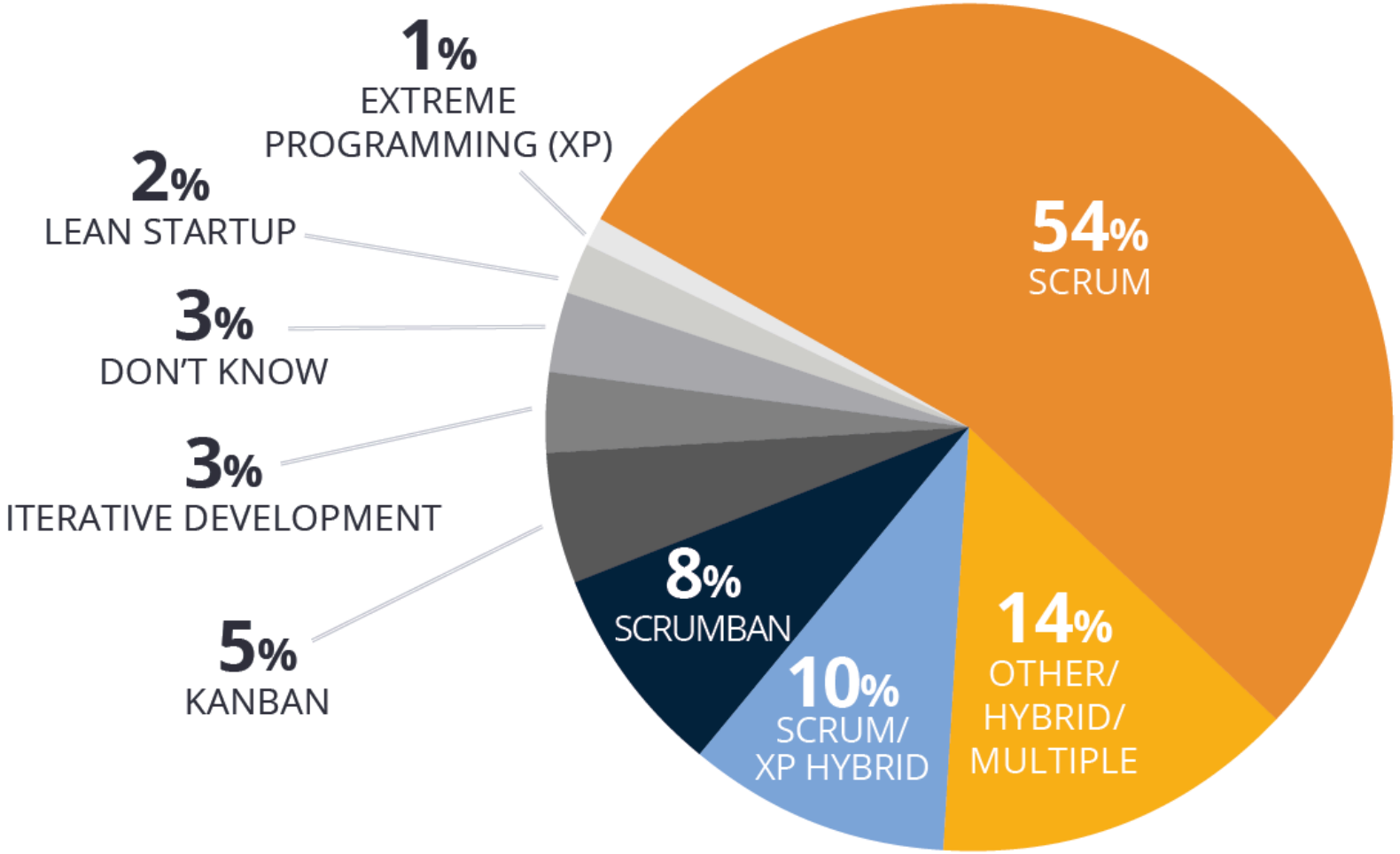
- Feature Driven Development (FDD)
- Dynamic System Development Method (DSDM)
- Behavior Driven Development (BDD):
- Extreme program (XP)
- Kanban
- Crystal
- Lean
- Test Driven Development (TDD)
- Scrum
- …



Ref. Malek Al-Zewairi et al.: Agile Software Development Methodologies: Survey of Surveys, JCC, V5 (5)
https://m.scirp.org/papers/75114

Practice

- SCRUM is most popular



13th Annual stage of AGILE report: https://www.stateofagile.com/#ufh-c-473508-state-of-agile-report

Using scrum

- Role:
  - Product owner, Scrum master and Development team
- Artifacts
  - Product backlogs, user stories, sprint backlogs, and burndown chat
- Meetings
  - Sprint planning, Daily scrum, Sprint review
- Workflow

Scrum workflow (customizable)

Product owner

Product backlog

User stories

As a___
I need ___
So that
___

Scrum master

Development team

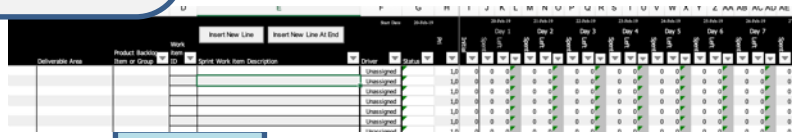
Scrum workflow (customizable)

Product owner

Product backlog

User stories

As a___
I need ___
So that
___



Scrum master

Sprint backlog

Sprint planning

Two parts:

1. **Check product backlog**, and decide the scope of next release
2. **Defines tasks** for the sprint.

A sprint is typically 2-4 weeks.

Development team

Scrum workflow (customizable)

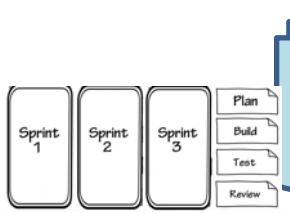
Product owner

Product backlog

User stories

As a___
I need ___
So that
___

Scrum master



Sprint backlog

Sprint planning

1. What did you yesterday (since the last scrum meeting)
2. What will you do today (before next scrum meeting)
3. What obstacles?

Daily Scrum

Development team

Scrum workflow (customizable)

Product owner

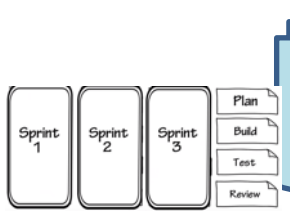
Product backlog

User stories

As a___
I need ___
So that
___

1. What did you yesterday (since the last scrum meeting)
2. What will you do today (before next scrum meeting)
3. What obstacles?

Scrum master

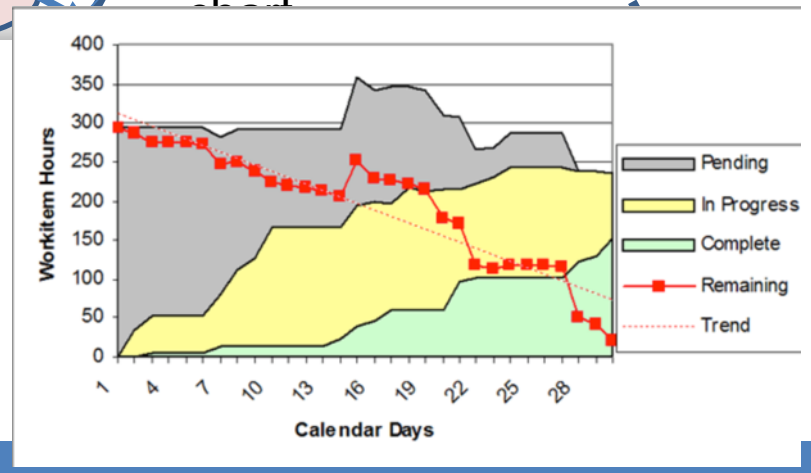


Sprint backlog

Sprint planning

Burn down chart

Daily Scrum



Development team

ENVRI FAIR

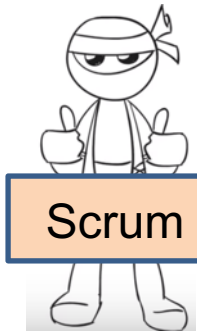
Scrum workflow (customizable)

Product owner

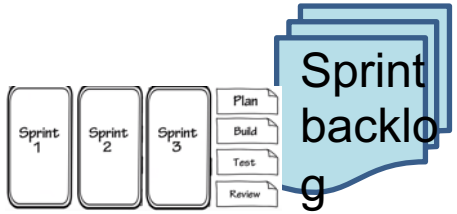
Product backlog

User stories

As a___
I need ___
So that
___



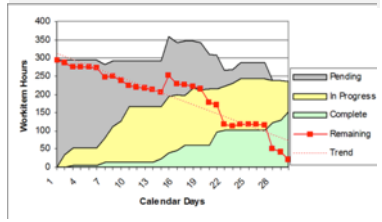
Scrum master



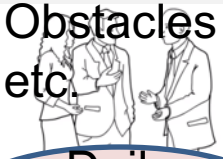
Sprint backlog

Sprint planning

Burn down



Did yesterday,
To do today
Obstacles
etc.



Daily Scrum

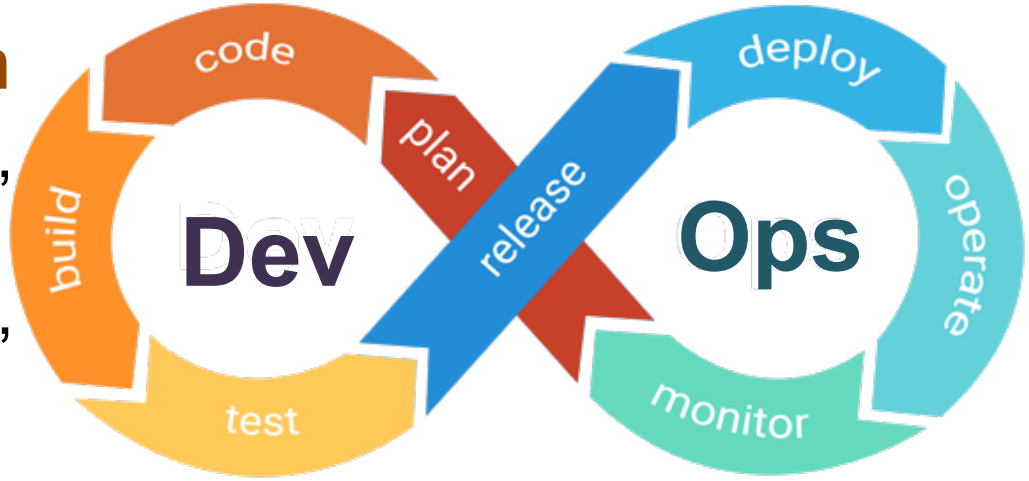
Development team

Sprint review, retrospective

Business and ICT aspects of services

Software implementation of services: e.g., Service Oriented Architecture (SOA), web services, micro services, business process modelling (BPM), etc.

Agile, incremental, iterative

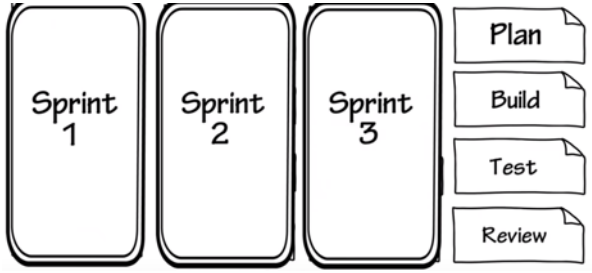


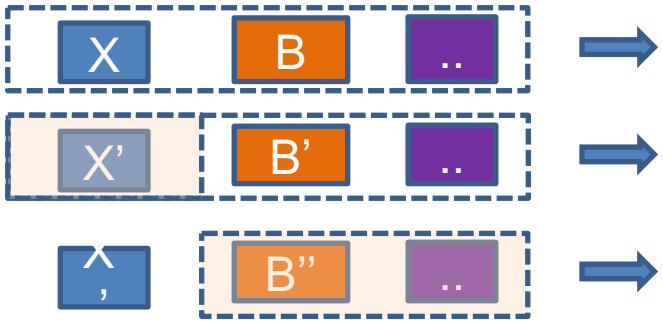
Continuous (integration, testing, deployment and operation)

Business value of services: e.g., response time of data search portal, availability of services when number of user increases …,

Teamwork and version control

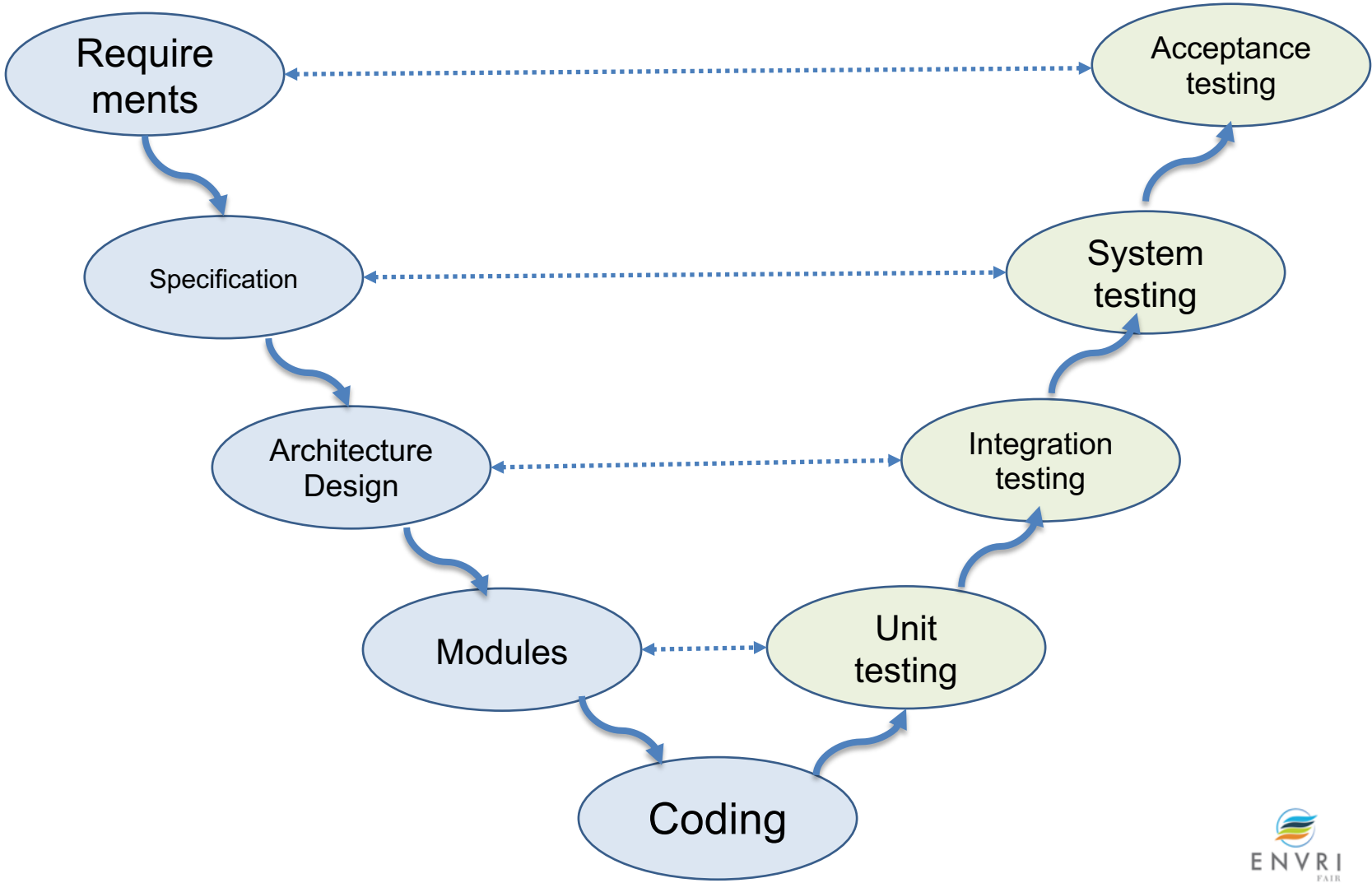
- **Contributions** from each team member need to be integrated
- The implementation of each component might be **iterative and incremental**





Software testing is needed at different phases

- V-Shaped model
- **Unit testing**
  - White box, code level, functional units, typically done by developers
- **Integration test**
  - Integration
- **System test**
  - Performance, non-functional testing
- **Customer acceptance testing**

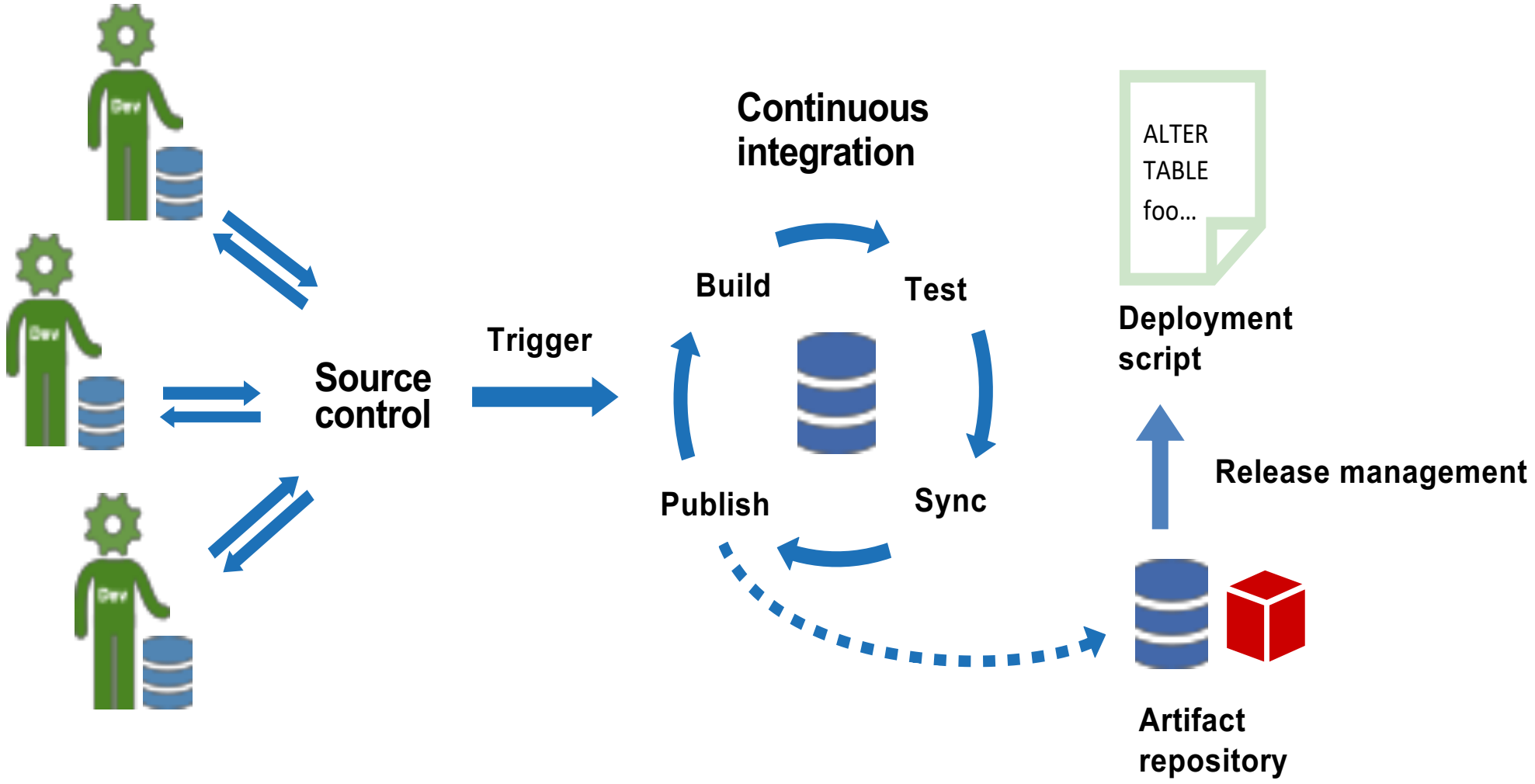


Software need test

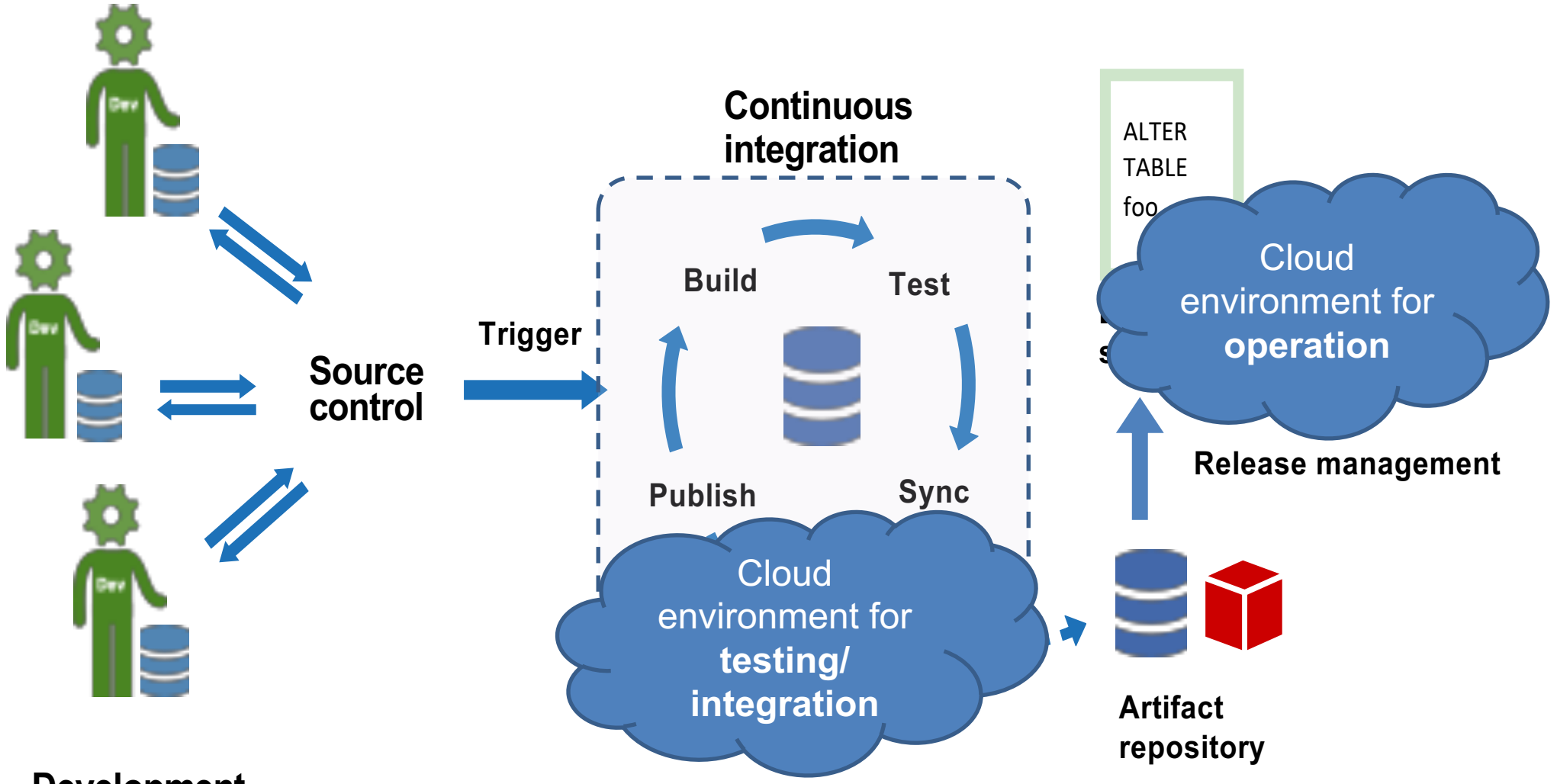
- **Verification:** did we develop the things **correctly?**
  - Check if the software delivers the expected function as the requirements define
  - Check if the software can run as we expected in the scenarios defined by the requirements

- **Validation:** did we develop **correct** things**?**
  - Check the system can behave correctly in real application, more than the features or scenarios defined in the requirements.

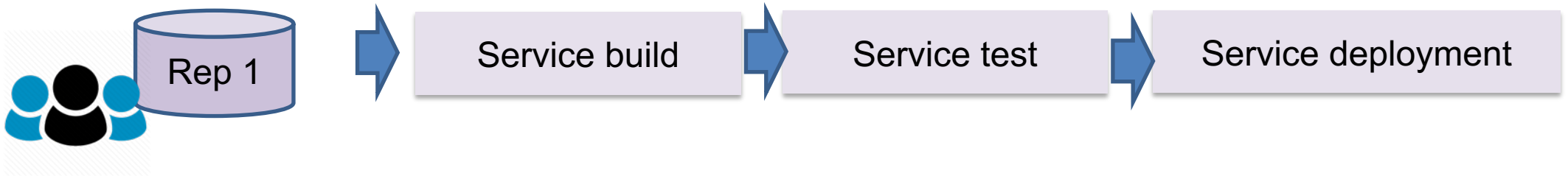
Continuous testing, integration and deployment



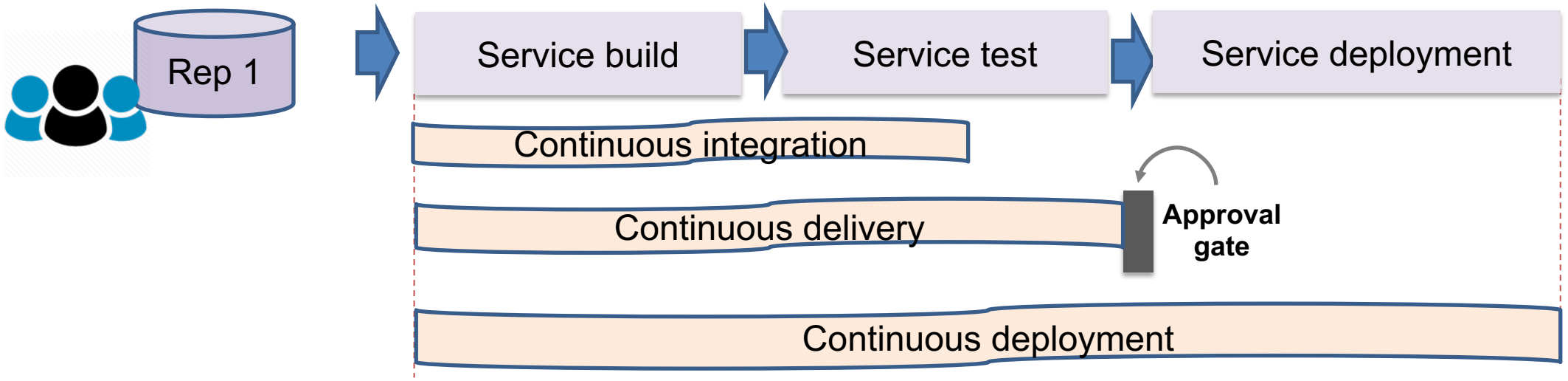
Continuous testing, integration and deployment



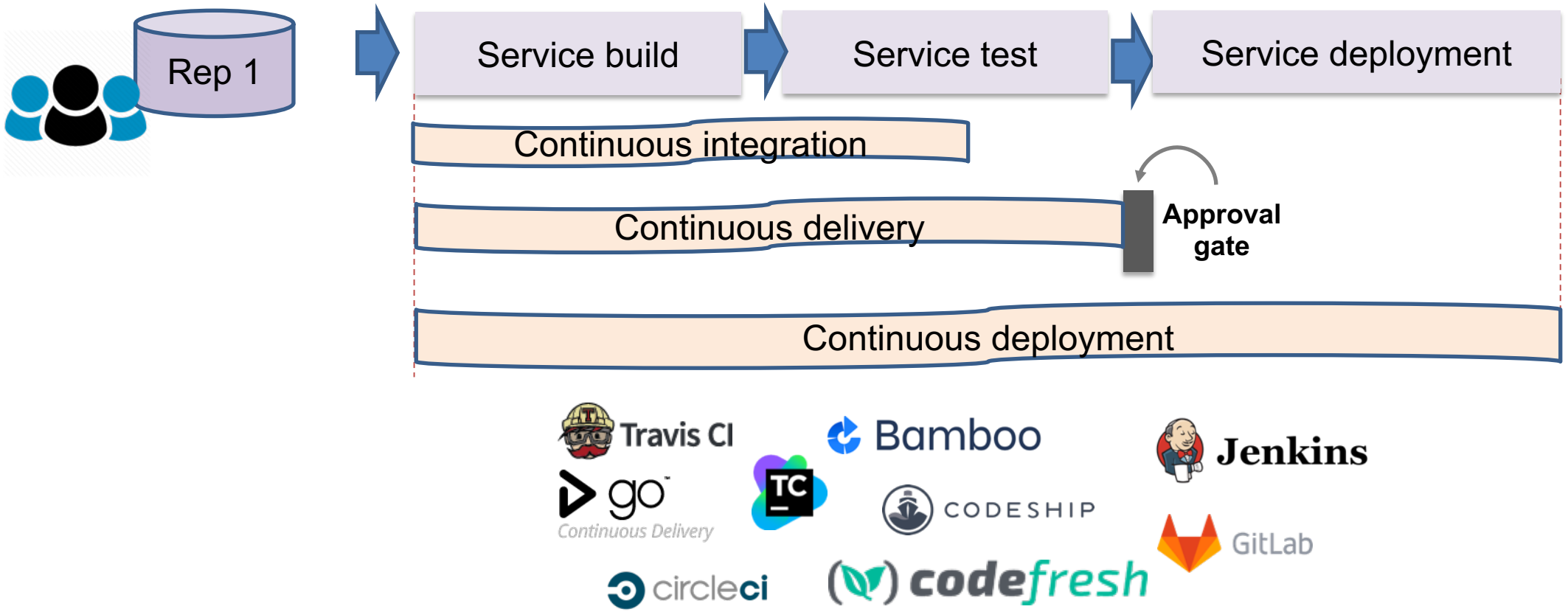
Continuous integration, delivery and deployment



Continuous integration, delivery and deployment

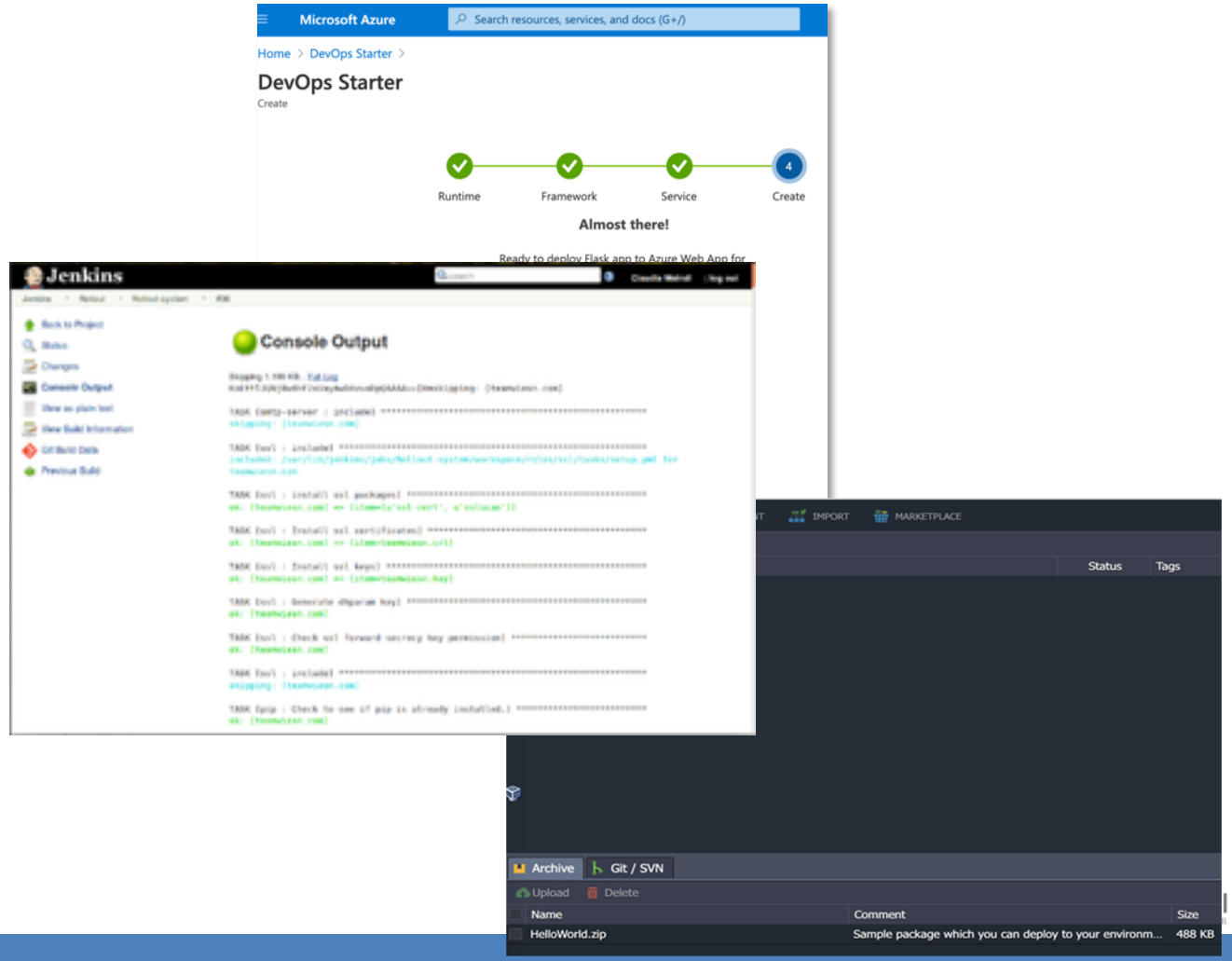


Continuous integration, delivery and deployment



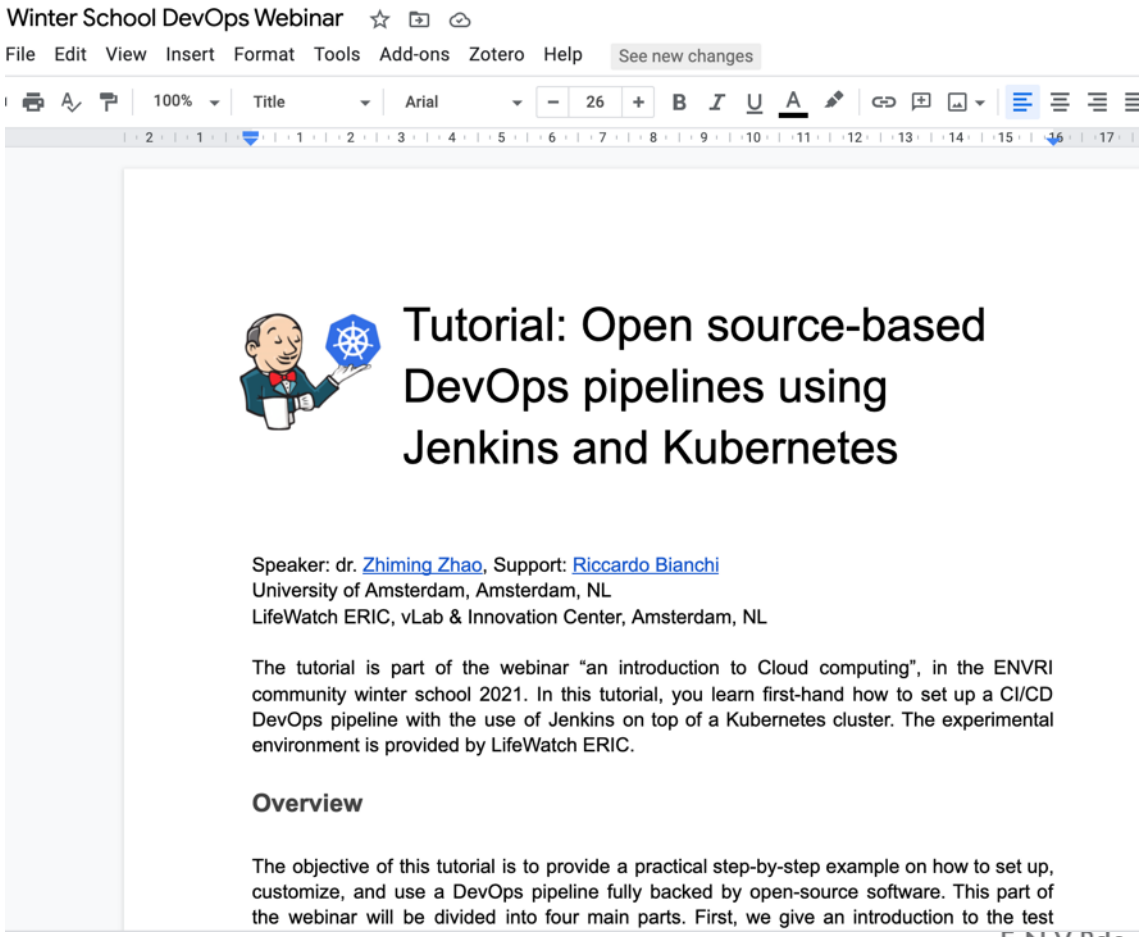
Automated framework

- Public cloud provider
  - AWS
  - Azure
- EOSC
  - Jelastic
- Open-source solution
  - Jenkins



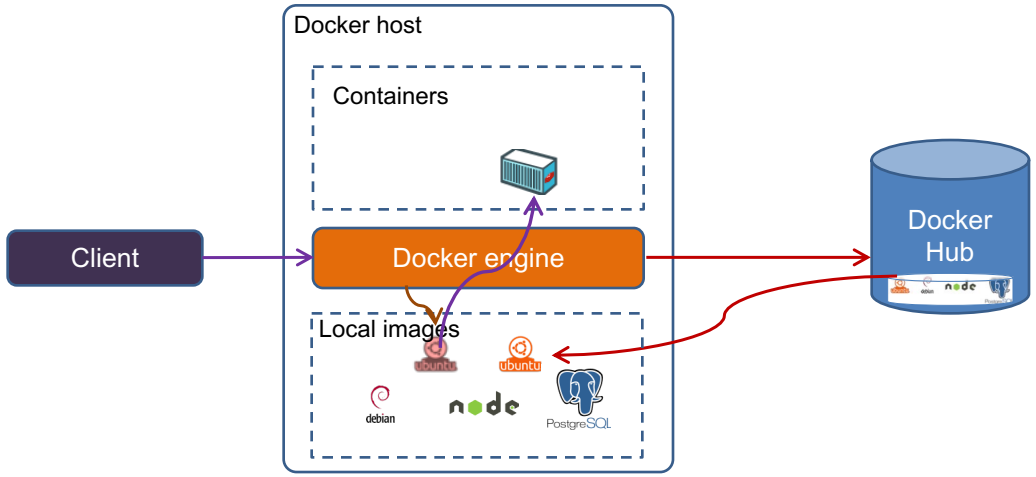
In the practical assignments

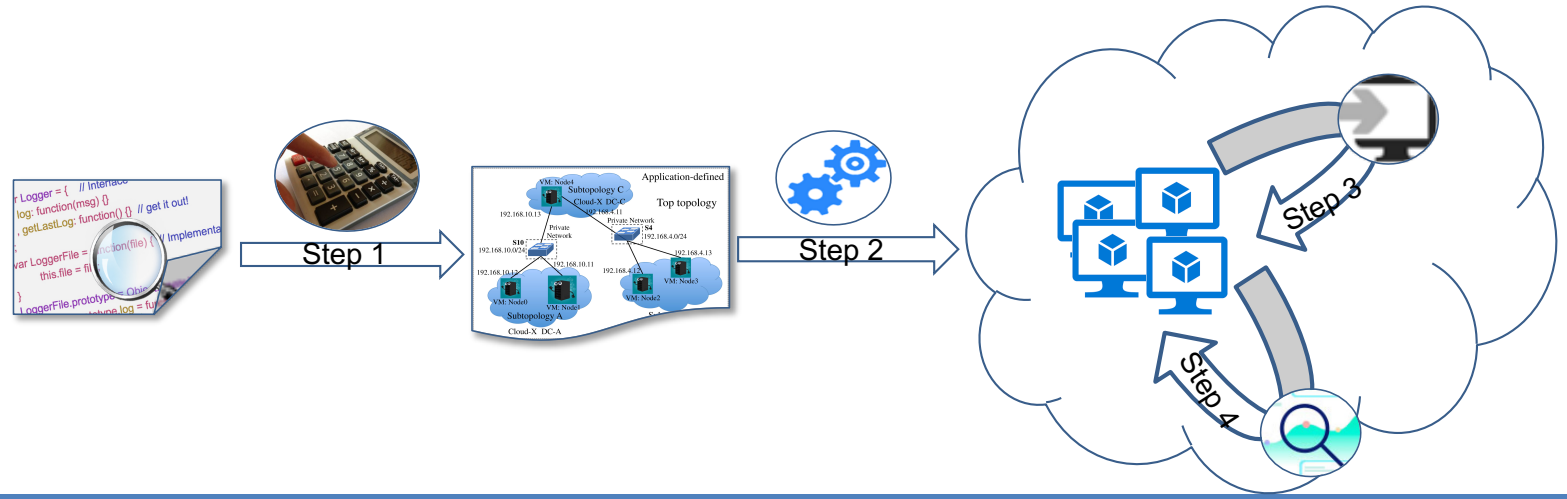
- How to set up an automated framework
- Link: **http://shorturl.at/mIQW0**



Summary 1: run an application in cloud

- Plan a virtual infrastructure
- Provision it in cloud
- Deploy software
- Execute and monitor

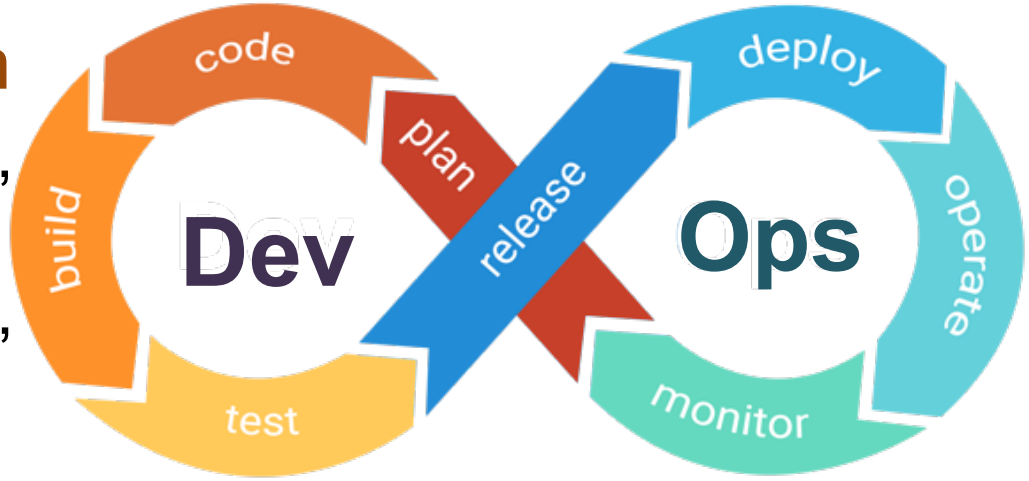




Summary 2: software development and operation

Software implementation of services: e.g., Service Oriented Architecture (SOA), web services, micro services, business process modelling (BPM), etc.

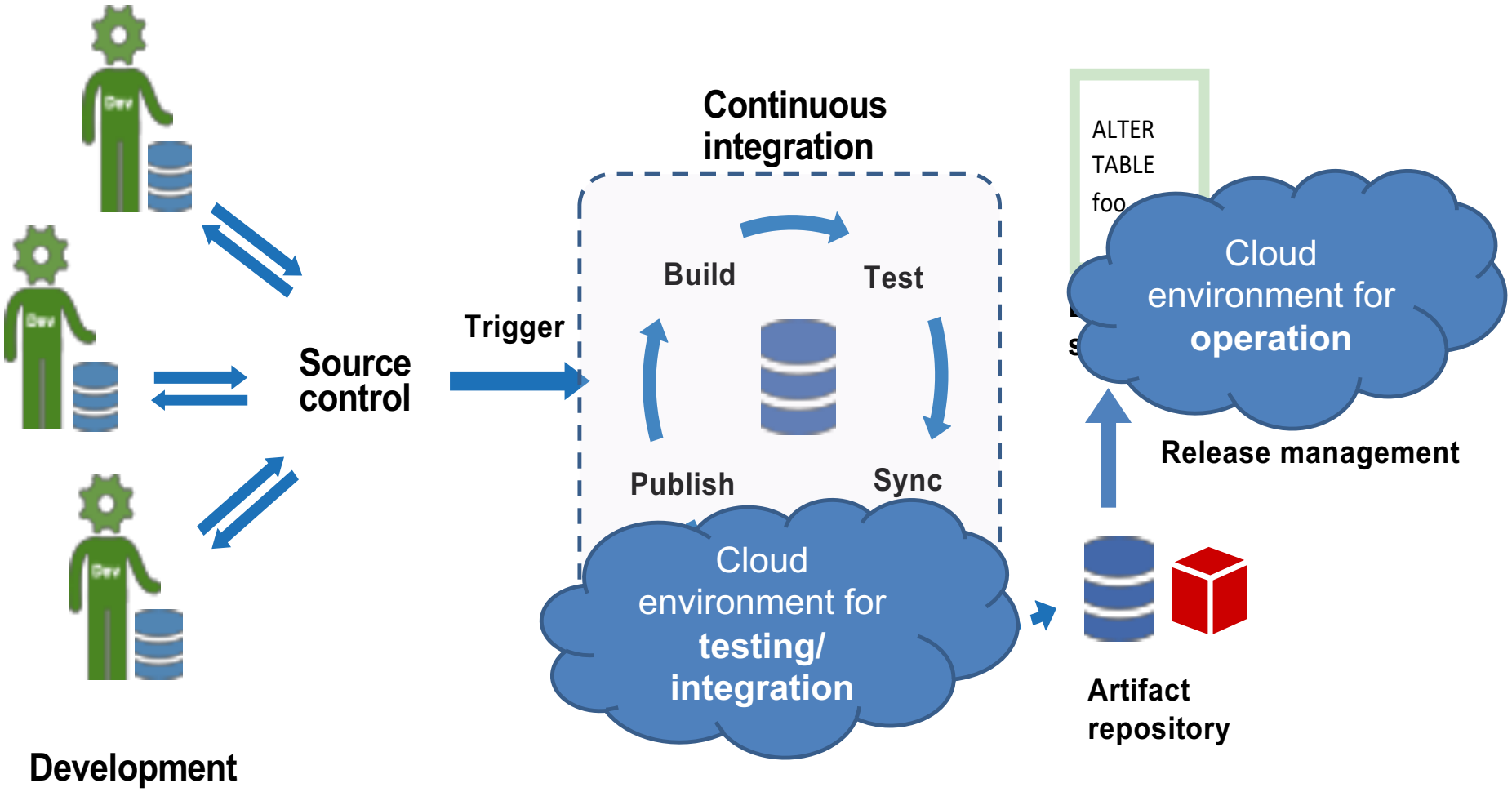
Agile, incremental, iterative



Continuous (integration, testing, deployment and operation)

Business value of services: e.g., market, number of customers, community interests, etc.

Summary 3: Continuously integration and deployment



Lab assignment

- Tutorial 1: create a container using Jupyter FAIR-CELL extension
  - **http://shorturl.at/bjMUV**
  - Containerize your code via Jupyter
  - Publish the container in Docker hub
  - Setup a Kubernetes cluster
- Tutorial 2: run the docker in a Kubernetes cluster
  - **http://shorturl.at/dnBF9**
  - Deploy the container in the Kubernetes cluster
- Tutorial 3: Jerkins, opensource DevOps pipeline
  - **http://shorturl.at/mIQW0**
  - Setup an open-source continuous testing/deployment framework using Jerkins in cloud
  - Test how the pipeline works (triggered by the changes in the code)
- Tutorial from the previous webinar: RESTful service, dockerization, K8S
  - http://shorturl.at/dorGX

Discussion

- Feedback and questions?