Cloud computing

Made by Yoni Krichevsky
ITC

Agenda

- Cloud computing – what it is
- Pluses and minuses
- Different layers
- Providers
- Practical example
  - Typical Python application deployment flow in IaaS
  - Security, SSH
  - Class exercise deployment on AWS

Cloud computing

Purpose of a company

Purpose of a software company

How to get there

IT infrastructure

< itc >

Resources:

- Compute (CPU)
- Storage (hard disks)
- Memory (RAM)
- Databases
- Network – routers, cables…
- …

IT activities:

- Buy
- Host – rooms, air conditioning
- Install, upgrade
- User management
- Security patches
- Monitoring, fixing
- Support users
- Learn, train, hire experts

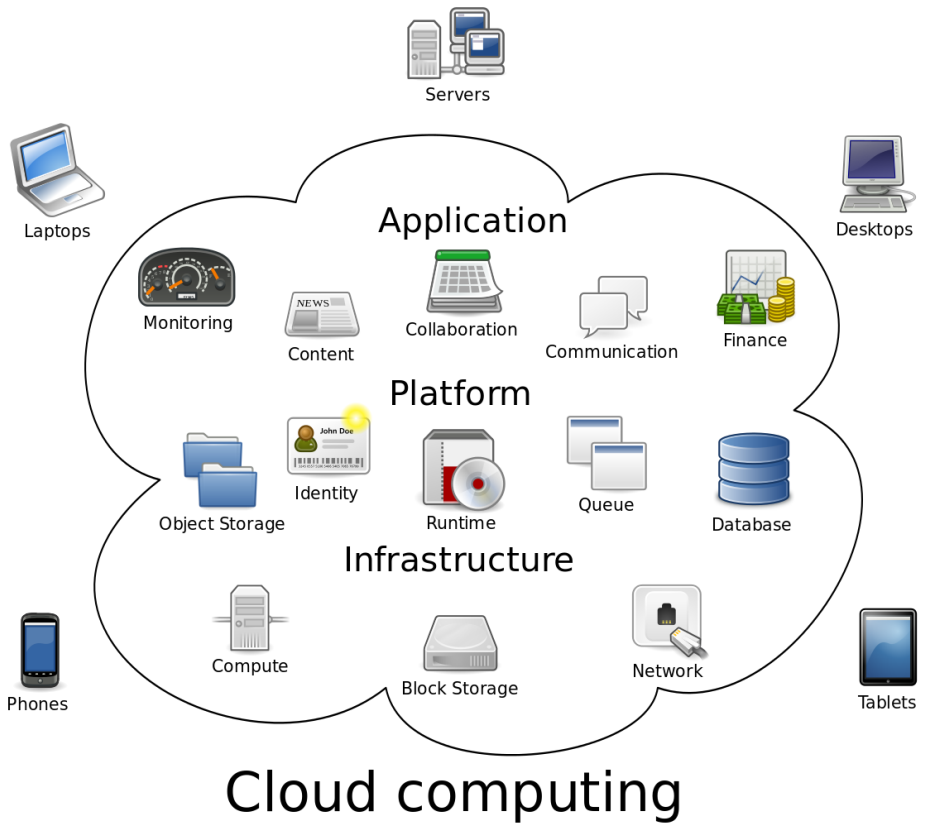
Before the cloud

Before the cloud

- **On Prem** (on Premises) – Every organization has all the IT infrastructure physically at company campus

- Pluses:
  - In control

Cloud computing

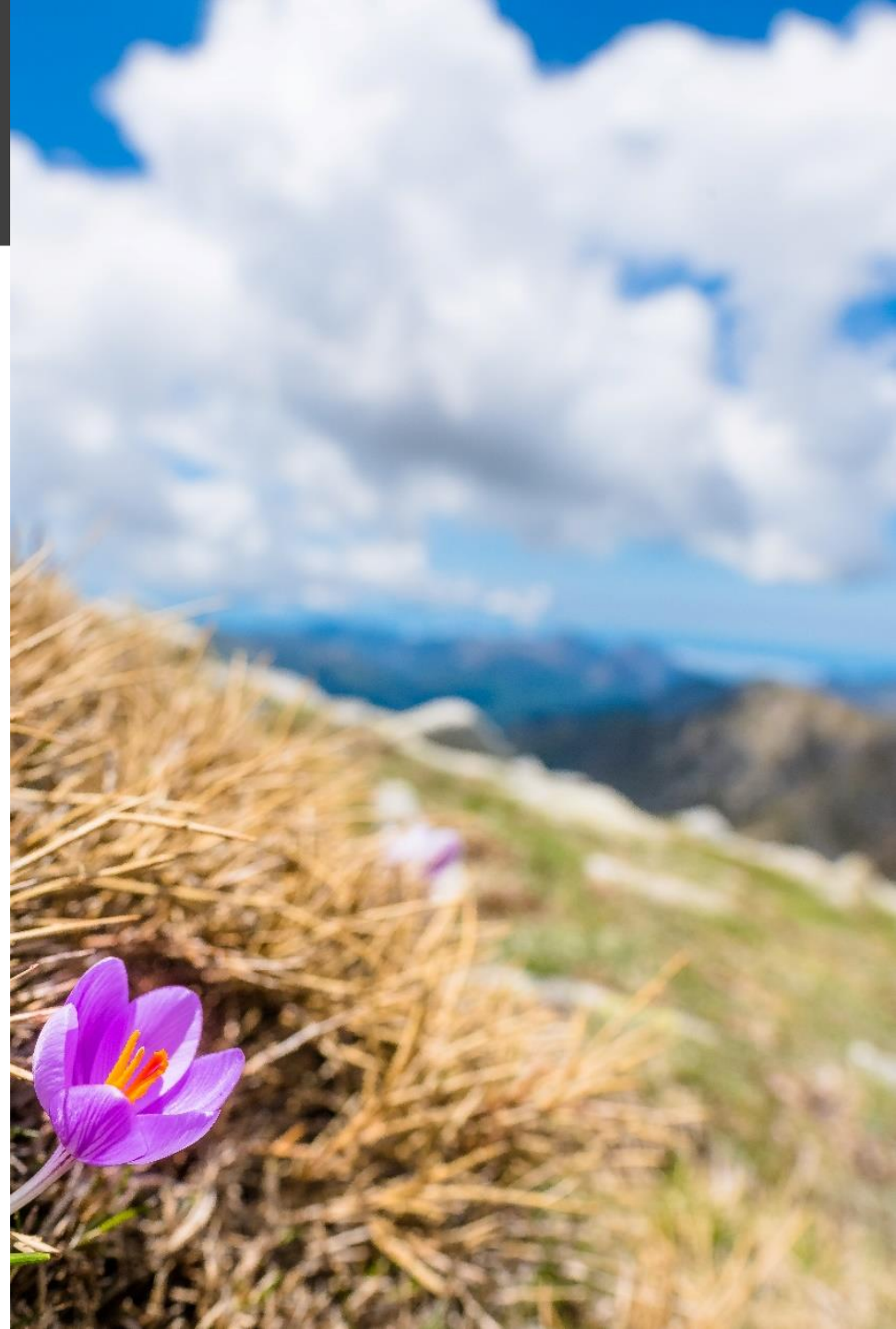


- **Cloud computing** – buying on demand computer resources and services over the internet
- **Economies of scale** – resources shared between customers
- **Pay-as-you-go** model

https://en.wikipedia.org/wiki/File:Cloud_computing.svg

Cloud computing – pluses

Focus on core business



Cloud computing – pluses

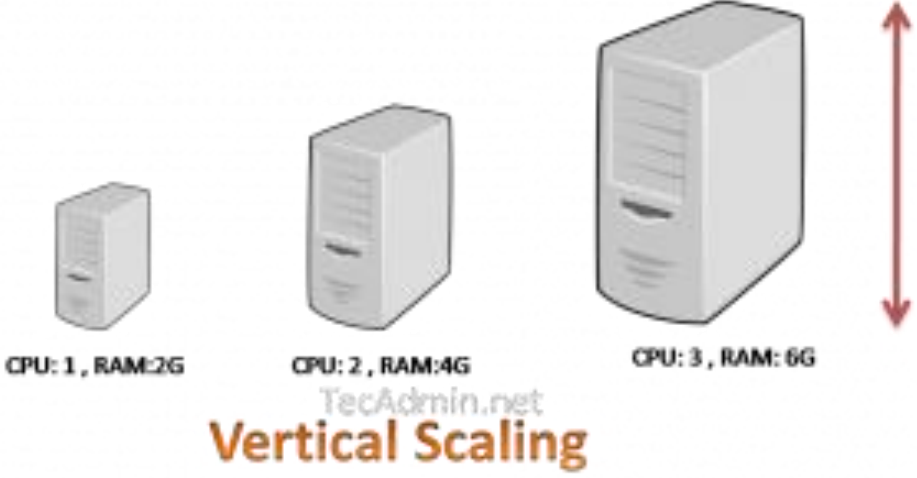
Scalability

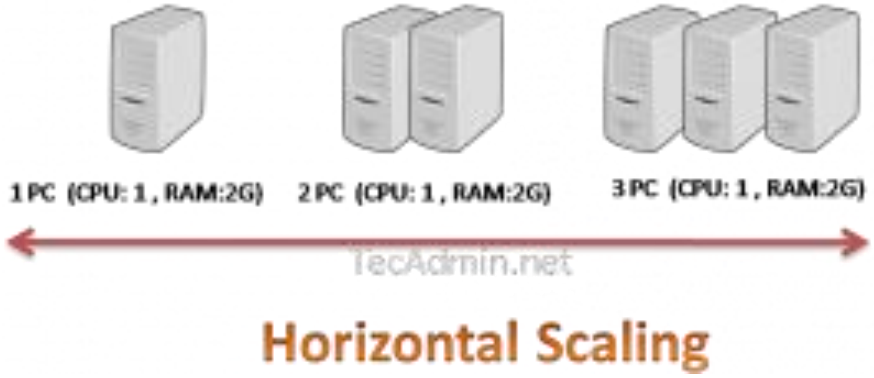
- Adaptivity to rapidly changing demand
- **Scale up** on demand, then **scale down**

- Scalability – **Horizontal, Vertical**

- How to scale: manually, or automatically:
  - Time based – only weekends
  - Load based – CPU > 70%
  - Latency based – response > 10 ms

- Unlimited resources from client perspective





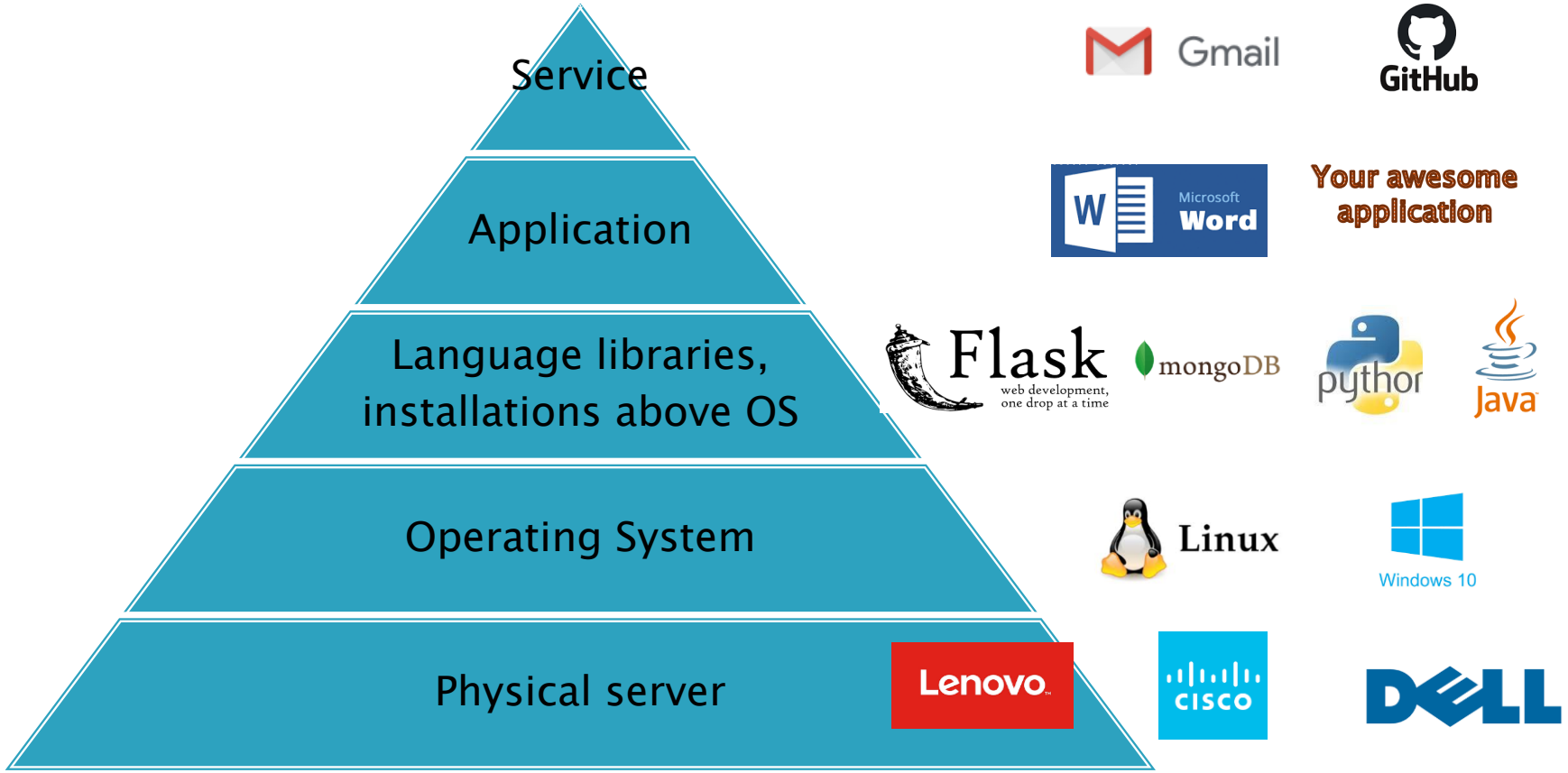
Cloud computing – mixed blessings

- Less in control – good or bad?
- Security, regulation
  - Same, or better, but different. Part of the specialization outsourced to experts
- Cost
  - Very large organization that sometimes do it cheaper themselves
  - Need to pay attention to bills and resources used
- Cloud vendor lock-in – can I switch?
- Latency (time for response) – are your customers local or global?

Possible solution:
Hybrid cloud – part On Prem, part in the cloud

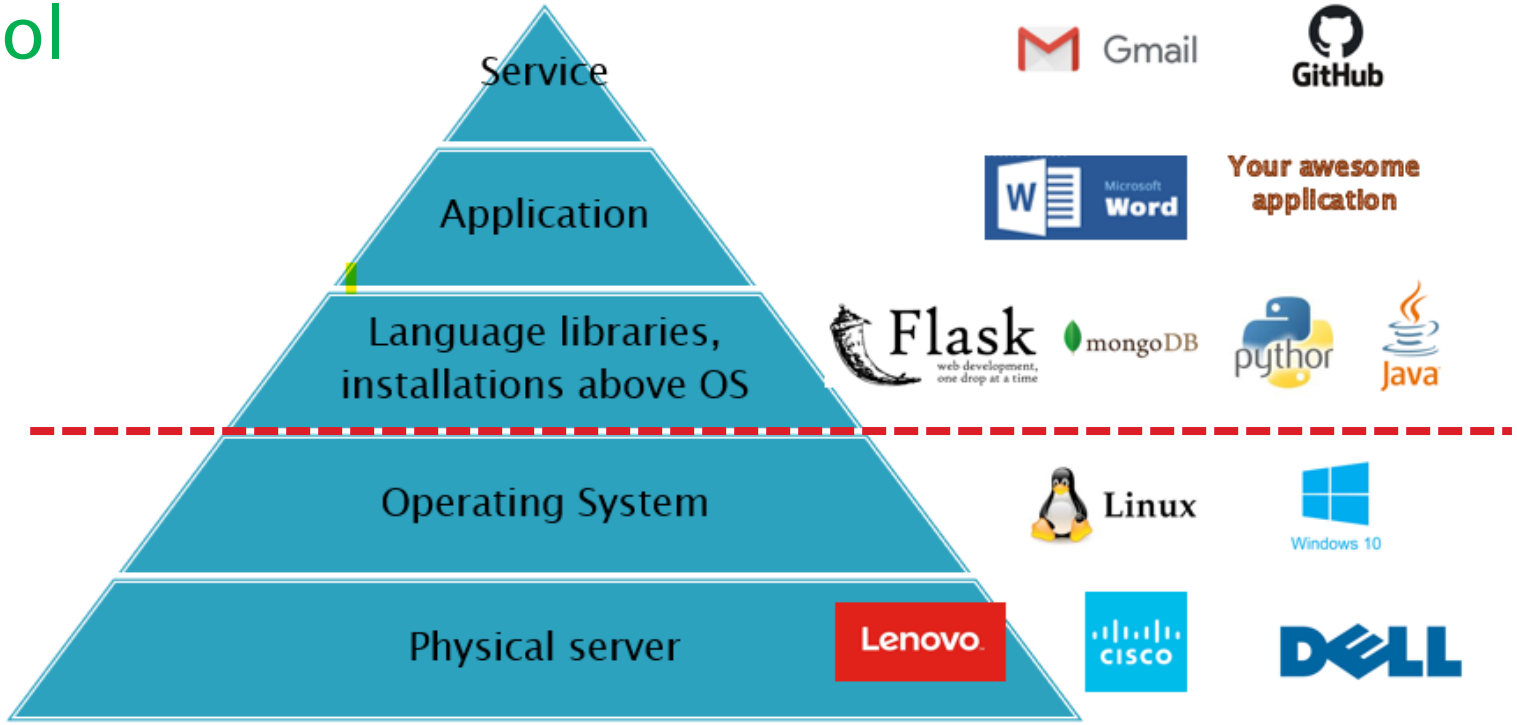
Cloud computing – layers

Computing Layers

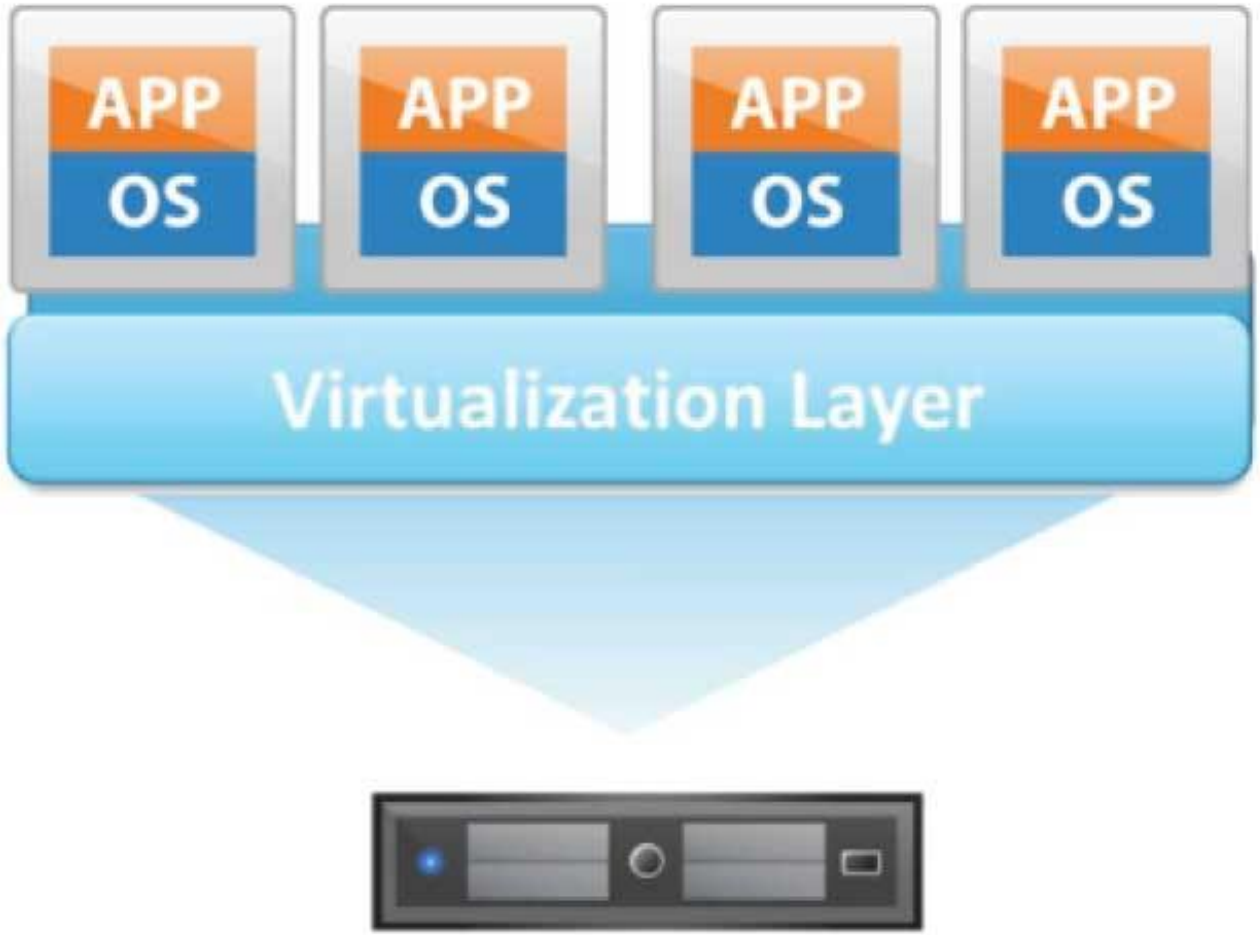


Computing Layers – IaaS

- Infrastructure as a service
  - Get servers
  - Do everything above on your own
- Most complex. Need DevOps support
- Most freedom and control

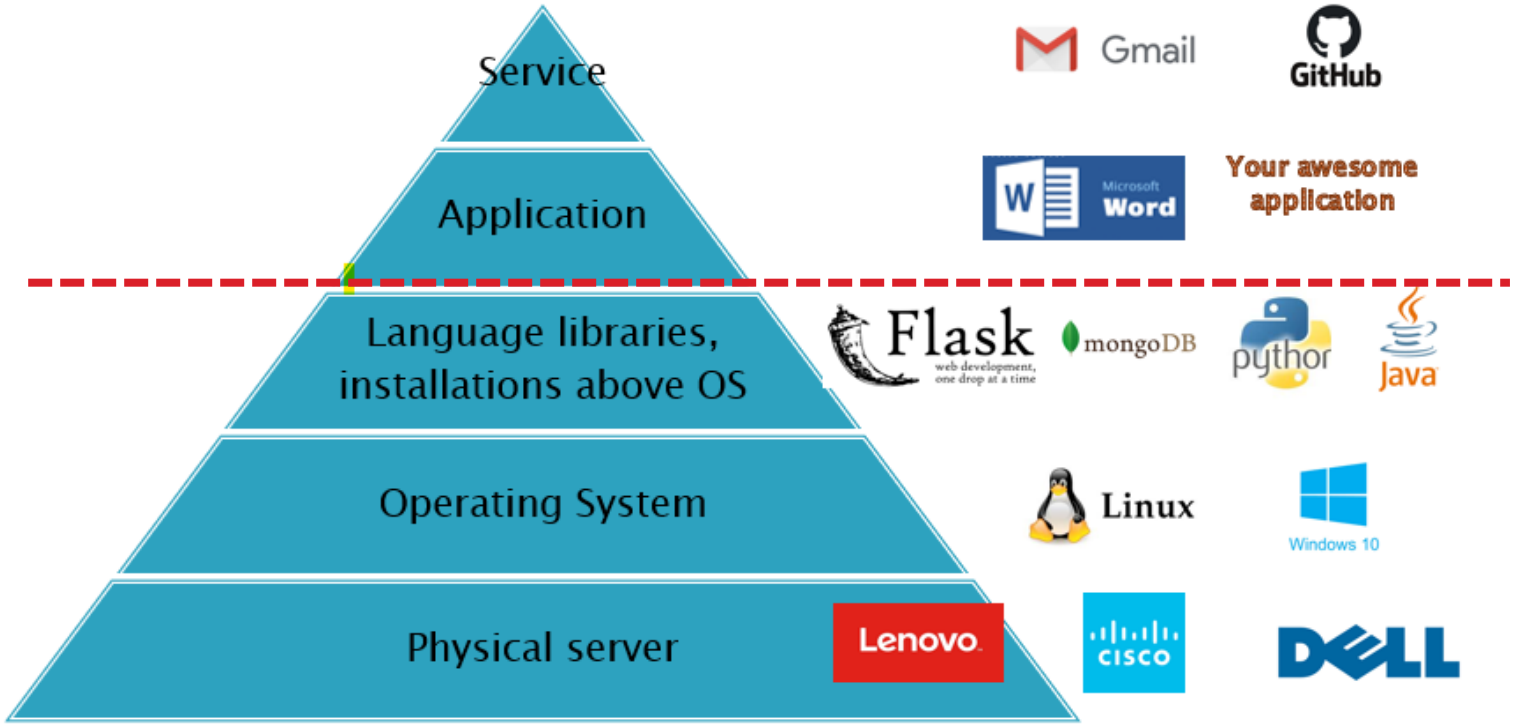


Virtualization



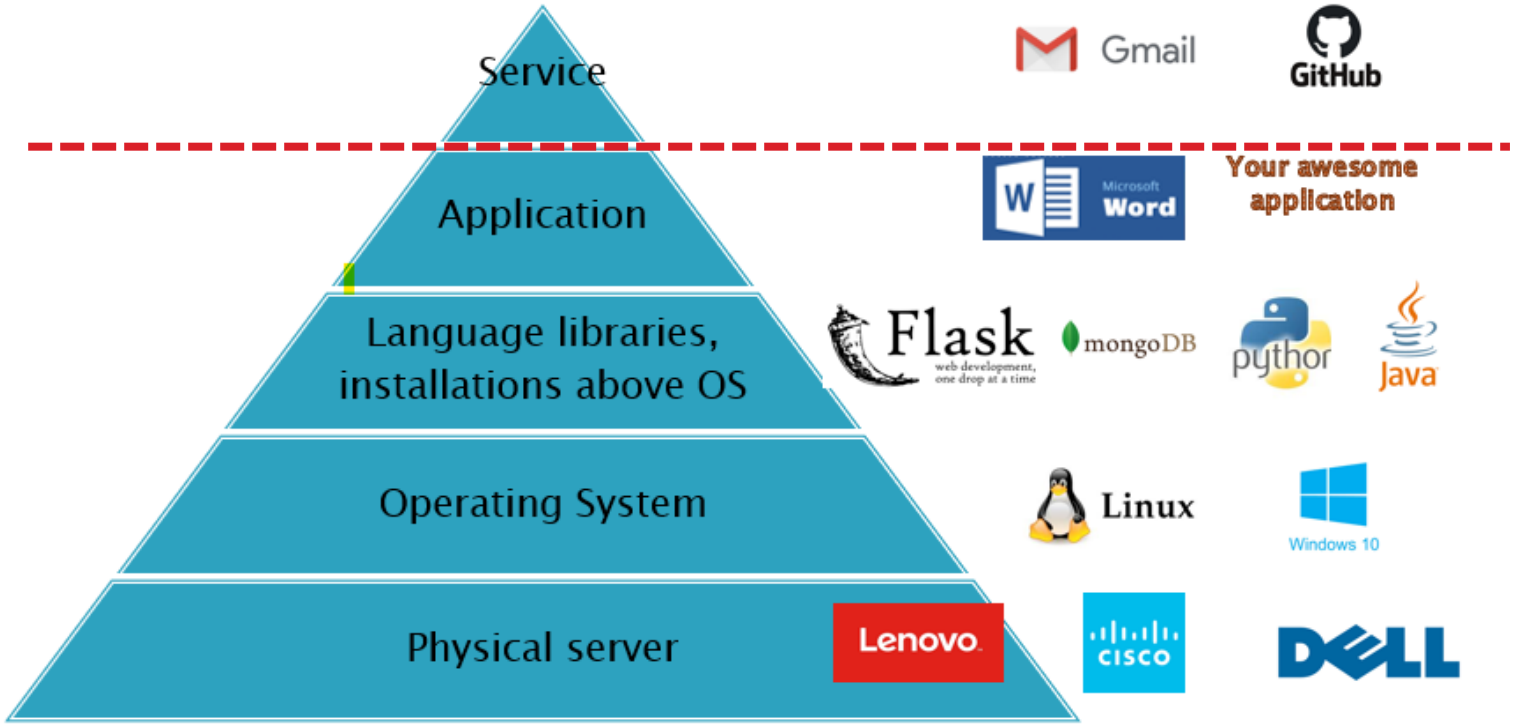
Computing Layers – PaaS

- **Platform as a Service**
  - Get environment with all the pre-requisites
  - Just deploy your application
- Less complex
- Less freedom



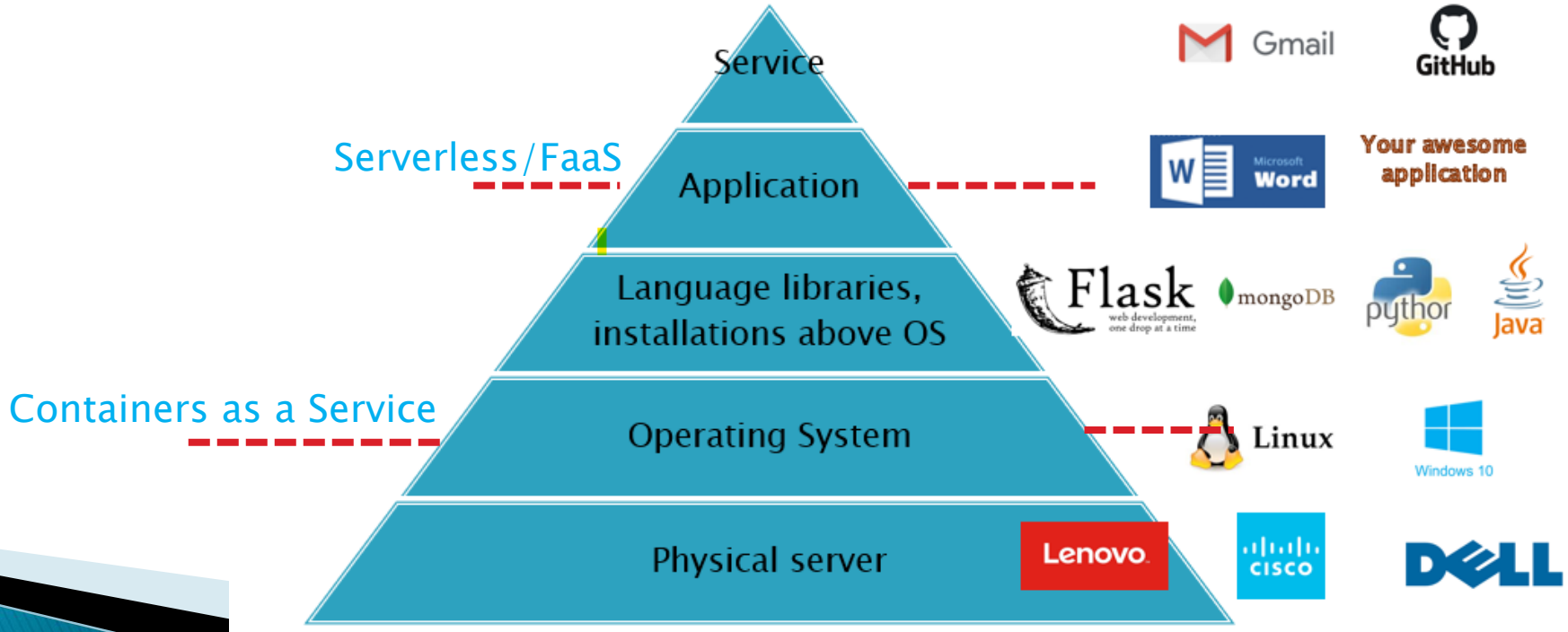
Computing Layers – SaaS

- **Software as a Service**
  - You don't deploy anything, you are only a user
  - Just access a service via internet
  - By browser etc.
- Google Docs, Github



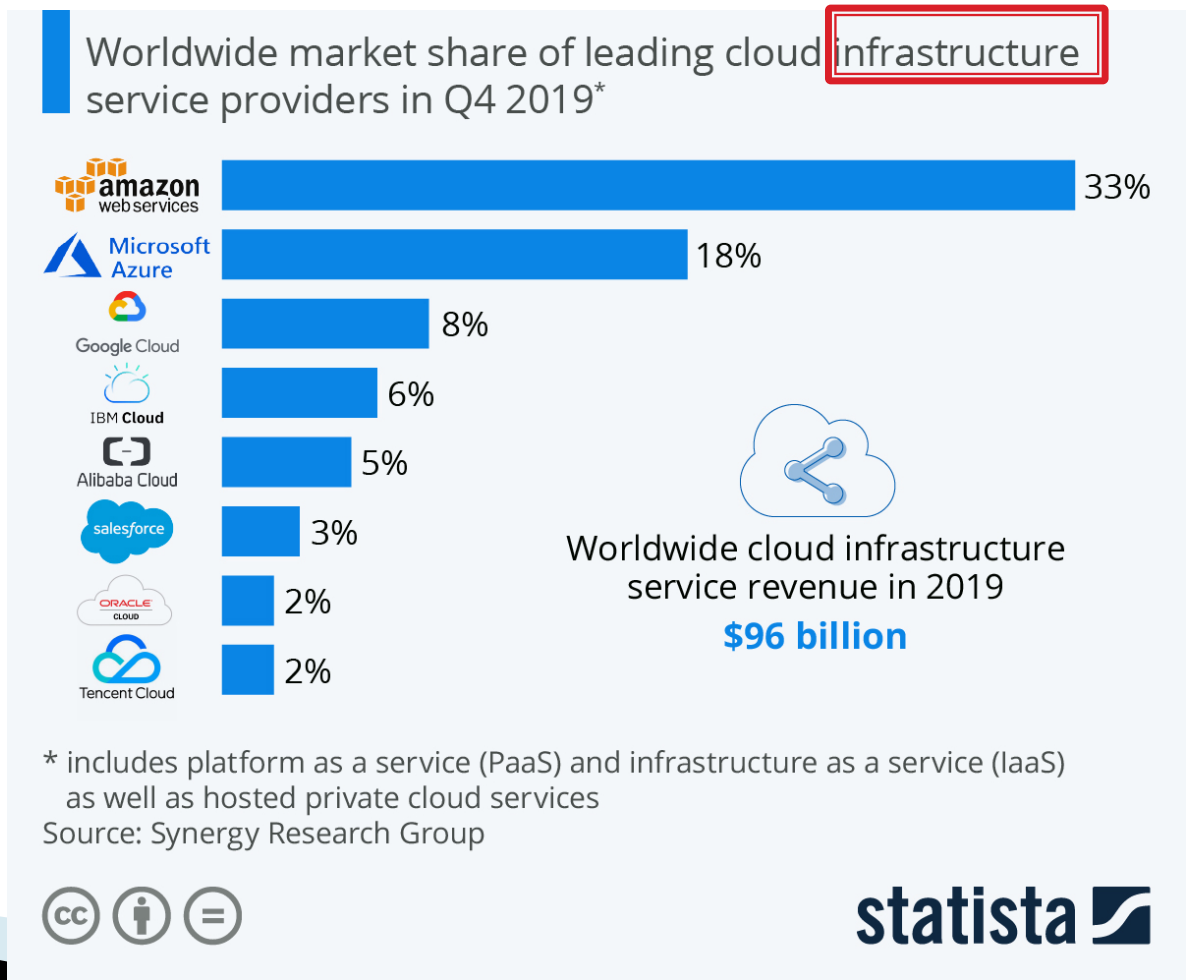
More Layers

- **Containers – CaaS – Container as a Service**
  - Instead of getting full Operating System, get a containerized environment
- **Serverless / FaaS – Function as a Service**
  - Don't even need to deploy an application, just deploy a function



Cloud providers

- Some providers give all levels – IaaS, PaaS, and more, some only part of the levels
- Leaders are slightly different in different levels



AWS IaaS
Class Exercise

Get to know the application

< itc >

plus_one.py:

- reads Pandas DF from file
- transforms it
- writes the result to file

requirements.txt

- Includes pandas package

IaaS Ex – 1 Github

- Create a local Git repo with the following attached files:
  - plus_one.py
  - requirements.txt

- Push this repo to Github **public** repo cloud-class under your Github account

  - ***Note**: You can use **private** Repos, but then you will usually need to put an SSH key on the server to access Github – out of scope for this exercise*

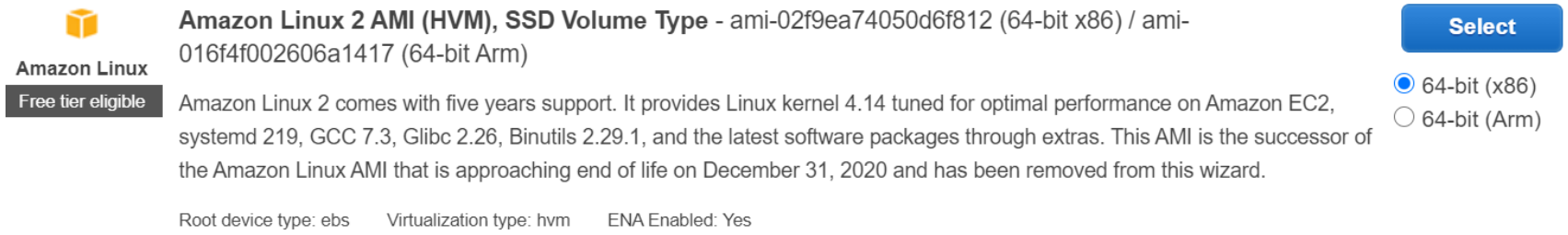
IaaS Ex – 2 AWS signup and setup

< itc >

- Create account in AWS: https://portal.aws.amazon.com/billing/signup
  - There is free tier where you get credits for free usage: https://aws.amazon.com/free
- Sign into AWS console: https://console.aws.amazon.com/
- Choose region Europe (Frankfurt) eu-central-1
- Go to EC2 service → Network & Security → Key pairs
- Create a SSH key pair named my-key-pair
  - Windows / Putty – use ppk file
  - Mac / Linux – use pem file
- Save this file for future use in a secure location (this is the secret)

IaaS Ex – 3 create the server

- Go to “EC2 service” → “Instances” → “Instances”
- Launch a new instance with “Launch Instances”
- Choose an AMI of the default AWS type (usually 1st in the list):



- Choose default options for instance type, and leave everything default
- Review and Launch
- For SSH key, choose the key that you created in step 2

IaaS Ex – 4 connect to the server

< itc >

- Wait for the server to be running
- Copy the DNS of the server in “Instance summary” → “Public IPv4 DNS”
- Connect to the server via SSH from your computer:
  - Default user is on this Server is ec2-user, so host name is ec2-user@DNS, example:
    ec2-user@ec2-35-159-10-38.eu-central-1.compute.amazonaws.com
  - Linux / Mac – via OpenSSH and the PEM file.
    - chmod 400 /path/my-key-pair.pem
    - ssh -i /path/my-key-pair.pem ec2-user@ec2-35-159-10-38.eu-central-1.compute.amazonaws.com
    - Details: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html
  - Windows (Putty) with PPK file:
    - Host name, like: ec2-user@ec2-35-159-10-38.eu-central-1.compute.amazonaws.com
    - Connection → SSH → Auth → Browse to choose the PPK file
    - Save the connection for future use
    - Details: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

For issues: https://docs.aws.amazon.com/console/ec2/instances/connect/docs

IaaS Ex – 5 setup the server

- Install Python 3 (Sometimes Python 2 is installed, and sometimes Python3 is also installed)
  - sudo yum install python3
- Install git: sudo yum install git
- Configure git
  - `git config --global user.email "<emailAddress>"`
  - `git config --global user.name "<gitUserName>"`
- Clone the git repo:
  - git clone HTTPS_URL (get from Github repo → Code → Clone → HTTPS)
- Go inside the cloned directory
- Install modules from requirements.txt:
  - pip3 install --user -r requirements.txt
    - ***Note***: --user *is needed due to permissions restrictions on this Server*

IaaS Ex – 6 Upload, download files, run program

- Upload file 1.csv to repo directory on server from your computer using SCP client
  - Windows – can use program like WinSCP / MobaXterm, details: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html
  - Linux / Mac – using scp:
    - scp -i /path/my-key-pair.pem local_path/1.scv ec2-user@ec2-35-159-10-38.eu-central-1.compute.amazonaws.com:remote_file_path
      - remote_file_path is the path from home directory. So if home directory is /home/ec2-user, to put into /home/ec2-user/cloud-class, remote_file_path is cloud-class
    - Details: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html
- Download file 2.csv from the web to the repo directory on server:

  wget "https://docs.google.com/uc?export=download&id=1gqDsky3SVNSsflMCp0URu_0NwmmRpc6y" -O 2.csv
- Execute plus_one.py
  - python3 plus_one.py 1.csv
  - python3 plus_one.py 2.csv
- Download back to your computer the output files: output1.csv and output2.csv
  - Windows – can use program like WinSCP / MobaXterm
  - Linux / Mac – using scp:
    - scp -i /path/my-key-pair.pem ec2-user@ec2-35-159-10-38.eu-central-1.compute.amazonaws.com:remote_file_path local_file_path

Class exercise

- Give 👍 on Slack that you finished
- Put in Slack print screen of downloading the output files (last exercise step) to your computer

Summary

< itc >

- Cloud computing – what it is
- Pluses and minuses
- Different layers
- Providers
- IaaS – AWS Demo
- Practical example
  - Typical Python application deployment flow in IaaS
  - Security, SSH
  - Class exercise deployment on AWS

Thank you!