

# Introduction & Motivation

Lecture: "Deploying Containerized Application to the Cloud in Practice"

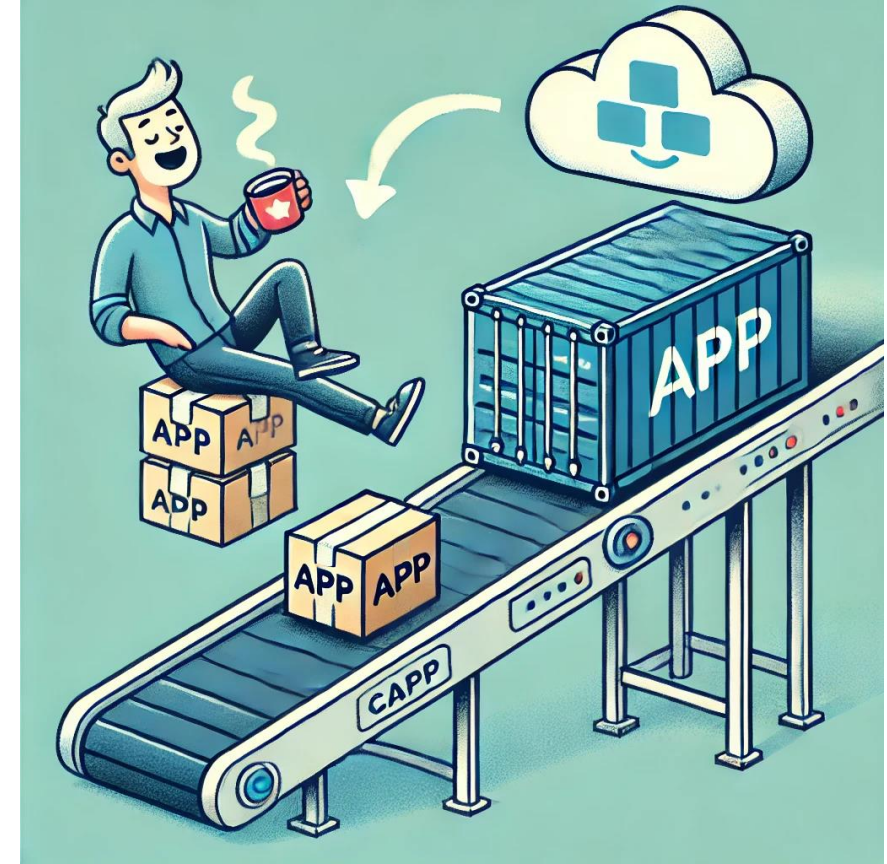
# Why Containerized Cloud Deployments? (1/2)

- The "traditional" deployment has... challenges.
  - Deployment inconsistencies
  - Manual setup
  - Scaling issues
  - Resource waste



# Why Containerized Cloud Deployments? (2/2)

- Thus, by utilizing containers and cloud(s), the goal is to achieve:
  - Portability
  - Automation
  - Scalability
  - Cost-effectiveness



# References and More to Read

- M. Narasimhulu, et al., "Investigating the Impact of Containerization on the Deployment Process in DevOps," 2023 2nd International Conference on Edge Computing and Applications (ICECAA), Namakkal, India, 2023, pp. 679-685, doi: 10.1109/ICECAA58104.2023.10212240. <https://ieeexplore.ieee.org/document/10212240>
- Microsoft, "Get started with cloud native apps and containerized deployments", <https://learn.microsoft.com/en-us/training/modules/get-started-cloud-native-apps-containerized-deployments/>
- Google Cloud, "Best practices for continuous integration and delivery to Google Kubernetes Engine", <https://cloud.google.com/kubernetes-engine/docs/concepts/best-practices-continuous-integration-delivery-kubernetes>

# **Automating Deployment (CI/CD)**

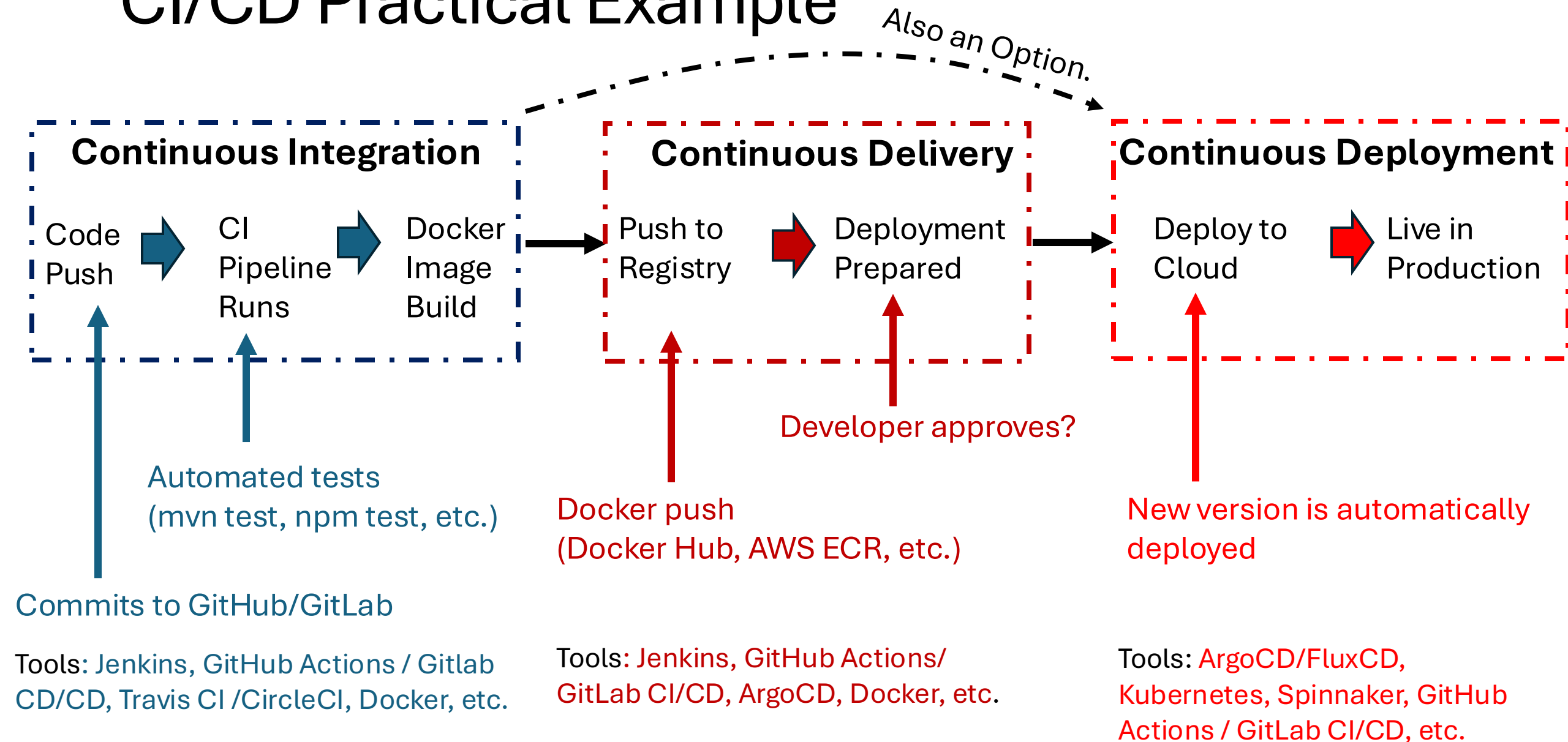
Lecture: "Deploying Containerized Application to the Cloud in Practice"

# What is CI/CD?

- CI/CD, Continuous Integration ( / Continuous Delivery) / Continuous Development, aims to automate the process of building, testing and deploying software



# CI/CD Practical Example





# References and More to Read

- Red Hat, "What is CI/CD?",  
<https://www.redhat.com/en/topics/devops/what-is-ci-cd>
- Martin Fowler, "Continuous Integration",  
<https://martinfowler.com/articles/continuousIntegration.html>
- **Visual Studio Code, "Developing inside a Container"**,  
<https://code.visualstudio.com/docs/devcontainers/containers>
- da Gíã, Hugo, et al. "Chronicles of CI/CD: A Deep Dive into its Usage Over Time." arXiv preprint arXiv:2402.17588 (2024).  
<https://doi.org/10.48550/arXiv.2402.17588>
- **Simple CI/CD pipeline to try-out**
  - GitHub Docs, "Quickstart for GitHub Actions", <https://learn.microsoft.com/en-us/azure/container-instances/container-instances-quickstart>



# **Containerizing an Application**

Lecture: "Deploying Containerized Application to the Cloud in Practice"

# Options for Cloud Deployment

	Scalability	Control	Cost-efficiency	Ease of Use	Performance	Use Cases	Examples
<b>Serverless</b>	High, automatic	Limited	Pay-per-use	Less operational management	Variable, possible cold starts	Event-driven applications, IoT, real-time data, unpredictable/fluctuating workloads	AWS Lambda, Azure Functions, Google Cloud Functions
<b>Platform-as-a-Service (PaaS)</b>	Moderate to high	Moderate	Generally cost-effective / pay-per-use	Simplified deployment	Depends on the service (generally good)	Rapid development, small to medium-sized applications	AWS Elastic Beanstalk, Azure App Service, Google App Engine
<b>Containers</b>	High, manual	Full	Steady cost, more for reserved resources	Orchestration required for complex applications	High	Complex, scalable applications, when environmental control is required, multi-cloud/hybrid environments	Docker, Kubernetes (also on, e.g., AWS, Azure)

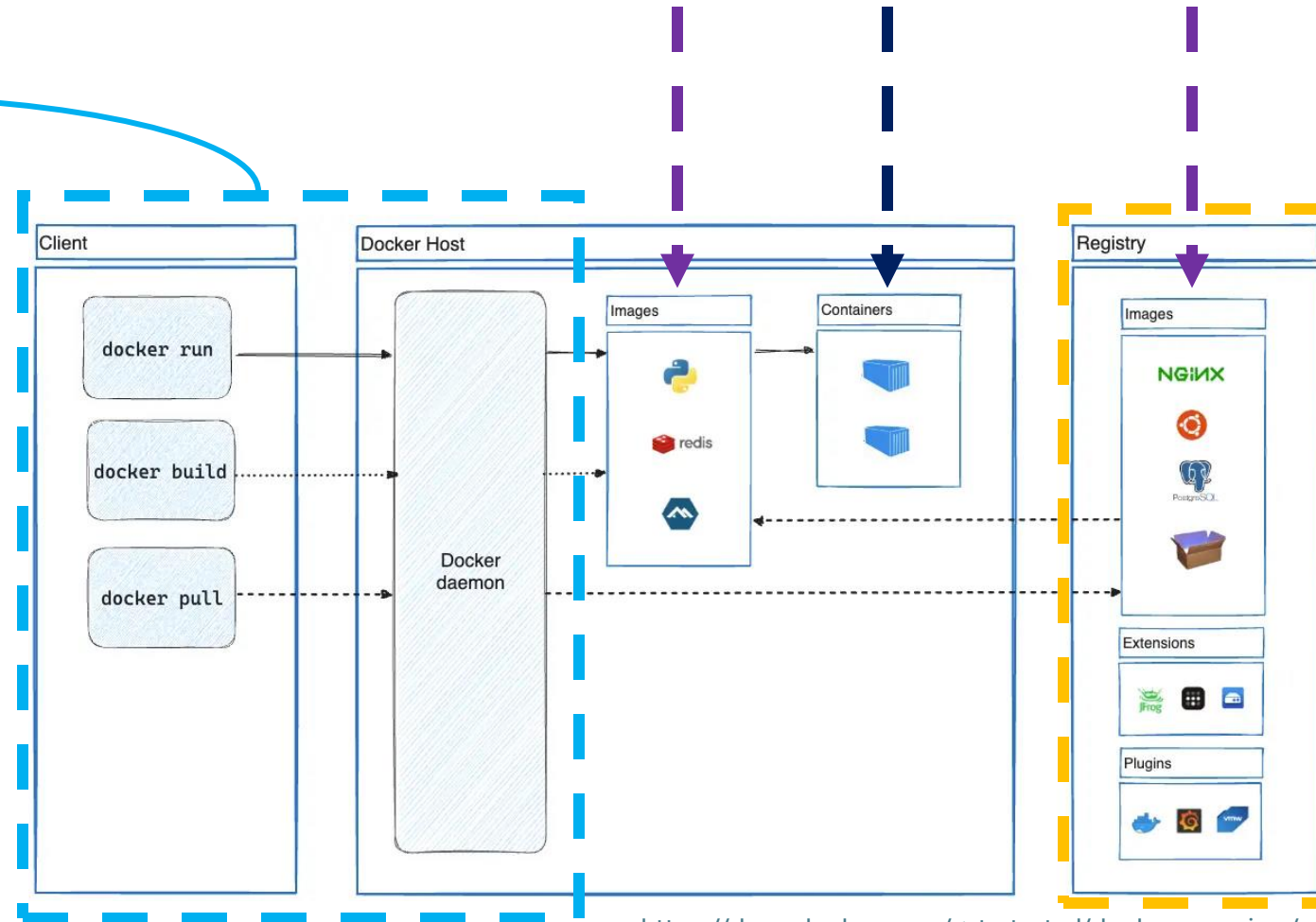
Adapted from Alok Mishra Blog, Journal of Distributed Software Engineering, Architecture and Design:  
<https://alok-mishra.com/2024/01/02/choosing-the-right-architecture-comparing-serverless-containers-and-platform-as-a-service-paas-for-microservices-applications/>

# Docker – the Basics

- Docker is a platform for developing, shipping, and running applications in lightweight, portable containers.

- Key Components

- **Docker Engine**, core runtime for building and running containers.
- **Docker Image**, a snapshot containing the application, dependencies and OS environment.
- **Dockerfile**, a script defining how a Docker image is built.
- **Docker Container**, a running instance of a Docker image.
- **Docker Hub / Registry**, a repository for storing and sharing images.



# Docker

## Pros

- Portability
  - Local, cloud, on-premises
- Isolation
  - Generally, no dependency conflicts with the host system
- Scalability
- Efficiency
  - Better than, e.g. VMs


## Cons

- Learning curve
  - Understanding images, networking, volumes
- Performance overhead
  - Slightly slower than native
- Security concerns
  - Not as well isolated as VMs

# Docker – in Practice

## Example with Quarkus

1. Install requirements
  1. Docker, Maven, Java 21+
2. Create example application
  1. `mvn io.quarkus.platform:quarkus-maven-plugin:3.18.2:create -DprojectGroupId=com.example -DprojectArtifactId=example-app -Dextensions="resteasy"`
3. Test your application works
  1. `cd example-app && mvn quarkus:dev`
  2. Web browser: <http://localhost:8080/hello>
4. Compile native binaries
  1. `mvn package -Pnative` (downloads images from Docker, docker/root (user) permissions required)
5. Create Docker image
  1. Create file named "**Dockerfile**" in the application root file:
  2. `docker build -t example-app .`
6. Done!
  1. `docker run -p 8080:8080 questions-app`
  2. Web browser: <http://localhost:8080/hello>



```
FROM quay.io/quarkus/quarkus-distrolless-image:2.0
WORKDIR /app
COPY target/*-runner /app/app
CMD ["/app/app"]
EXPOSE 8080
```

# References and More to Read

- Platforms/tools for container management (orchestration)
  - Docker, "What is Docker?", <https://docs.docker.com/get-started/docker-overview/>
  - Kubernetes, "Why you need Kubernetes and what it can do", <https://kubernetes.io/docs/concepts/overview/>
- Information Age, "Kubernetes vs Docker - pros and cons", <https://www.information-age.com/kubernetes-vs-docker-pros-and-cons-123501812>
- B. Rad, et al. "An Introduction to Docker and Analysis of its Performance.", IJCSNS International Journal of Computer Science and Network Security, 2017, vol. 17, n. 3, p. 8., [https://www.researchgate.net/publication/318816158\\_An\\_Introduction\\_to\\_Docker\\_and\\_Analysis\\_of\\_its\\_Performance](https://www.researchgate.net/publication/318816158_An_Introduction_to_Docker_and_Analysis_of_its_Performance)
- Alok Mishra, Journal of Distributed Software Engineering, Architecture and Design, "Choosing the Right Architecture: Comparing Serverless, Containers, and Platform-as-a-Service (PaaS) for Microservices Applications", <https://alok-mishra.com/2024/01/02/choosing-the-right-architecture-comparing-serverless-containers-and-platform-as-a-service-paas-for-microservices-applications/>
- DigitalOcean, "Serverless vs Containers: Which is best for your needs?", <https://www.digitalocean.com/resources/articles/serverless-vs-containers>

# Deploying to the Cloud



Lecture: "Deploying Containerized Application to the Cloud in Practice"



# References and More to Read

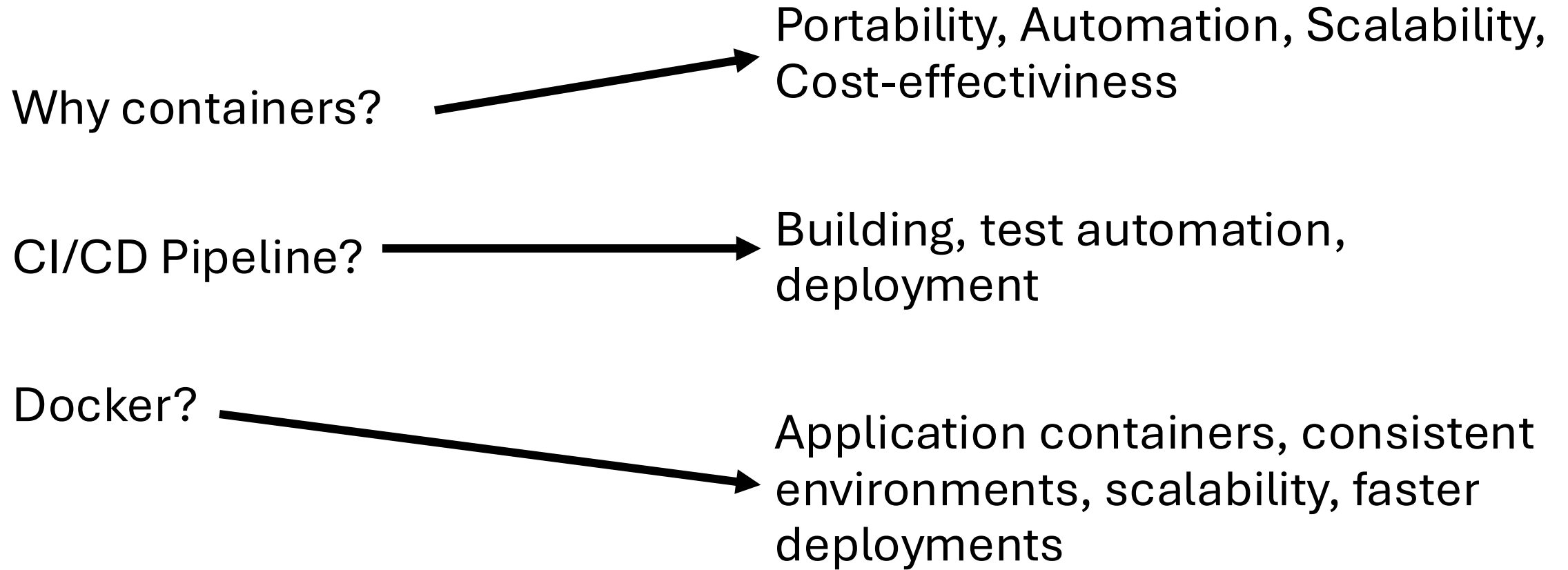
- dockerdocs, "Build and push your first image",  
<https://docs.docker.com/get-started/introduction/build-and-push-first-image/>
- **Cloud deployment:**
  - Amazon AWS, "Deploy Docker Containers on Amazon ECS",  
<https://aws.amazon.com/getting-started/hands-on/deploy-docker-containers/>
  - Microsoft, "Quickstart: Deploy a container instance in Azure using the Azure CLI", <https://learn.microsoft.com/en-us/azure/container-instances/container-instances-quickstart>

# **Key Takeaways**

## **(summary)**

Lecture: "Deploying Containerized Application to the Cloud in Practice"

# Summary



And, a bonus: Consider multi-stage builds, scan for vulnerabilities, test your tests, and secure your secrets!