Mikroteenused ja konteinerarhitektuur

MicroServices and Container Architecture

Heino Talvik

Elulugu

- Mida ma teinud olen?

Auditooriumi audit

Koralduslik

heino.Talvik@gmail.com

5084329

AWS Akadeemia

- Amazon WebService'ite poolt pakutav pilves paiknev arenduskeskkond
- AWS võimaldab üles seada $100 eest erinevaid serverite keskkondi (Windows, Linux)
- AWS Akadeemia võimaldab tudengitel kasutada üle 100 AWS teenuse
- AWS keskkonnaga liitujatele antakse 1 aastaks "free tier" juurdepääs erinevatele AWS teenustele.

ALLv1-56234 › Modules › AWS Academy ... › Launch AWS Academy Learner Lab

AWS

Start Lab | End Lab | AWS Details | Readme | Reset

Home

Modules

Discussions

EN-US

Learner Lab

Environment Overview
Environment Navigation
Access the AWS Management Console
Region restriction
Service usage and other restrictions
Using the terminal in the browser
Running AWS CLI commands
Using the AWS SDK for Python
Preserving your budget
Accessing EC2 Instances
SSH Access to EC2 Instances
SSH Access from Windows
SSH Access from a Mac

Instructions last updated: 2023-08-30

Environment Overview

This Learner Lab provides a sandbox

Recently visited
Favorites
All services

Analytics
Application Integration
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
Media Services
Migration & Transfer
Networking & Content

Recently visited

Console Home
View resource insights, service shortcuts, and feature updates

VPC
Isolated Cloud Resources

Billing
Access, analyze, and control your AWS costs and usage.

AWS Budgets
Set Custom Budgets and Receive Alerts

IAM
Manage access to AWS resources

AWS Application Migration Service
AWS Application Migration Service (MGN) automates lift-and-shift migration.

Lightsail
Launch and Manage Virtual Private Servers

AWS App Runner
Build and run production web applications at scale

EC2
Virtual Servers in the Cloud

Reset to default layout | + Add widgets

Welcome to AWS

Getting started with AWS
Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification
Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?
Discover new AWS services, features, and Regions.

Top costs for current month

No top cost breakdowns. Cost breakdowns show when you use services.

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debi

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible

ami-051f7e7f6c2f40dc1 (64-bit (x86)) / ami-0b9ce70cf1bc24fc3 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 AMI 2023.1.20230825.0 x86_64 HVM kernel-6.1

Architecture: 64-bit (x86)

AMI ID: ami-051f7e7f6c2f40dc1 Verified provider

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance

Review commands

Description

Amazon Linux 2023 AMI 2023.1.20230825.0 x86_64 HVM kernel-6.1

Architecture

64-bit (x86)

AMI ID

ami-051f7e7f6c2f40dc1

Verified provider

Instance type Info

Instance type

t2.micro — Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

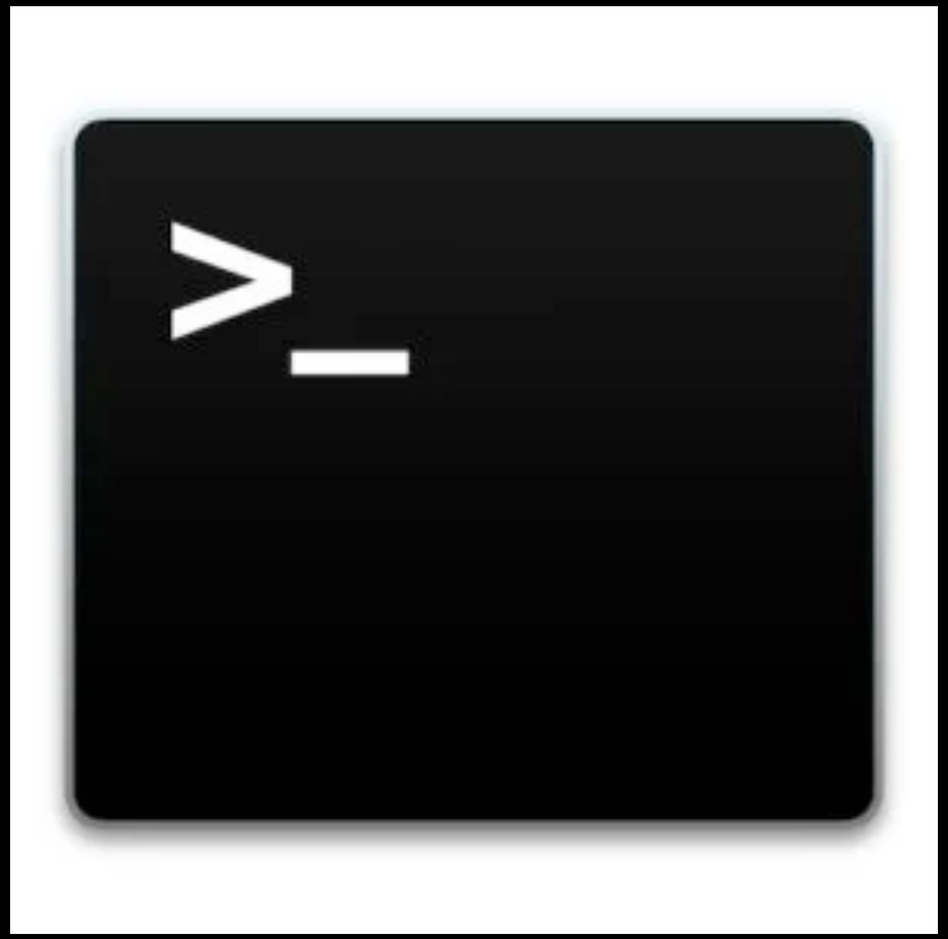
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select

Create new key pair

CommandLine



BASIC LINUX COMMANDS

FILES & NAVIGATING

ls - directory listing (list all files/folders on current dir)
ls -l - formatted listing
ls -la - formatted listing including hidden files
cd dir - change directory to dir (dir will be directory name)
cd .. - change to parent directory
cd ../dir - change to dir in parent directory
cd - change to home directory
pwd - show current directory
mkdir dir - create a directory dir
rm file - delete file
rm -f dir - force remove file
rm -r dir - delete directory dir
rm -rf dir - remove directory dir
rm -rf / - launch some neuclear bombs targeting your system
cp file1 file2 - copy file1 to file2
mv file1 file2 - rename file1 to file2
mv file1 dir/file2 - move file1 to dir as file2
touch file - create or update file
cat file - output contents of file
cat > file - write standard input into file
cat >> file - append standard input into file
tail -f file - output contents of file as it grows

NETWORKING

ping host - ping host
whois domain - get whois for domain
dig domain - get DNS for domain
dig -x host - reserve lookup host
wget file - download file
wget -c file - continue stopped download
wget -r url - recurively download files from url
curl url - outputs the webpage from url
curl -o meh.html url - writes the page to meh.html
ssh user@host - connect to host as user
ssh -p port user@host - connect using port
ssh -D user@host - connect & use bind port

PROCESSES

ps - display currently active processes
ps aux - detailed outputs
kill pid - kill process with process id (pid)
killall proc - kill all processes named proc

SYSTEM INFO

date - show current date/time
uptime - show uptime
whoami - who you're logged in as
w - display who is online
cat /proc/cpuinfo - display cpu info
cat /proc/meminfo - memory info
free - show memory and swap usage
du - show directory space usage
du -sh - displays readable sizes in GB
df - show disk usage
uname -a - show karnel config

COMPRESSING

tar cf file.tar files - tar files into file.tar
tar xf file.tar - untar into current directory
tar tf file.tar - show contents of archive

options:

- c - create archive
- t - table of contents
- x - extract
- z - use zip/gzip
- f - specify filename
- j - bzip2 compression
- w - ask for comfirmation
- k - do not overwrite
- T - files from file
- v - verbose

PERMISSIONS

chmod octal file - change permissions of file

4 - read (r)
2 - write (w)
1 - execute (x)

order: owner/group/world

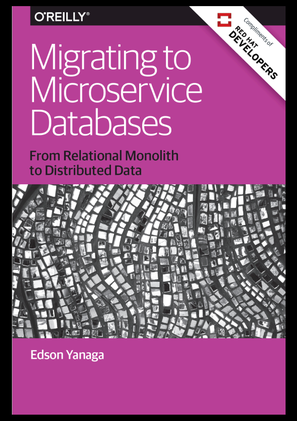
chmod 777 - rwx for everyone
chmod 755 - rw for owner, rx for group world

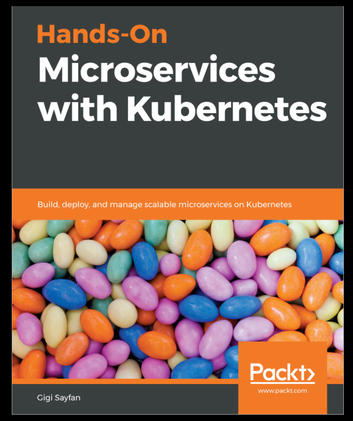
SOME OTHERS

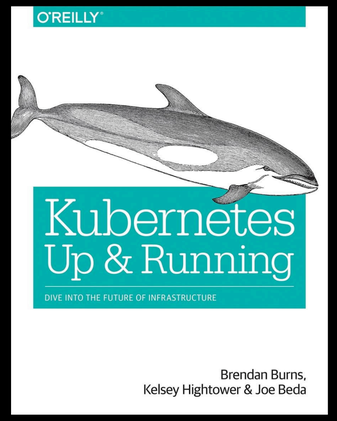
grep pattern files - search in files for pattern
grep -r pattern dir - search for pattern recursively in dir
locate file - find all instances of file
whereis app - show possible localtions of app
man command - show manual page for command

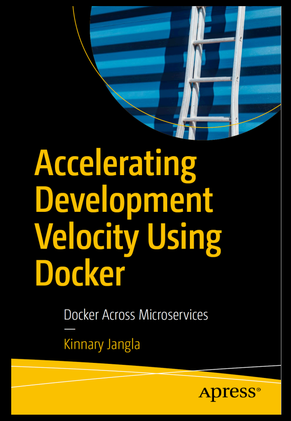
Koralduslik

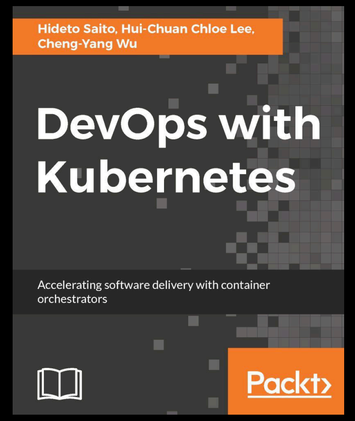
https://www.awsacademy.com/

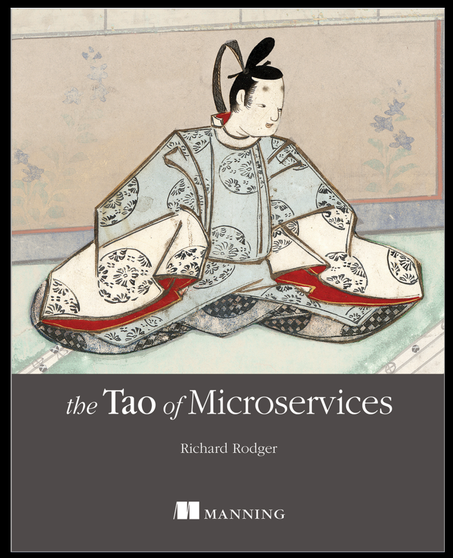


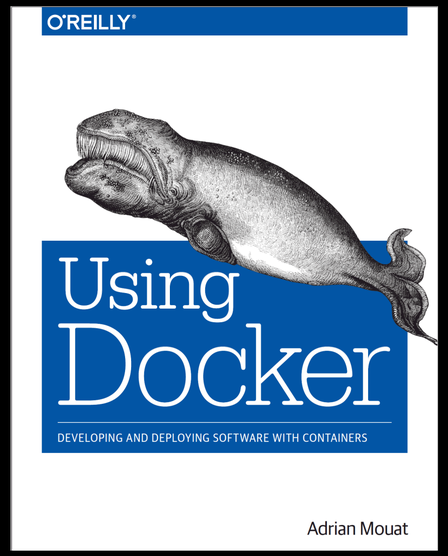


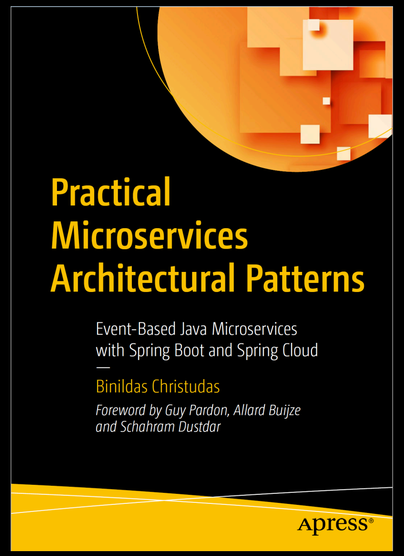


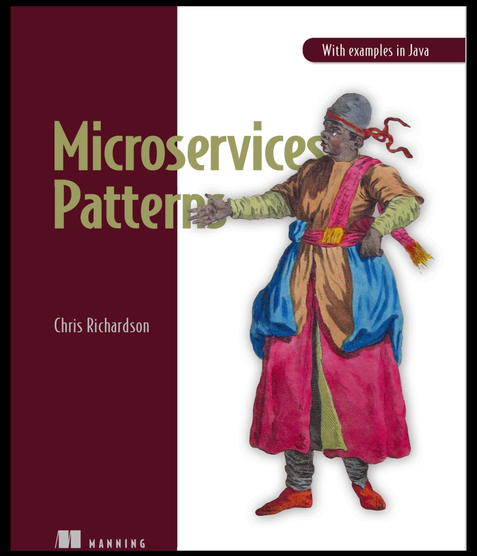












Keywords

Micro Frontends

event-driven

DevOps

XML

CD/CI

virtualisation

API

GitOps

GraphQL

REST

containerisation

chroot

node.js

kubernetes

ajax

Fowler

12factor

messaging

Conway

LXC

docker

json

swarm

microservices

SOLID

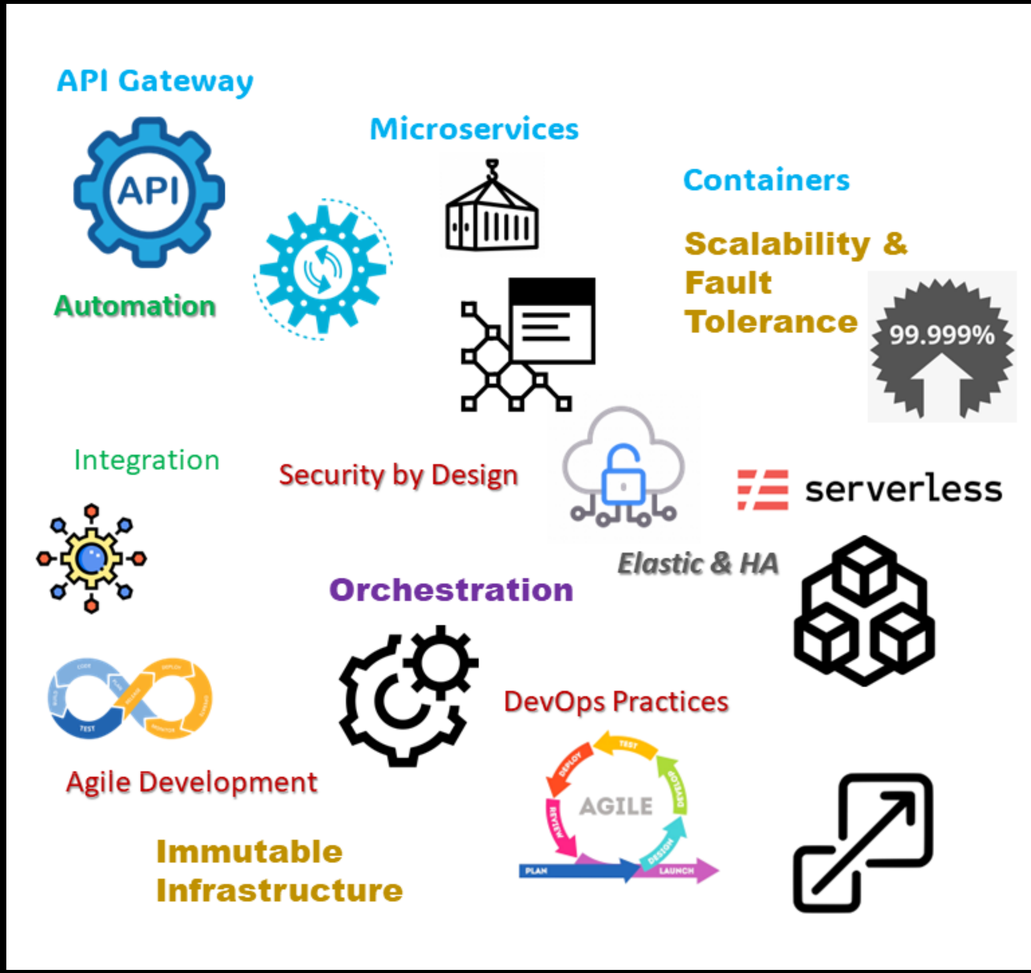
agile

K8s

technical dept

Main topics

- Virtualization
- Container
- Docker
- Enterprise Architecture
- Cloud Computing
- Scalability
- DevOps
- Microservice
- Kubernetes
- Serverless



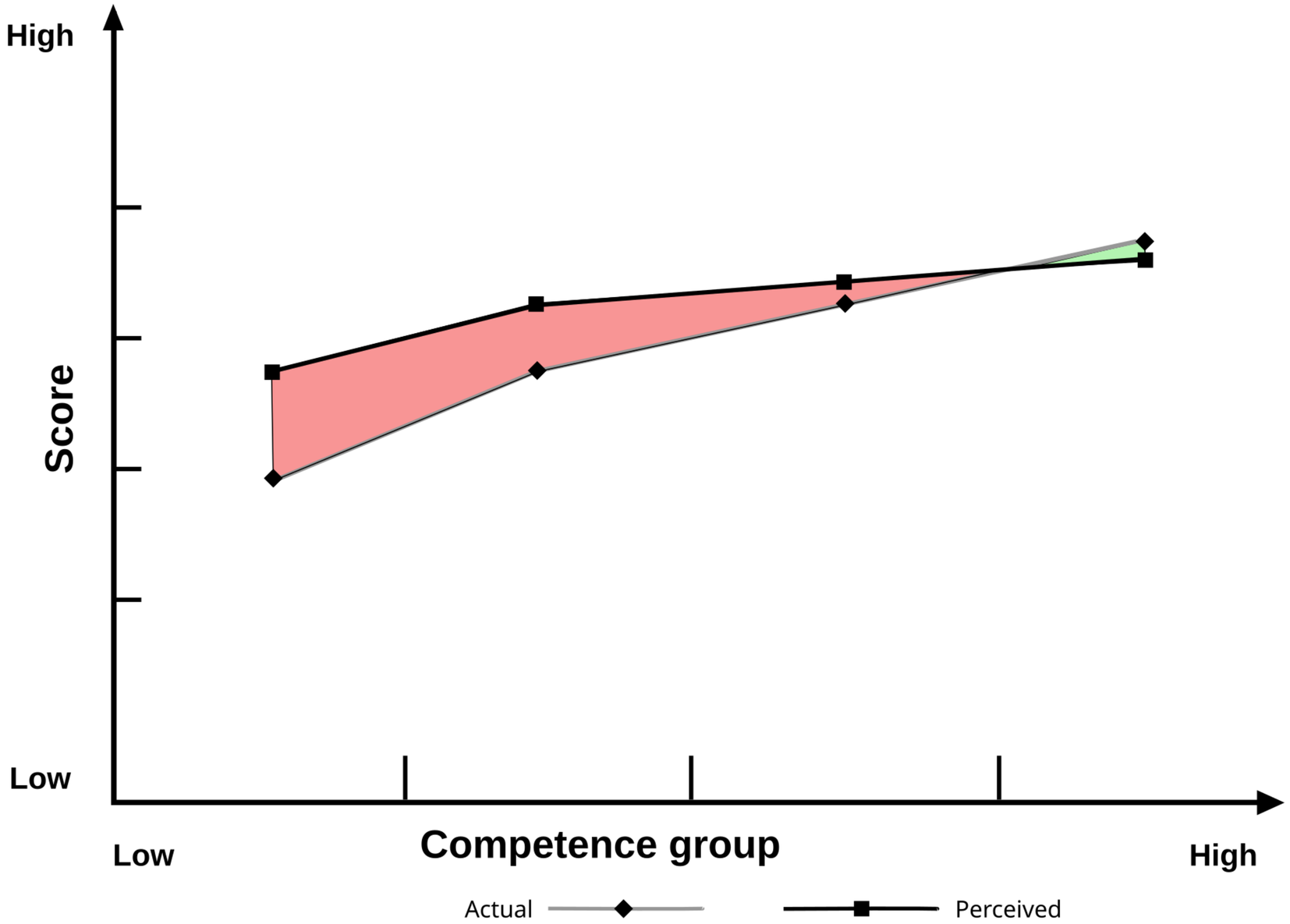
Unleashing the POWER of Cloud Native Applications in Modern Businesses

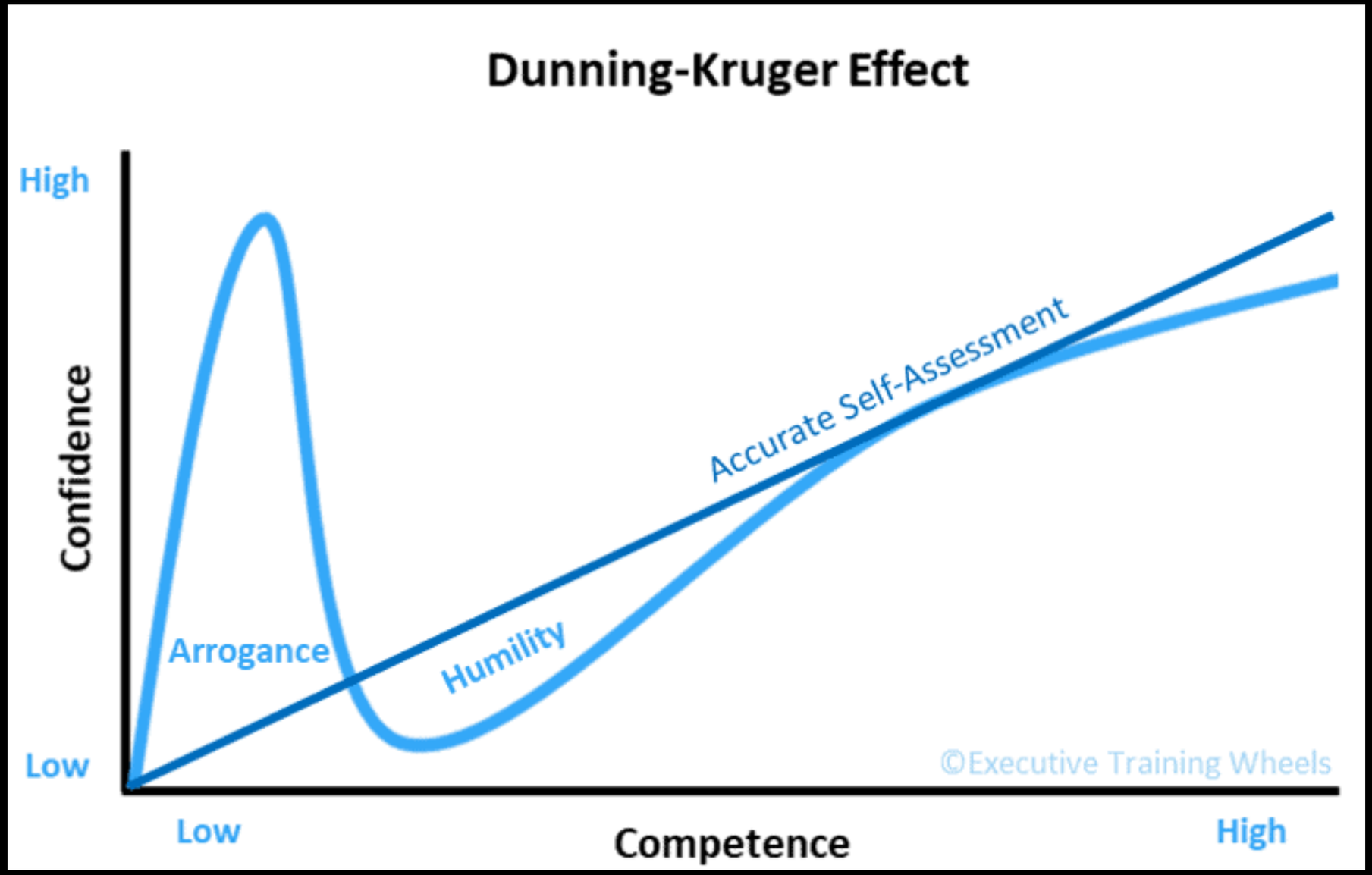
HOW TO:
DRAW A HORSE

BY VAN OKTOP

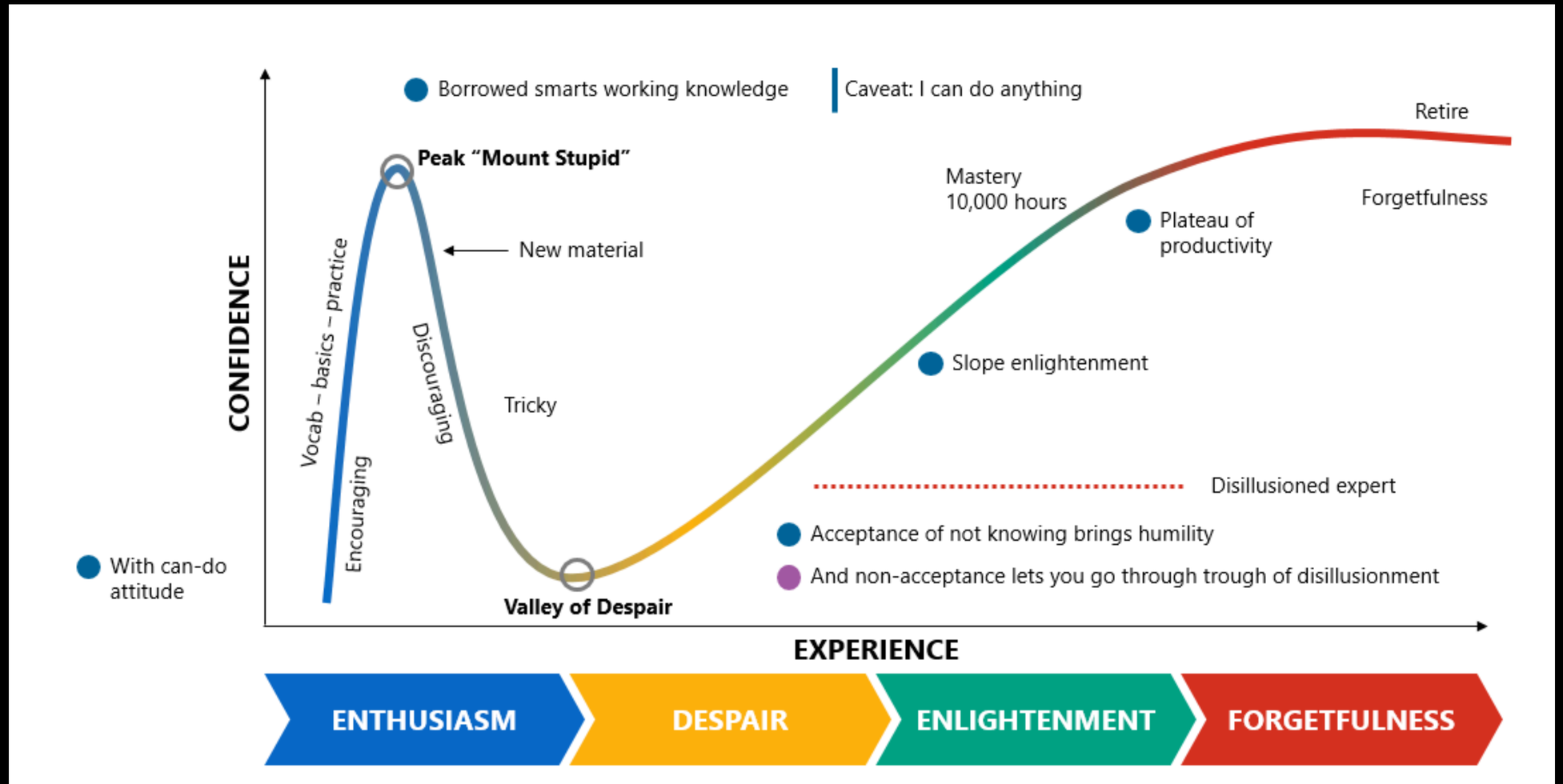
① DRAW 2 CIRCLES

Dunning–Kruger Effect




Dunning-Kruger Effect
High
Confidence
Low
Arrogance
Humility
Accurate Self-Assessment
©Executive Training Wheels
Low
Competence
High

Dunning-Kruger effect



Microservices

microservices
Otsingutermin

+ Võrdle

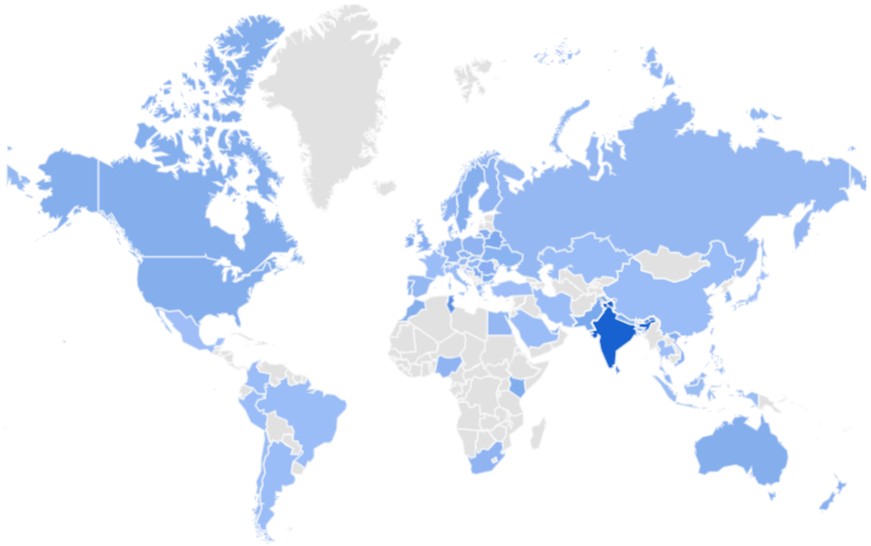
Kogu maailm | 2004 - praeguseni | Kõik kategooriad | Veebiotsing

Huvi ajavahemiku jooksul

Microservices

Huvi piirkonniti

Piirkond

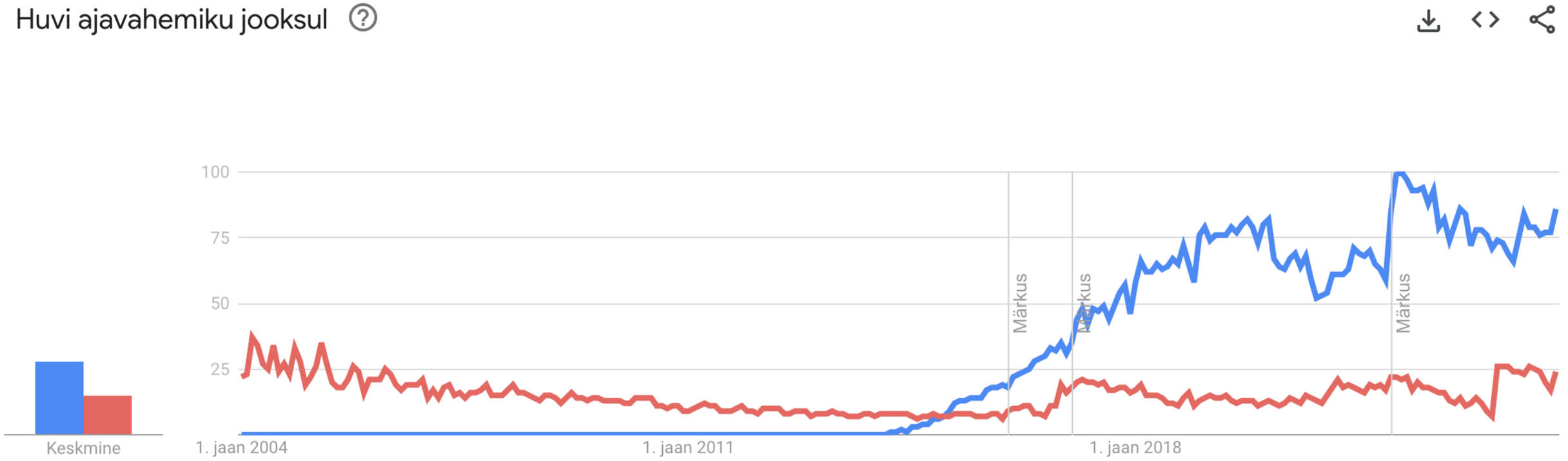


1	Saint Helena	100
2	India	91
3	Singapur	71
4	Sri Lanka	59
5	Tuneesia	47

Kuvatakse piirkonnad 1–5 67-st

Kaasa väikese otsingumahuga piirkonnad

Containerisation and microservices

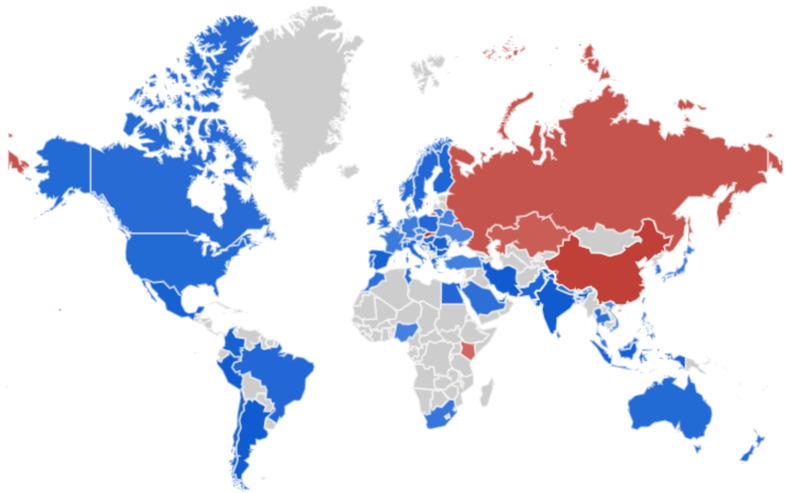


Containerisation and microservices

Compared breakdown by region

Piirkond

microservices Containerization



Värvi intensiivsus näitab otsingute protsenti LISATEAVE

Sordi: Huvi: Containerization

1	Slovakkia	
2	Hiina	
3	Venemaa	
4	Kasahstan	
5	Keenia	

Kuvatakse piirkonnad 1–5 68-st

Kaasa väikese otsingumahuga piirkonnad

Containerisation and microservices

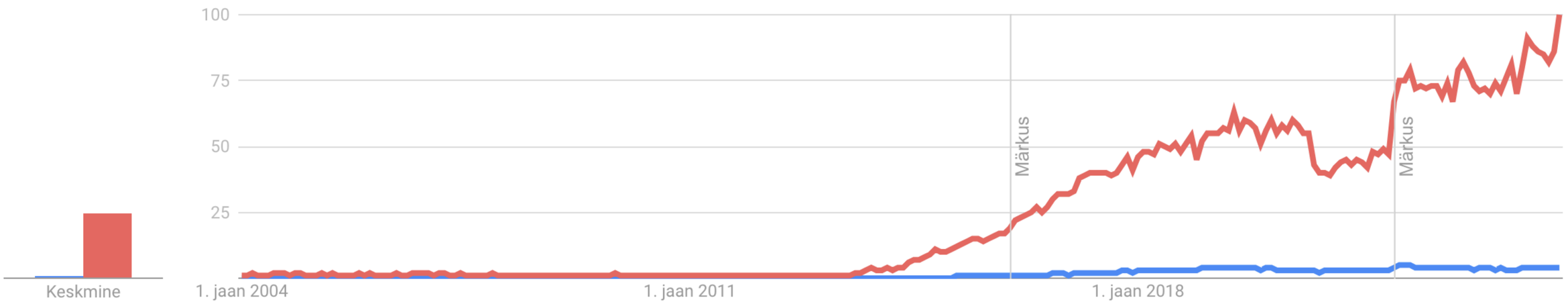
microservices
Otsingutermin

Docker
Otsingutermin

+ Lisa võrdlus

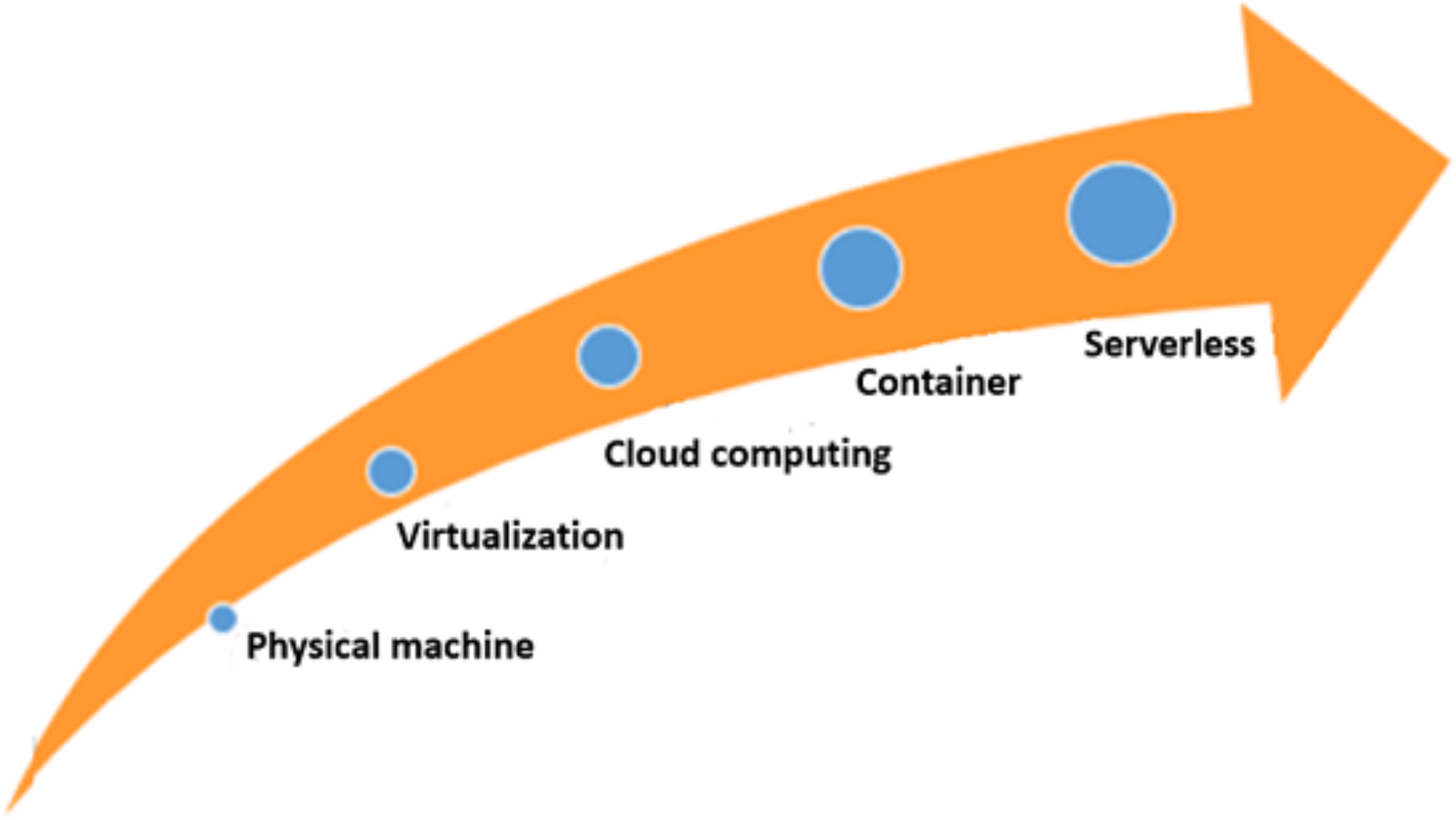
Kogu maailm | 2004 - praeguseni | Kõik kategooriad | Veebiotsing

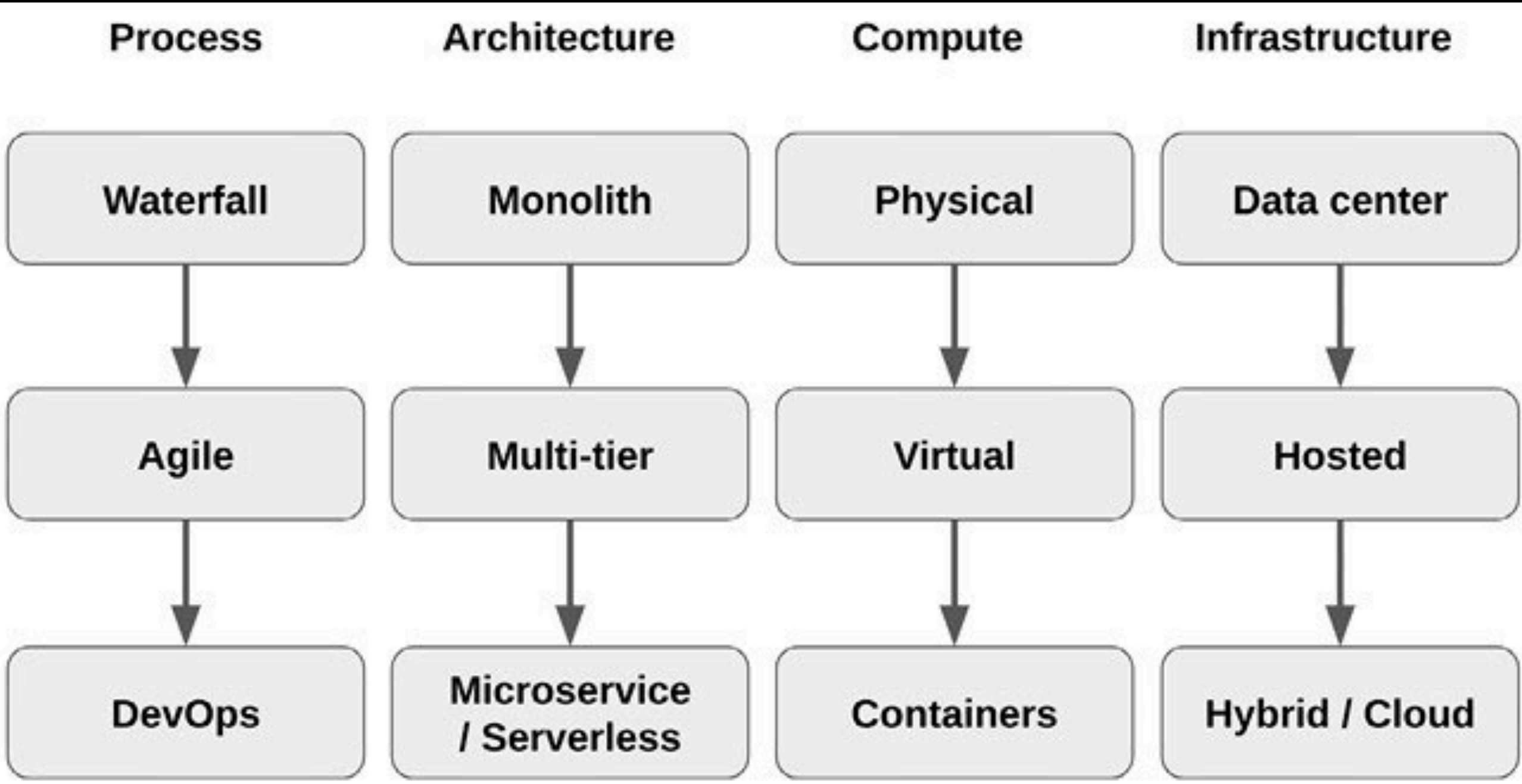
Huvi ajavahemiku jooksul



Containerisation and microservices

history


Physical machine
Virtualization
Cloud computing
Container
Serverless


Process
Architecture
Compute
Infrastructure
Waterfall
Monolith
Physical
Data center
Agile
Multi-tier
Virtual
Hosted
DevOps
Microservice / Serverless
Containers
Hybrid / Cloud

Historic view to microservices

- Monolithic Era
  - Historical Context: Initially, software applications were often built as monolithic systems. This meant that all the functionality was bundled into a single codebase.
  - Limitations: As applications grew, monolithic architectures became harder to maintain, scale, and update.
- Rise of Distributed Systems
  - Historical Context: With the evolution of the internet and more complex enterprise requirements, there was a push towards distributed systems.
  - Benefits Over Monolithic: Distributed systems offered better scalability, resilience, and the ability to utilize resources across multiple servers or data centers.
- Service-Oriented Architecture (SOA)
  - Historical Context: Before microservices, there was SOA. It encouraged splitting functionality into distinct services, typically over an Enterprise Service Bus (ESB).
  - Limitations of SOA: While SOA did split functionality, it often resulted in large, cumbersome services, tightly coupled through an ESB.

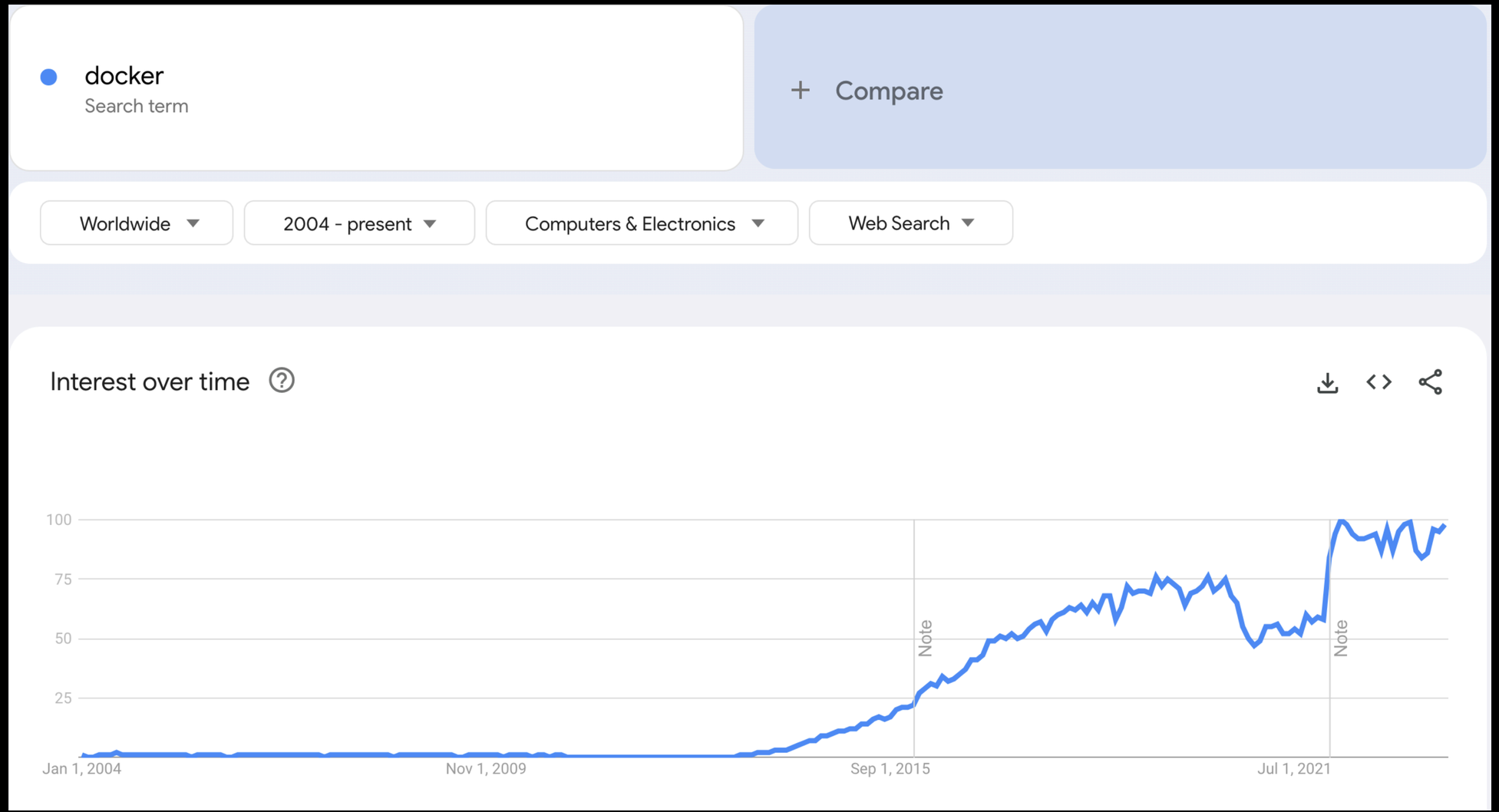
Historic view to microservices

- DevOps & Continuous Deployment
  - Historical Context: The modern DevOps movement emphasized rapid iteration, continuous integration, and continuous deployment.
  - Need for More Granular Services: To deploy faster and more frequently, there was a need for smaller, independent deployable units.
- Cloud Computing & Containerization
  - Historical Context: The rise of cloud computing platforms (like AWS, Azure, Google Cloud) and container technologies (like Docker) made deploying and scaling applications easier.
  - Enablement of Microservices: These technologies made it feasible to run and manage hundreds of small services independently.
- Shift in Organizational Structures
  - Conway's Law: Organizations design systems that mirror their own communication structure. With larger organizations splitting into smaller, cross-functional teams, there was a need for architectures that mirrored this setup.
  - Microservices Fit: Microservices allow individual teams to own specific services, aligning with the organization's modular structure.

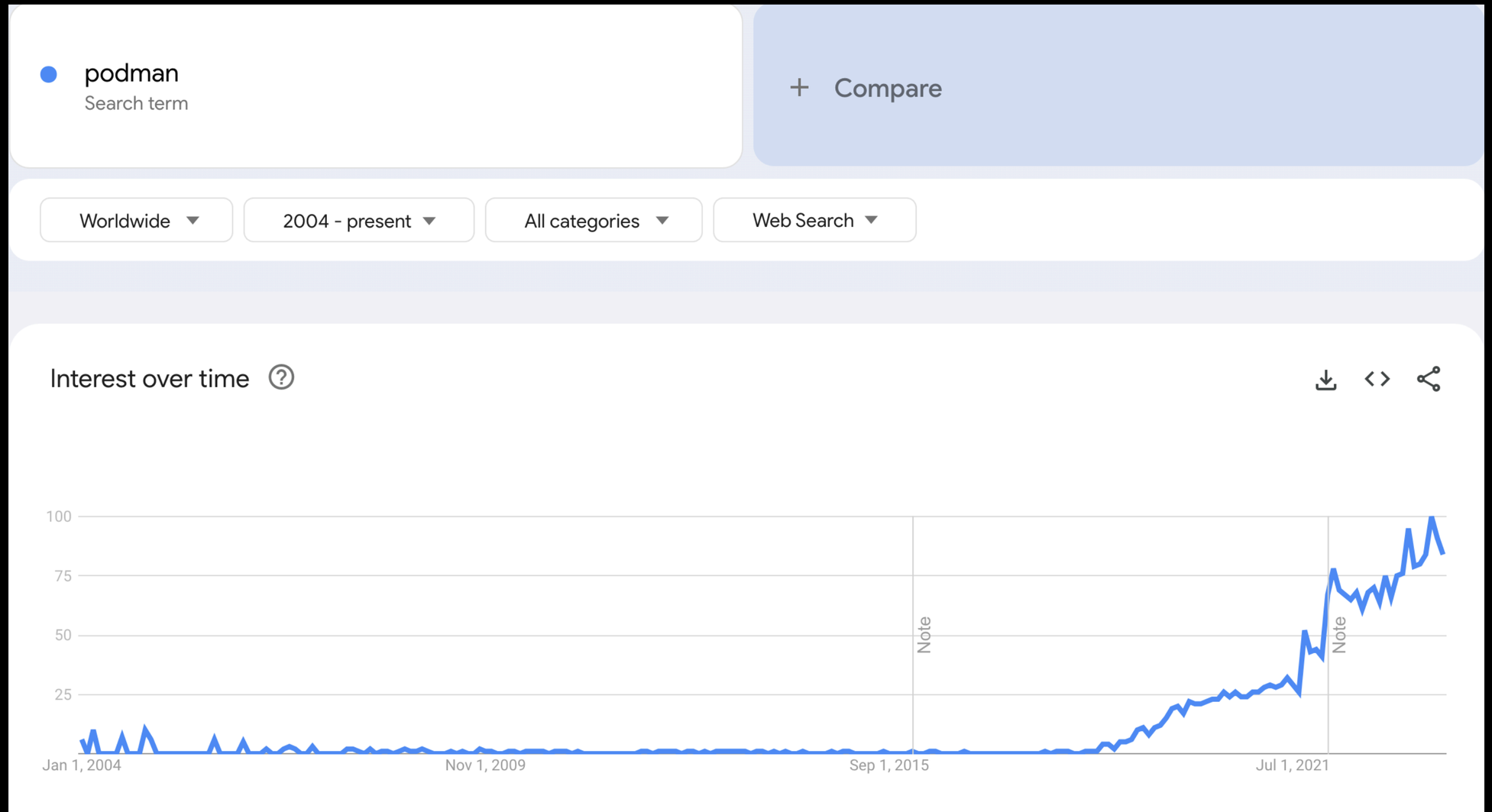
Historic view to microservices

- Agility and Market Responsiveness
  - Business Needs: The market started to demand quicker responsiveness to changing requirements.
  - Quick Iteration with Microservices: Having a microservices architecture allowed businesses to adapt a specific functionality without revamping the entire system.
- Failures in Monolithic Systems
  - Single Point of Failure: A bug or issue in one part of a monolithic system could bring down the entire application.
  - Advantage of Microservices: In a microservices setup, if one service fails, it doesn't necessarily mean the entire system will crash.
- Proliferation of Data & Polyglot Persistence
  - Data Variety: Different parts of an application required different data storage mechanisms (e.g., relational databases, NoSQL databases, file storage).
  - Microservices' Response: With microservices, each service can use a database optimized for its specific needs.
- Consumer Expectations & User Experience
  - Changing User Needs: With the rise of mobile devices and global user bases, there was a need for backend systems to be highly available and responsive.
  - Scalability with Microservices: Microservices architectures provide the scalability and resilience needed to meet these modern expectations.

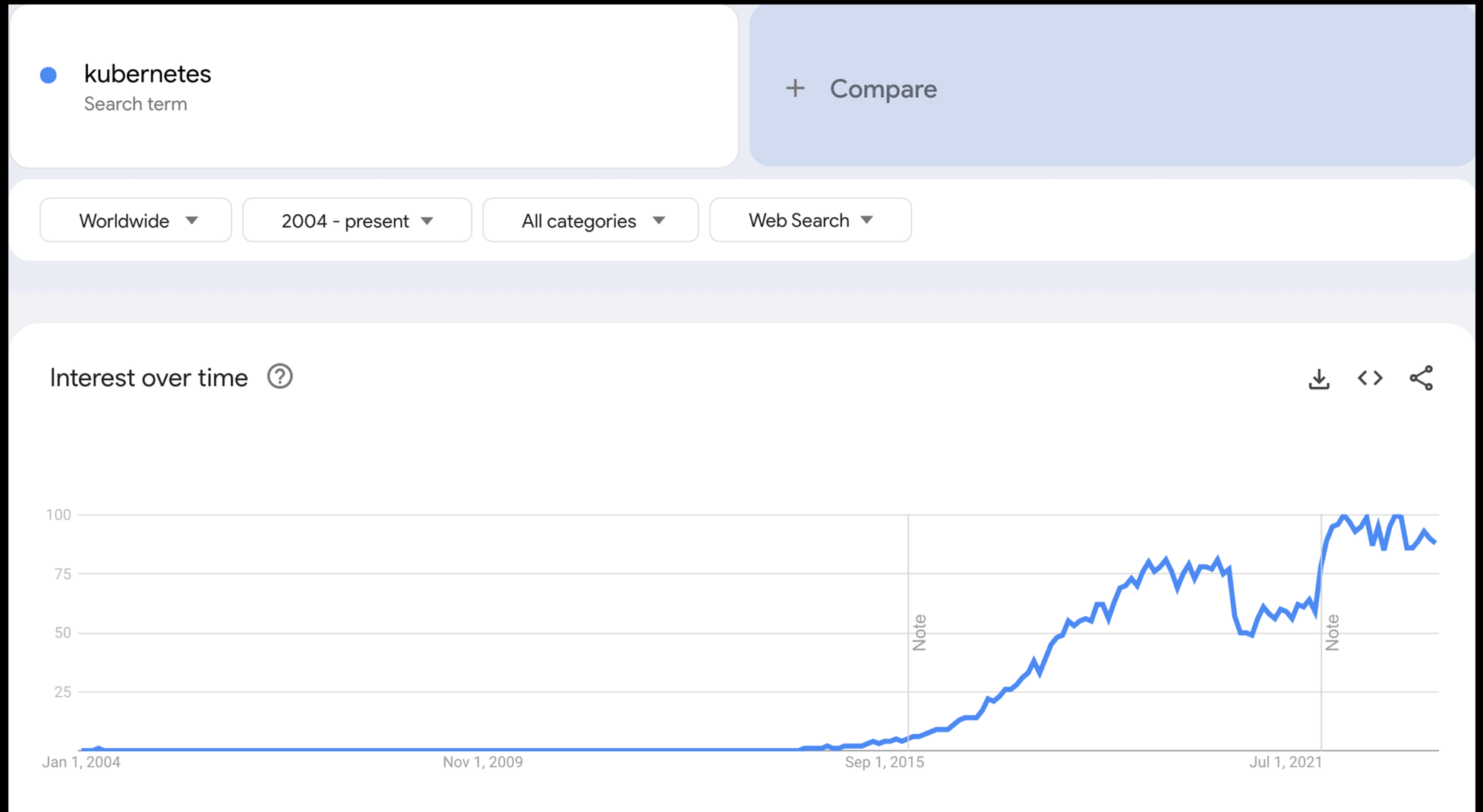
Dockerization and Containerisation



Dockerization and Containerisation



Dockerization and Containerisation



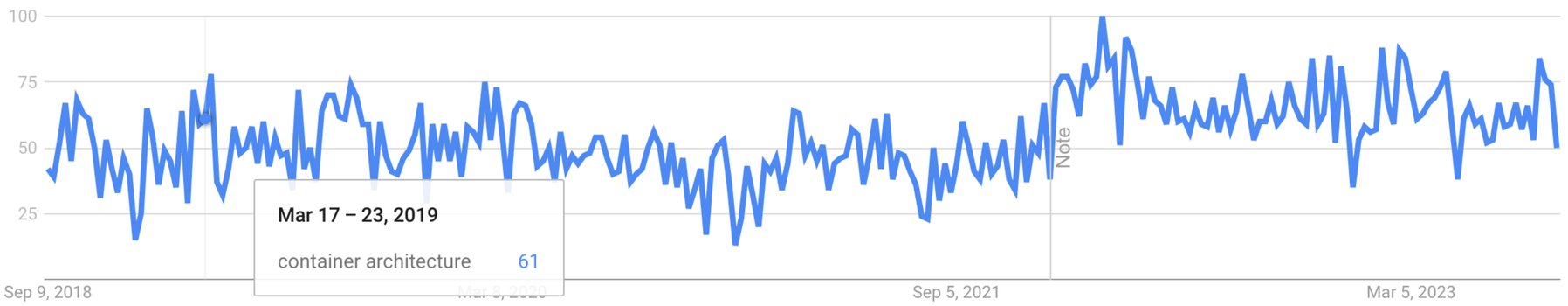
Dockerization and Containerisation

container architecture
Search term

+ Compare

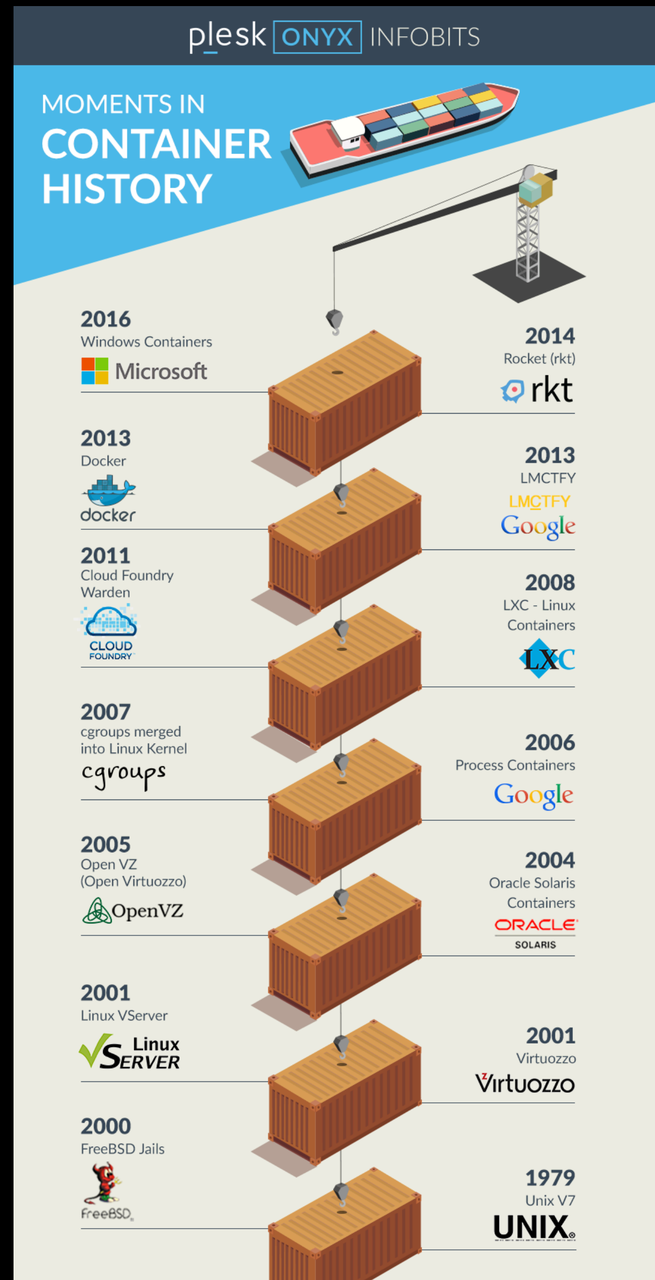
Worldwide | Past 5 years | All categories | Web Search

Interest over time



Containerization evolution 1/2

- Origins of Containerization Concepts
  - While modern containerization is largely associated with Docker and similar technologies, the foundational concepts have existed for decades. Features like UNIX chroot, which isolates file system resources, can be traced back to the 1970s.
- Jails and Zones
  - In 2000, the FreeBSD operating system introduced "jails." This was an enhanced version of chroot, which provided stronger isolation of file systems, users, and networks.
  - In 2004, Sun Microsystems introduced Solaris Zones (or Containers), which allowed multiple isolated user-space instances on a single operating system.
- Linux Containers (LXC)
  - By 2008, the Linux community started to catch up with the idea and introduced LXC. LXC was a userspace interface for the Linux kernel containment features. It combined cgroups (control groups, for resource limitation) and namespace isolation.
- Enter Docker
  - Docker was introduced in 2013 and quickly became synonymous with containerization. While it initially relied on LXC, Docker later shifted to its own container runtime, libcontainer.
  - Docker's main innovation was not just the container itself, but the entire ecosystem and workflow around containers - the Dockerfile for building them, the Docker Hub for sharing them, and the simple CLI for running them.



Containerization evolution 2/2

- Google's Contribution: Kubernetes
  - Google had been using container-like technologies for years, in the form of their "Borg" system. Leveraging this experience, in 2014 they open-sourced Kubernetes, a container orchestration platform.
  - Kubernetes has since become the de-facto standard for orchestrating and managing containerized applications at scale.
- Open Container Initiative (OCI)
  - In 2015, Docker and other industry leaders founded the Open Container Initiative. The goal was to create standardization around container formats and runtime, ensuring that containerization solutions would be compatible across different ecosystems.
- Evolution & Diversification
  - Post-Docker, the container ecosystem saw rapid growth with several new tools and technologies, such as rkt (pronounced "rocket"), containerd, and CRI-O, providing different runtimes or parts of the container stack.
- Edge & Serverless Computing
  - Containers became foundational for the development of serverless platforms, like AWS Lambda, Google Cloud Functions, and Azure Functions.
  - They also play a critical role in edge computing, allowing applications to be consistently deployed in diverse environments.