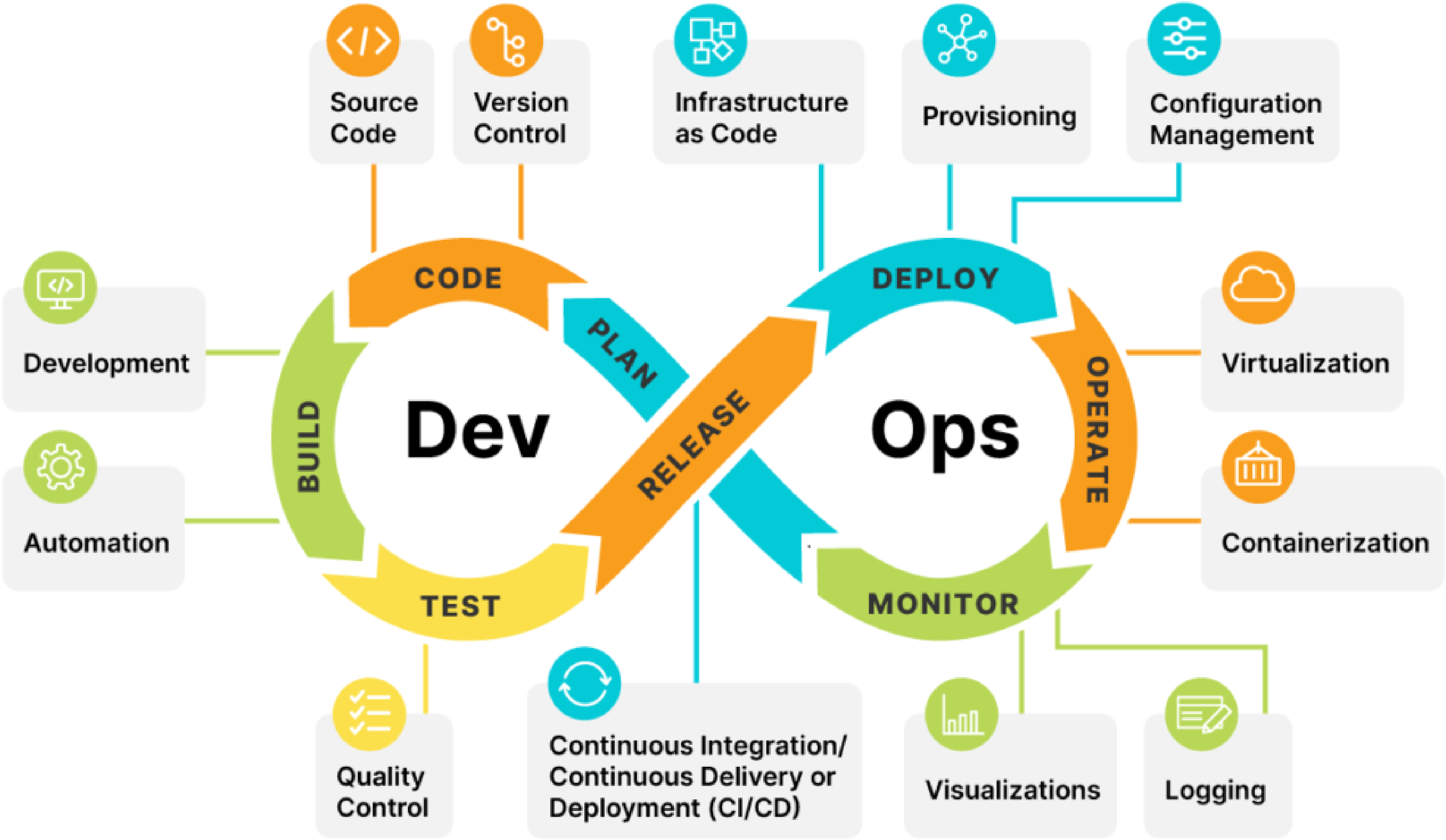
CS304 SOFTWARE ENGINEERING

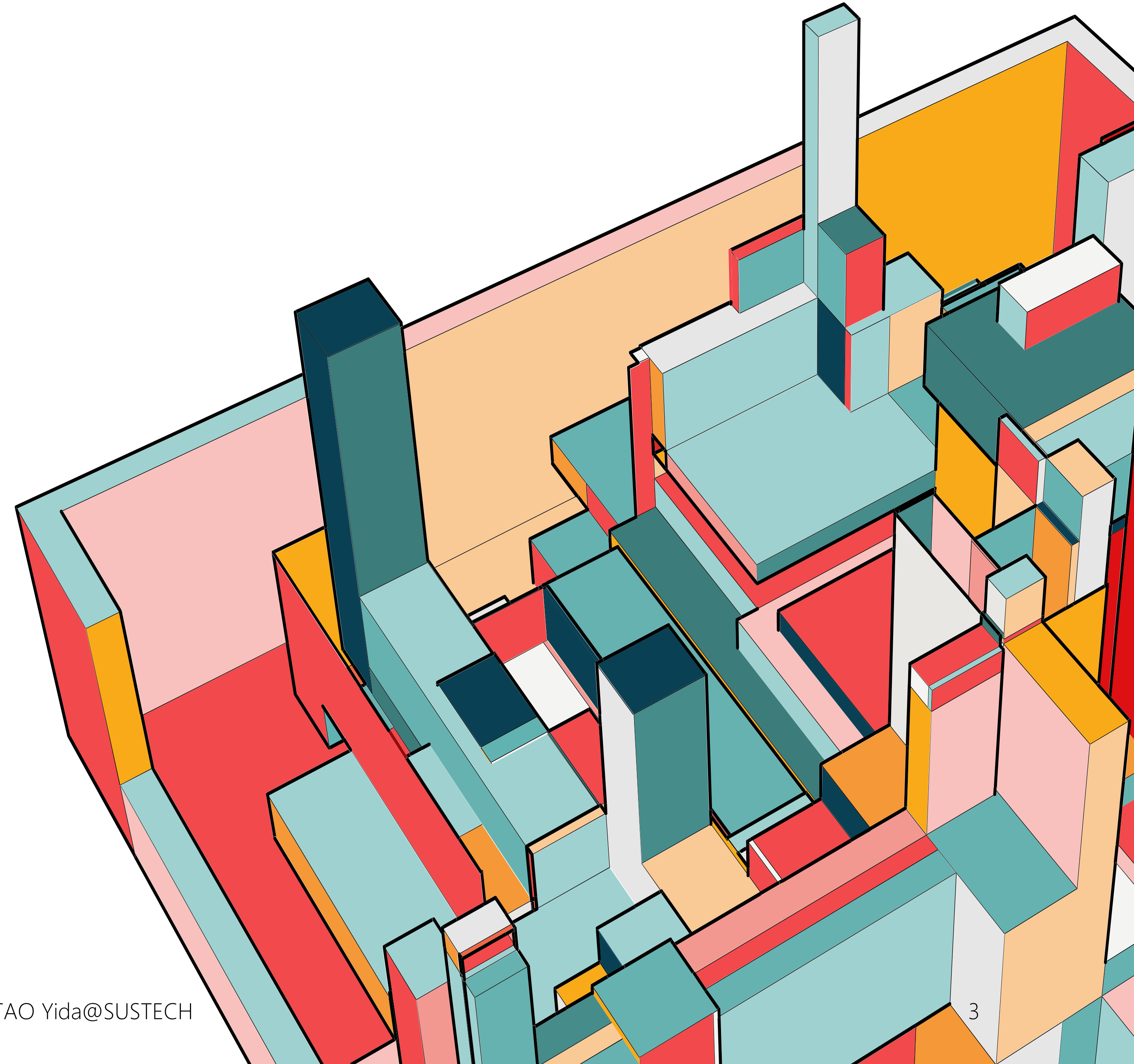
Yida Tao
taoyd@sustech.edu.cn

WHERE ARE WE NOW?



LECTURE 12

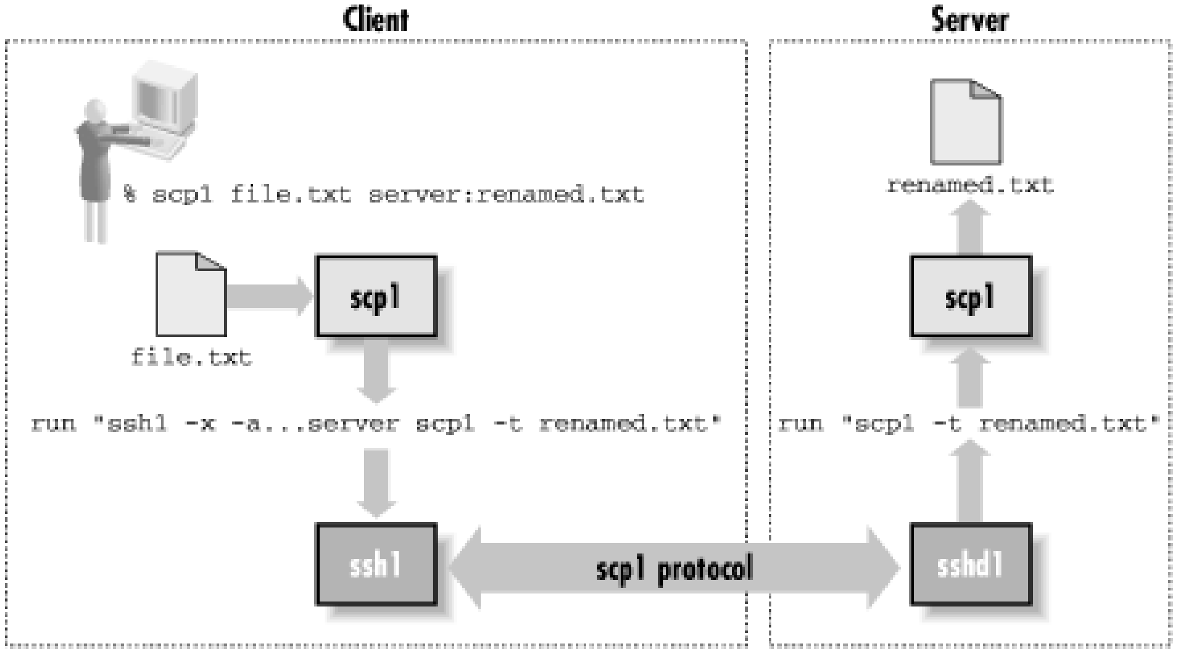
- Deploy & Infrastructure
- Cloud-native Applications
- Case Study



DEPLOY A SIMPLE PROGRAM ON A SINGLE SERVER

1. SFTP the code onto the server
2. SSH to the server
3. Run the code

What if we need to deploy on 50+ servers?



DEPLOY TO 50+ SERVERS

“It is a logistical, time-consuming nightmare. ”

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implemented, less-than-optimal scheduling, and increased contention for the scarce machine resources

-- Jeff Dean. Google. 2002. About running an automated data-processing task as a part of the release process.

DEPLOY TO 50+ SERVERS

“It is a logistical, time-consuming nightmare. ”

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implen scarce machi

Solution 1: automate the deployment through a shell script, which should be reusable, robust, easy-to-maintain

DEPLOY TO 50+ SERVERS

"It is a logistical, time-consuming nightmare. "

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implem… scarce machi…

Solution 2: automate the monitoring of server status, exporting key healthy metrics, detecting anomalies.

DEPLOY TO 50+ SERVERS

"It is a logistical, time-consuming nightmare. "

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implen scarce machi

Solution 3: (autohealing) automate the anomaly handling, e.g., kill and reboot the process

DEPLOY TO 50+ SERVERS

"It is a logistical, time-consuming nightmare. "

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implemented, less-than-optimal scheduling, and increased contention for the scarce machine resources

Solution 4: Isolation, a guarantee that a process can safely proceed without being disturbed by other processes.

DEPLOY TO 50+ SERVERS

"It is a logistical, time-consuming nightmare. "

It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implemented, less-than-optimal scheduling, and increased contention for the scarce machine resources

Solution 5: (automated scheduling) a central service that knows the complete list of machines available to it and can—on demand—pick unoccupied machines and automatically deploy the binary to those machines.

DEPLOY TO 50+ SERVERS

"It is a logistical, time-consuming nightmare. "

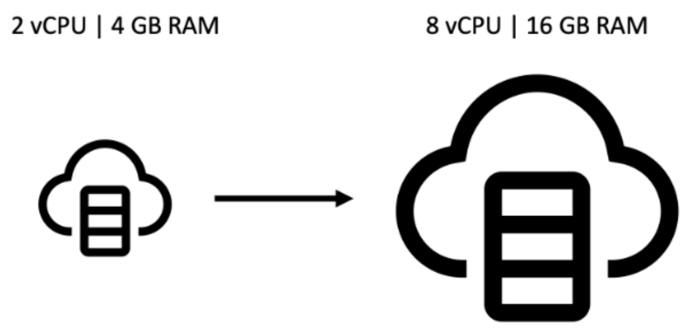
It currently requires getting a list of 50+ machines, starting up a process on each of these 50+ machines, and monitoring its progress on each of the 50+ machines.

- There is no support for automatically migrating the computation to another machine if one of the machines dies
- Monitoring the progress of the jobs is done in an ad hoc manner
- Since processes can interfere with each other, there is a complicated, human-implemented, less-than-optimal scheduling, and increased contention for the scarce machine resources

Solution 5: (autoscaling) automate the settings of configuration parameters.

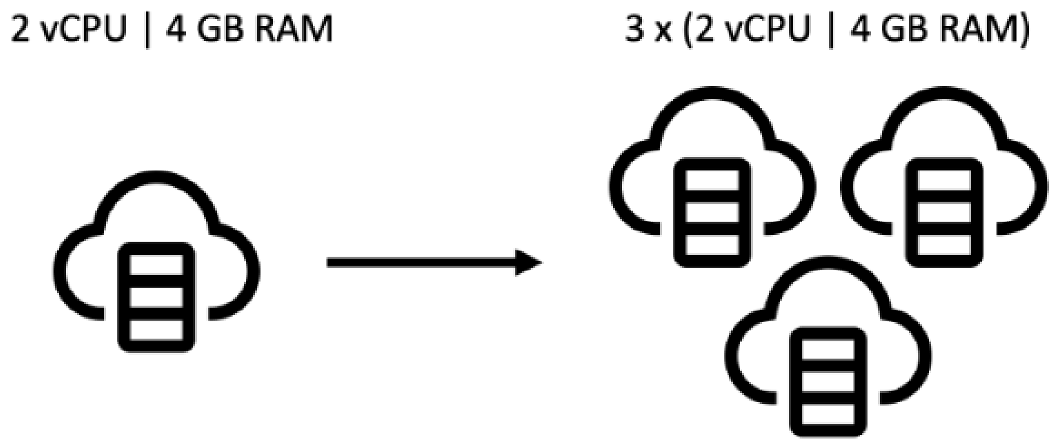
IT'S ALL ABOUT SCALE

Scale up (Vertical scaling)



Adding more resources to an existing server/node

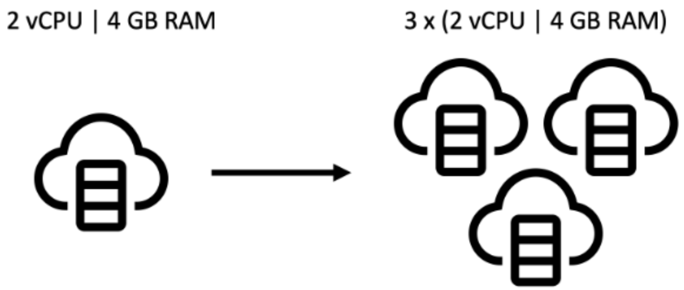
Scale out (Horizontal scaling)



Adding more servers or nodes and distribute the workload across them

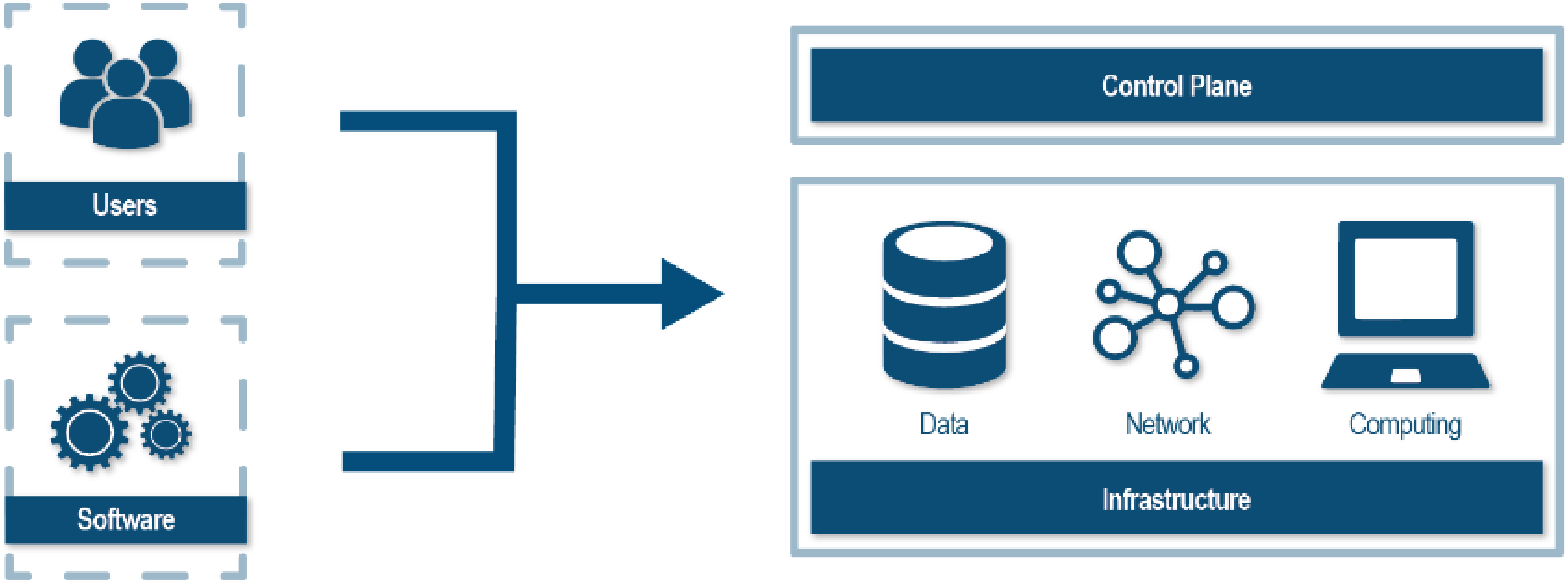
https://developer.ibm.com/articles/scale-up-and-scale-out-vms-vs-containers/

SCALE OUT - COMPLEXITIES



- Load Balancing: With more servers added, it's important to distribute incoming traffic across these servers to ensure that the load is evenly distributed
- Configuration: each new server or node needs to be configured properly to work with the rest of the infrastructure (e.g., network config, security config). As the number of servers grows, the complexity of configuring and managing them can increase.
- Monitoring/Troubleshooting: With more servers in the infrastructure, it becomes more difficult to monitor the system and identify issues.
- Maintenance: With more servers, there is more hardware and software that needs to be maintained and updated, which increases the workload and complexity of managing the infrastructure.

INFRASTRUCTURE (基础设施)

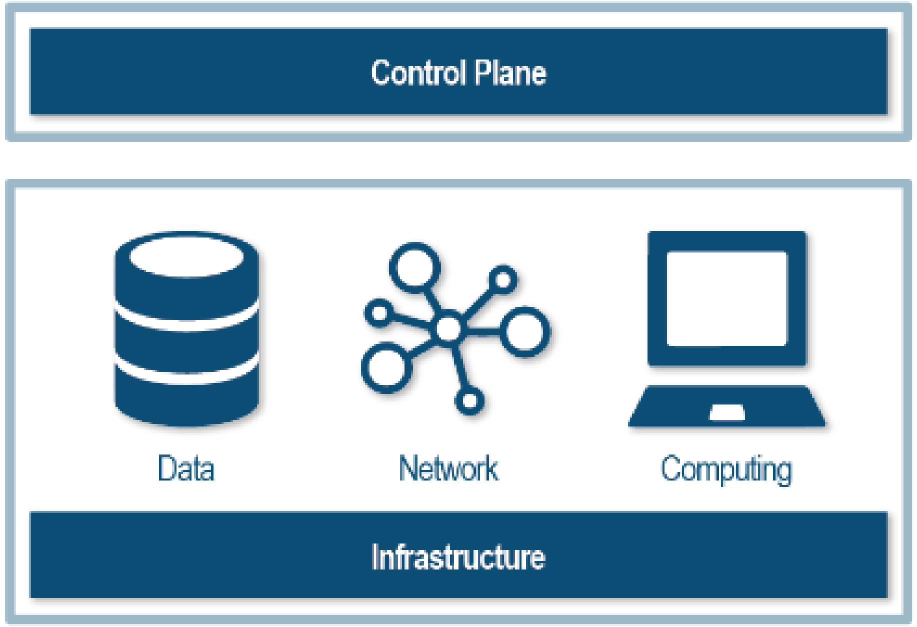


INFRASTRUCTURE (基础设施)

Computing

- **Hardware**: servers, computers, network devices, storage systems, and peripheral devices. Hardware forms the physical foundation of the infrastructure.

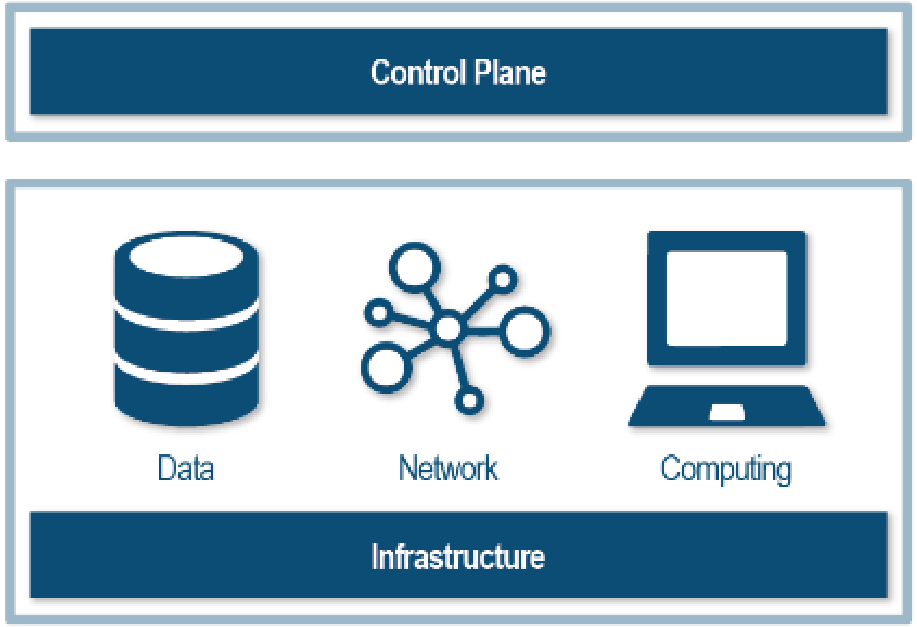
- **Software**: operating systems, applications, firewalls, antivirus software, virtualization, and other software programs that enable various functionalities within the infrastructure.



INFRASTRUCTURE (基础设施)

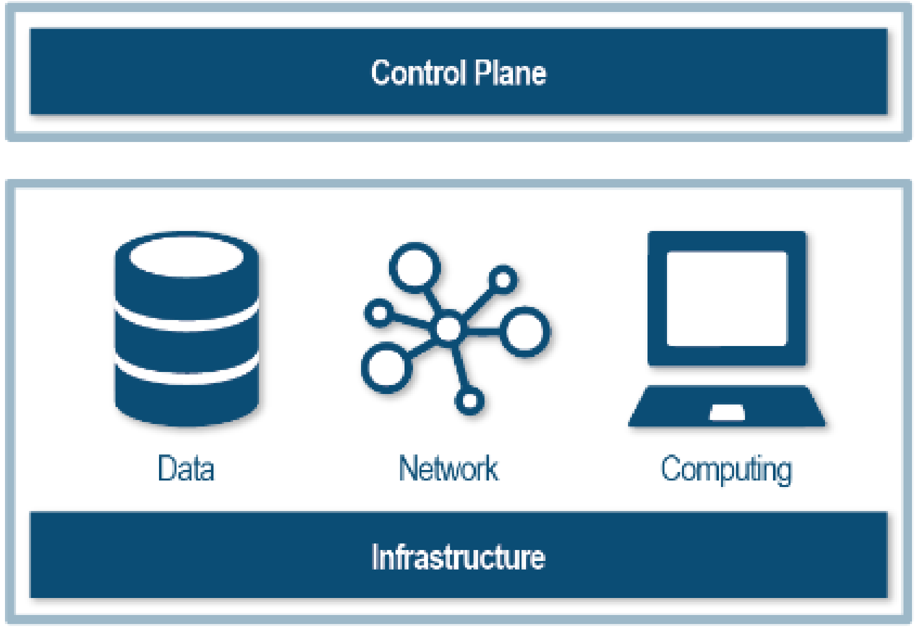
- **Networks**: networking components such as routers, switches, firewalls, and cables connect devices and facilitate data transmission across the infrastructure.

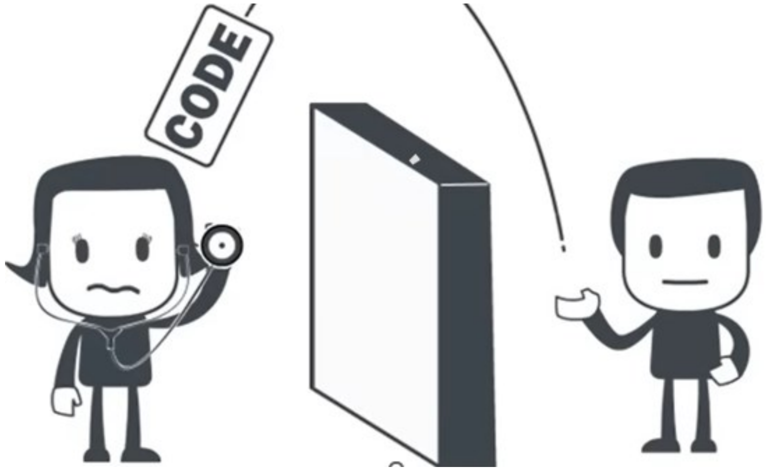
- **Data Centers:** Data center services encompass the planning, construction, and management of physical data center facilities. This includes power and cooling management, rack and cabinet setup, physical security, and monitoring systems to ensure high availability and reliability of the infrastructure.



INFRASTRUCTURE (基础设施)

- **Monitoring and control plane**: captures log files from throughout a network and aggregates them into a single database where they can be sorted, queried, and analyzed manually or by machine algorithms.
- 3 forms of infrastructure monitoring: hardware, network, and application monitoring.
- Infrastructure monitoring creates opportunities to proactively identify security risks and mitigate operational issues before they negatively impact customers.





REMEMBER THIS?

Ops (运维团队) responsibility:

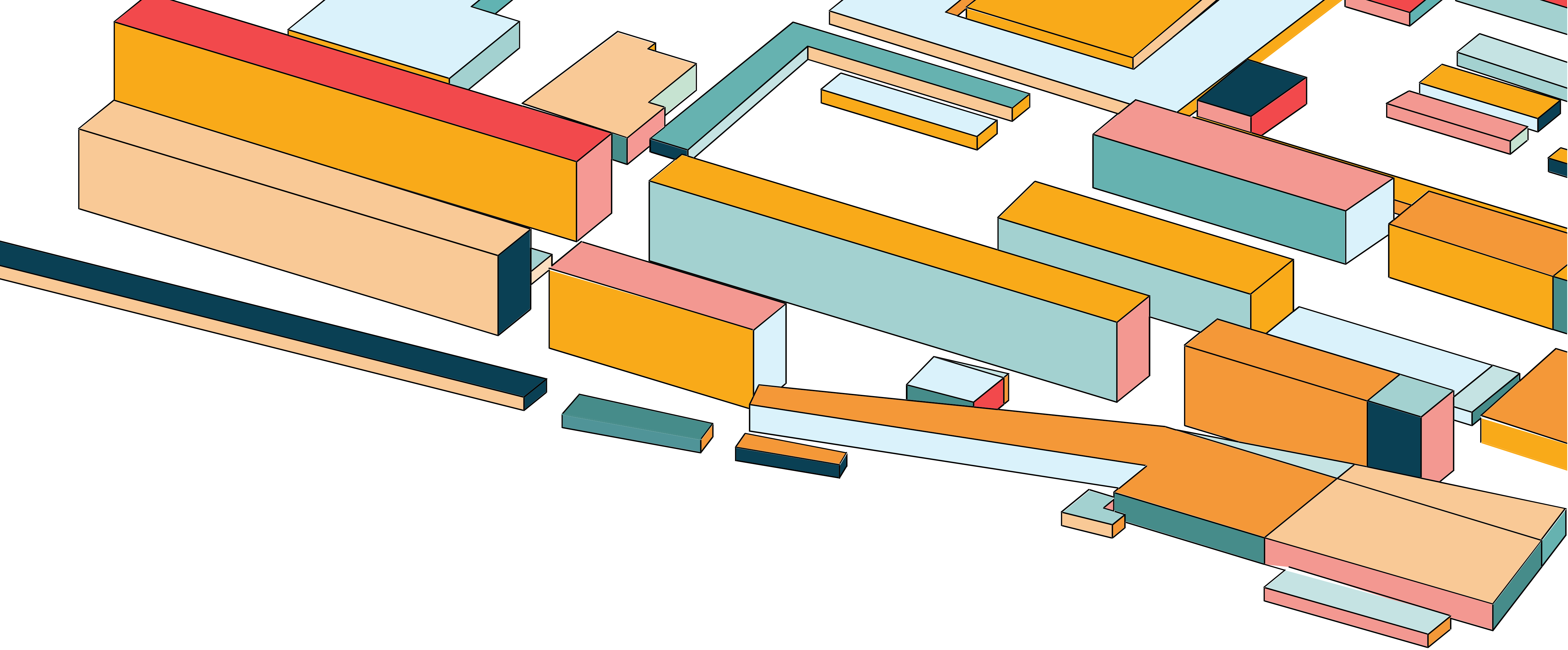
- Deploy
- Release
- Stability of service
- Manage infrastructure
- Maintain & Feedback

TRADITIONAL ON-PREMISE INFRASTRUCTURE

On-premise (本地部署): organizations need to purchase and maintain their own hardware and software in-house, including servers, storage devices, and networking equipment.

Drawbacks

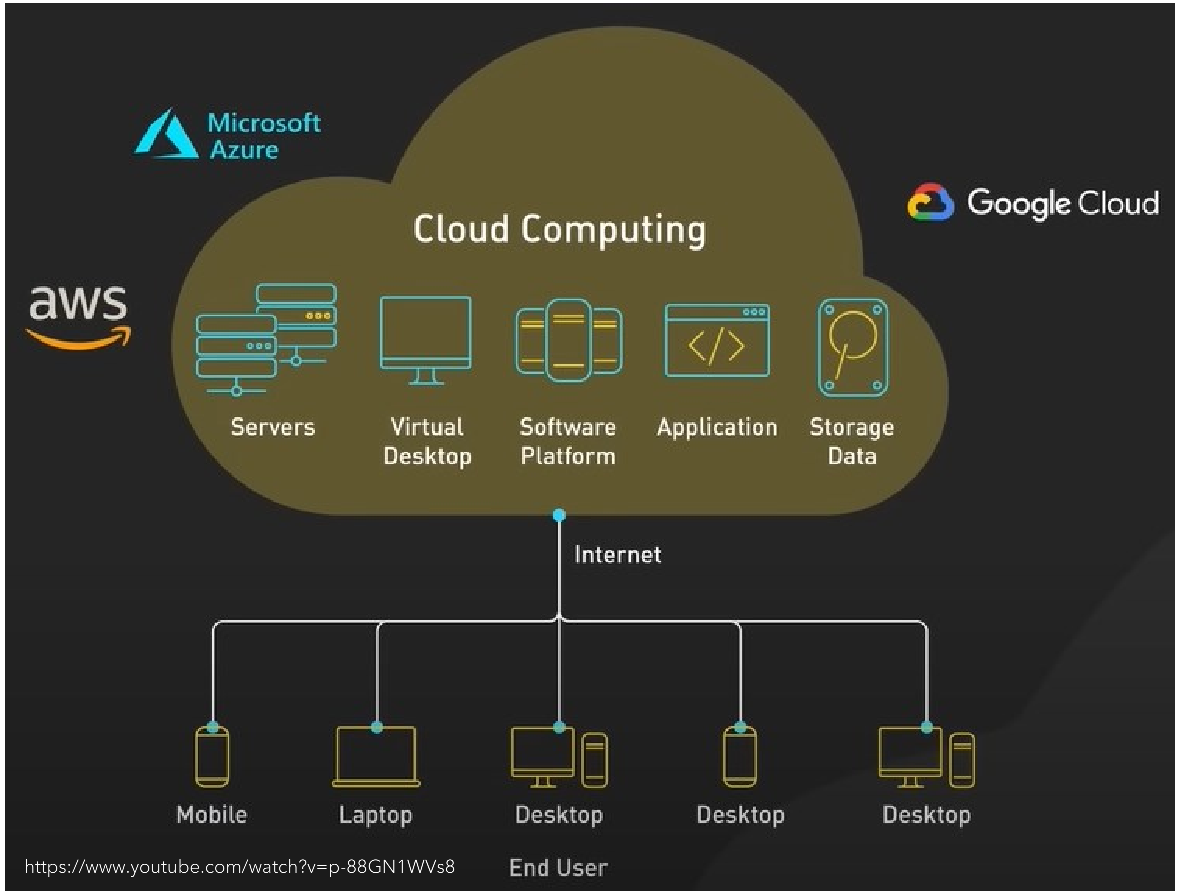
- Significant manual intervention
- Long release cycle
- Complex, time consuming, error-prone



CLOUD-NATIVE APPLICATIONS

CLOUD COMPUTING

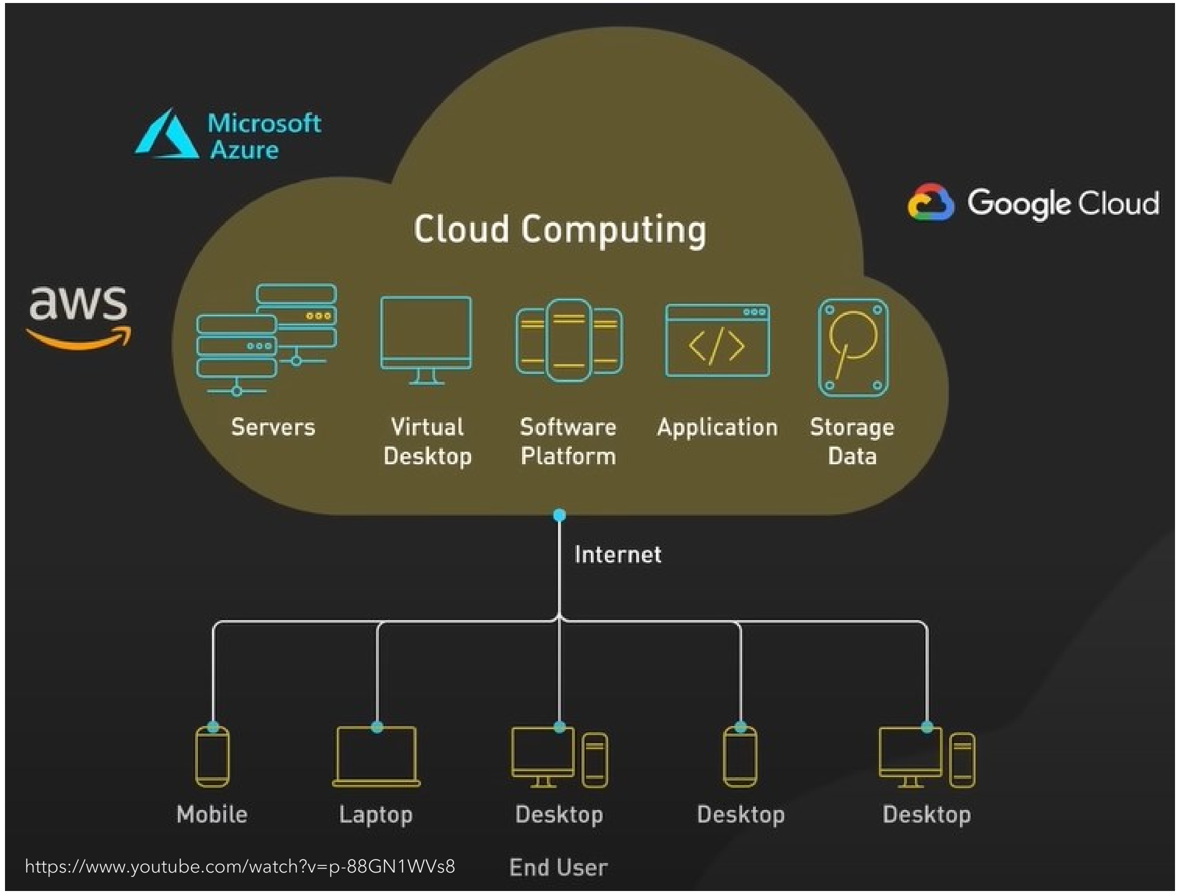
Running applications on computing resources managed by cloud providers (e.g., aws), without having to purchase and manage the infrastructure by ourselves



Still not cloud-native applications

CLOUD COMPUTING

- Free the team from managing infrastructure
- Fast to support new computing resources
- Scaling is effortless
- Cost-effective: cloud providers offer pay-as-you-go pricing models

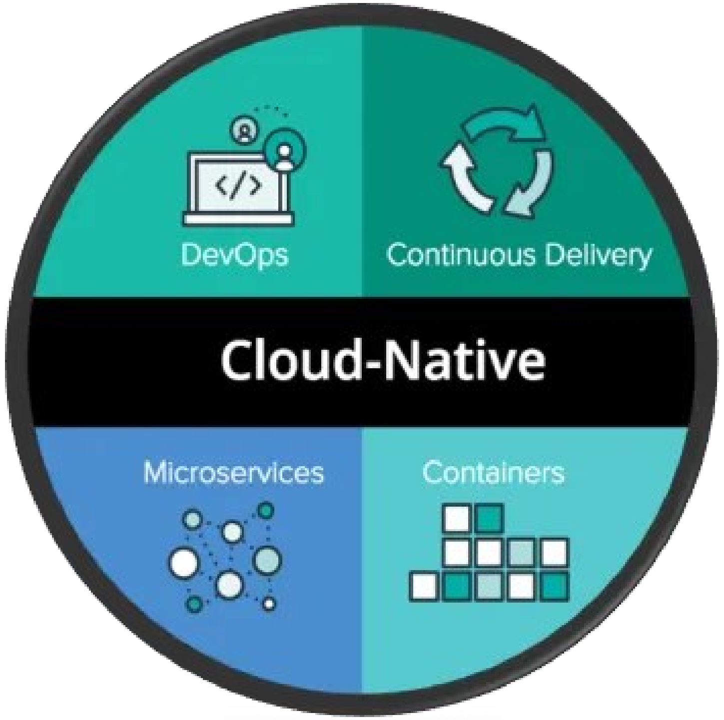


CLOUD NATIVE APPLICATIONS

Cloud native apps are designed and built to exploit the scale, resiliency, and flexibility the cloud provides.

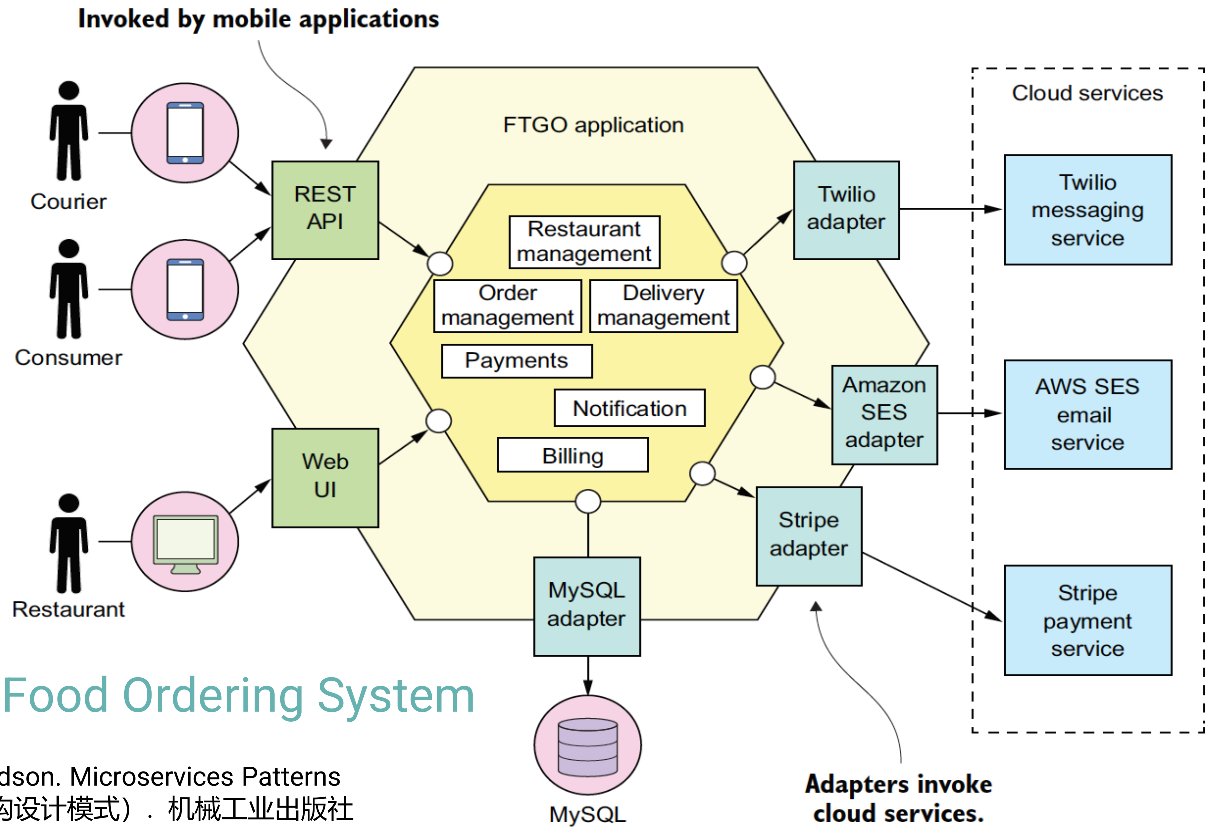
For an application to be considered as cloud-native, it should at least adopt these 4 approaches or technologies

- **Processes**: DevOps & CI/CD
- **Architecture**: Microservices
- **Deployment**: Containers
- **Practice**: Infrastructure as code



MICROSERVICES

Cloud-native applications consist of multiple small, interdependent services called microservices.



Online Food Ordering System

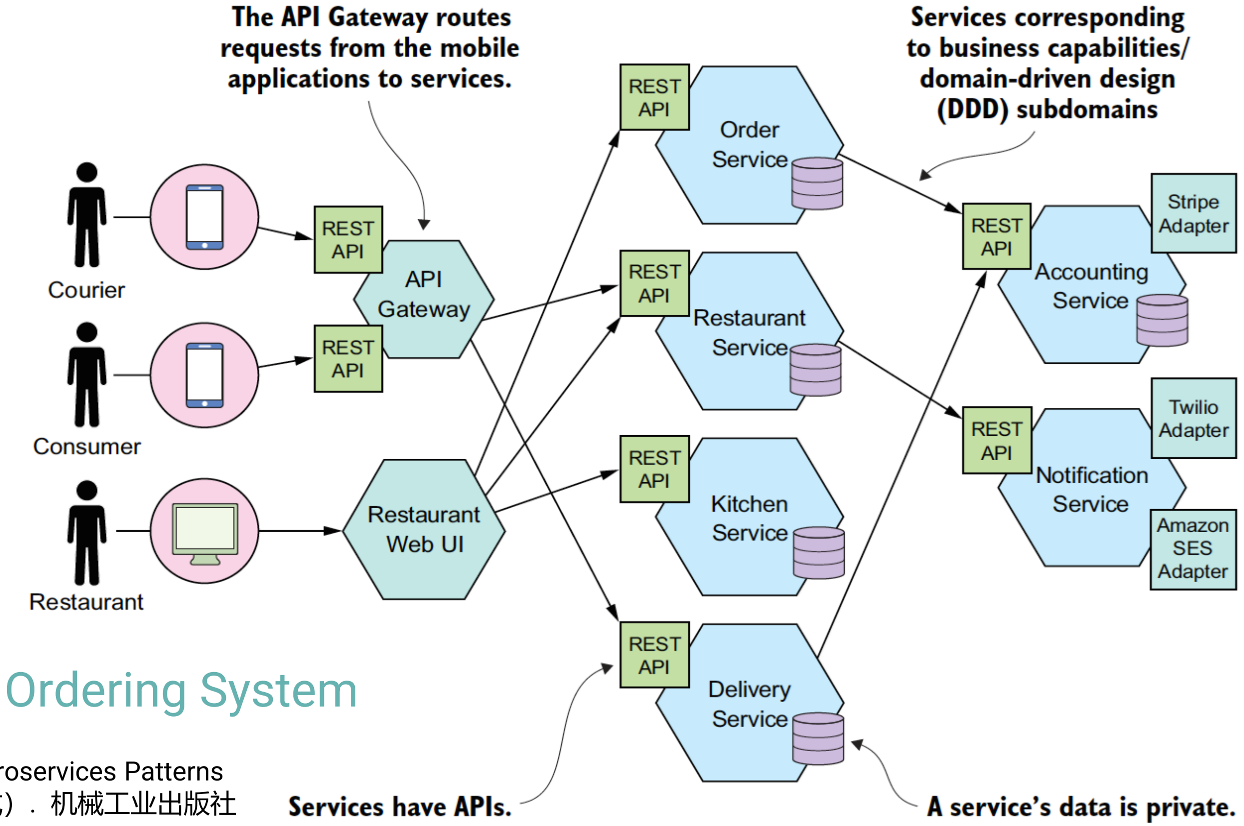
Chris Richardson. Microservices Patterns
（微服务架构设计模式）．机械工业出版社

Monolithic architecture

- The whole application is packaged into a single executable
- Less flexible for large team/code base
- Difficult to scale, wasting deployment resources

MICROSERVICES

Cloud-native applications consist of multiple small, interdependent services called microservices.



Online Food Ordering System

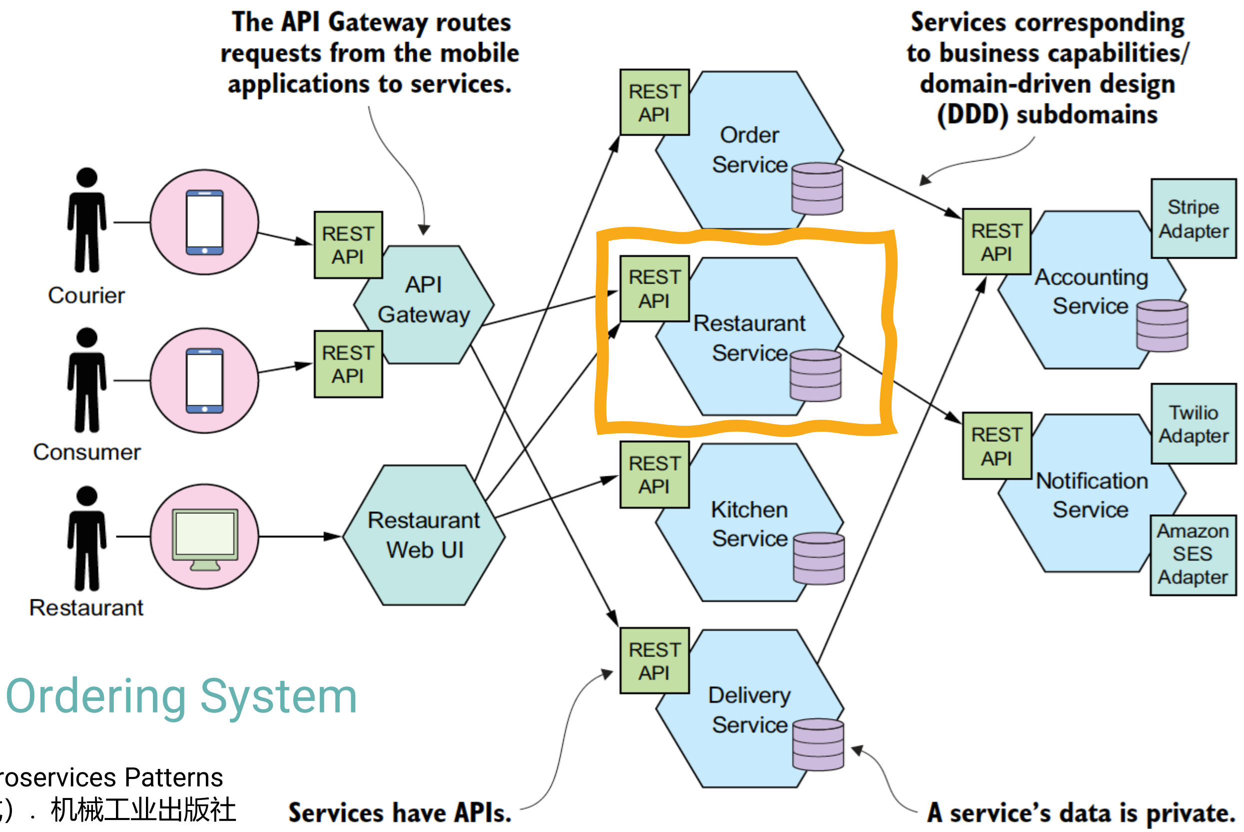
Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

Microservice Architecture

- Microservices allow a large application to be separated into smaller independent parts, with each part having its own responsibility
- Each service can be developed, managed, and deployed independently.
- Services communicate with each other by using well-defined APIs (e.g., REST)
- We could scale only the required microservice

MICROSERVICES

Cloud-native applications consist of multiple small, interdependent services called microservices.



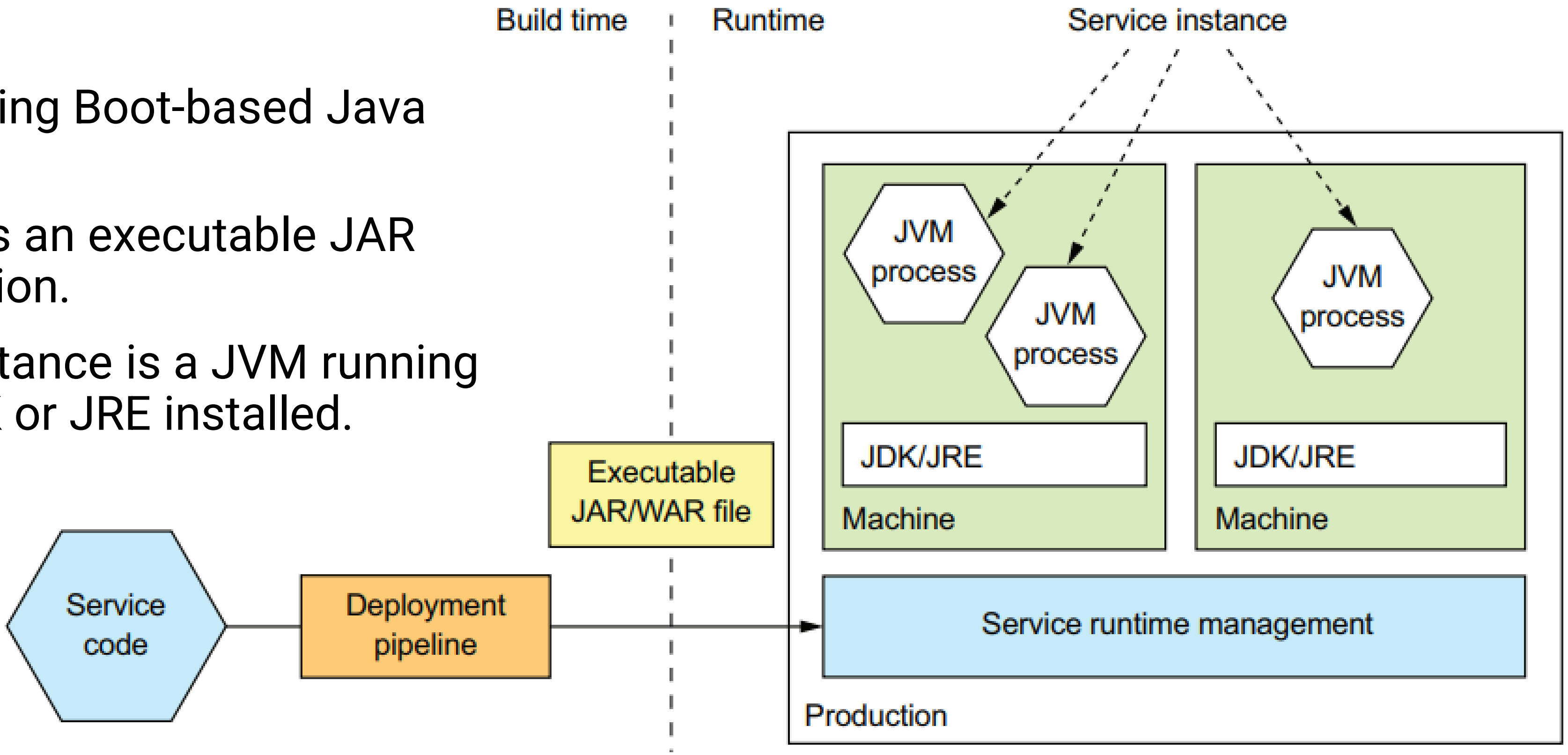
If you're responsible for deploying the restaurant service, what would you do?

Online Food Ordering System

Chris Richardson. Microservices Patterns （微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 1: LANGUAGE-SPECIFIC PACKAGE PATTERN

- `Restaurant service` is a Spring Boot-based Java application
- The deployment pipeline builds an executable JAR file and deploys it into production.
- In production, each service instance is a JVM running on a machine that has the JDK or JRE installed.

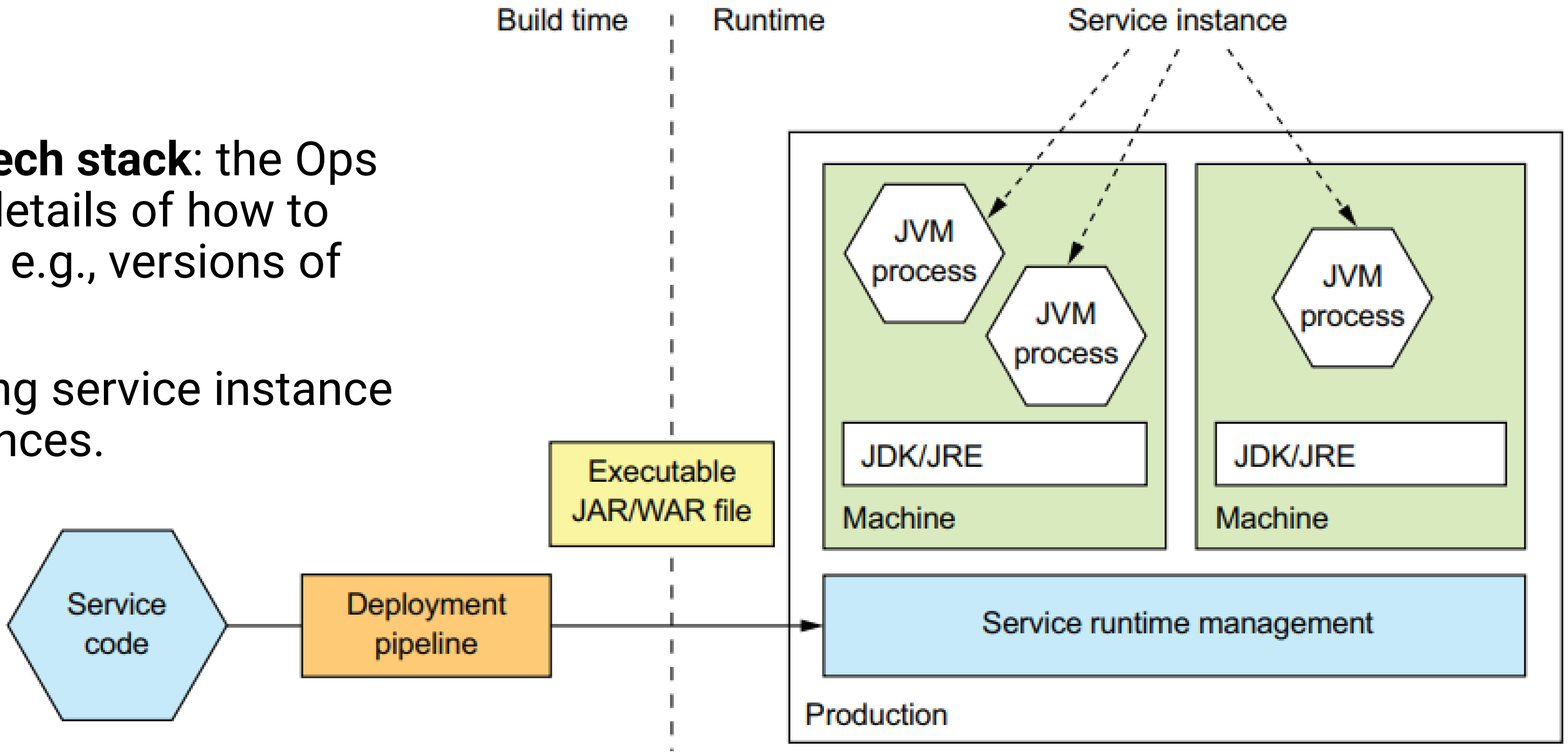


Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 1: LANGUAGE-SPECIFIC PACKAGE PATTERN

Drawbacks

- **Lack of encapsulation of the tech stack**: the Ops team must know the specific details of how to deploy each and every service, e.g., versions of Tomcat or JRE runtime.
- **Lack of isolation**: a misbehaving service instance can impact other service instances.

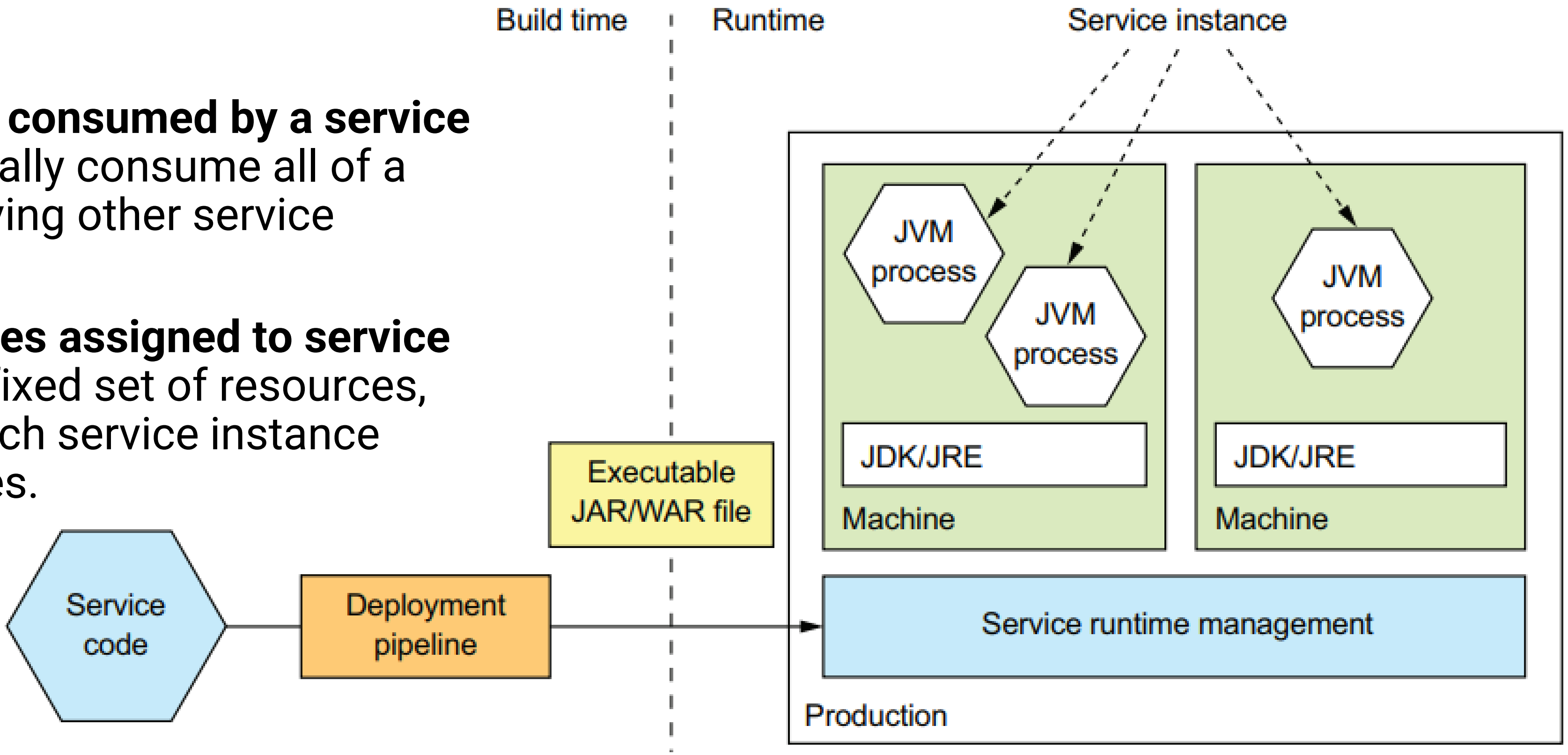


Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 1: LANGUAGE-SPECIFIC PACKAGE PATTERN

Drawbacks

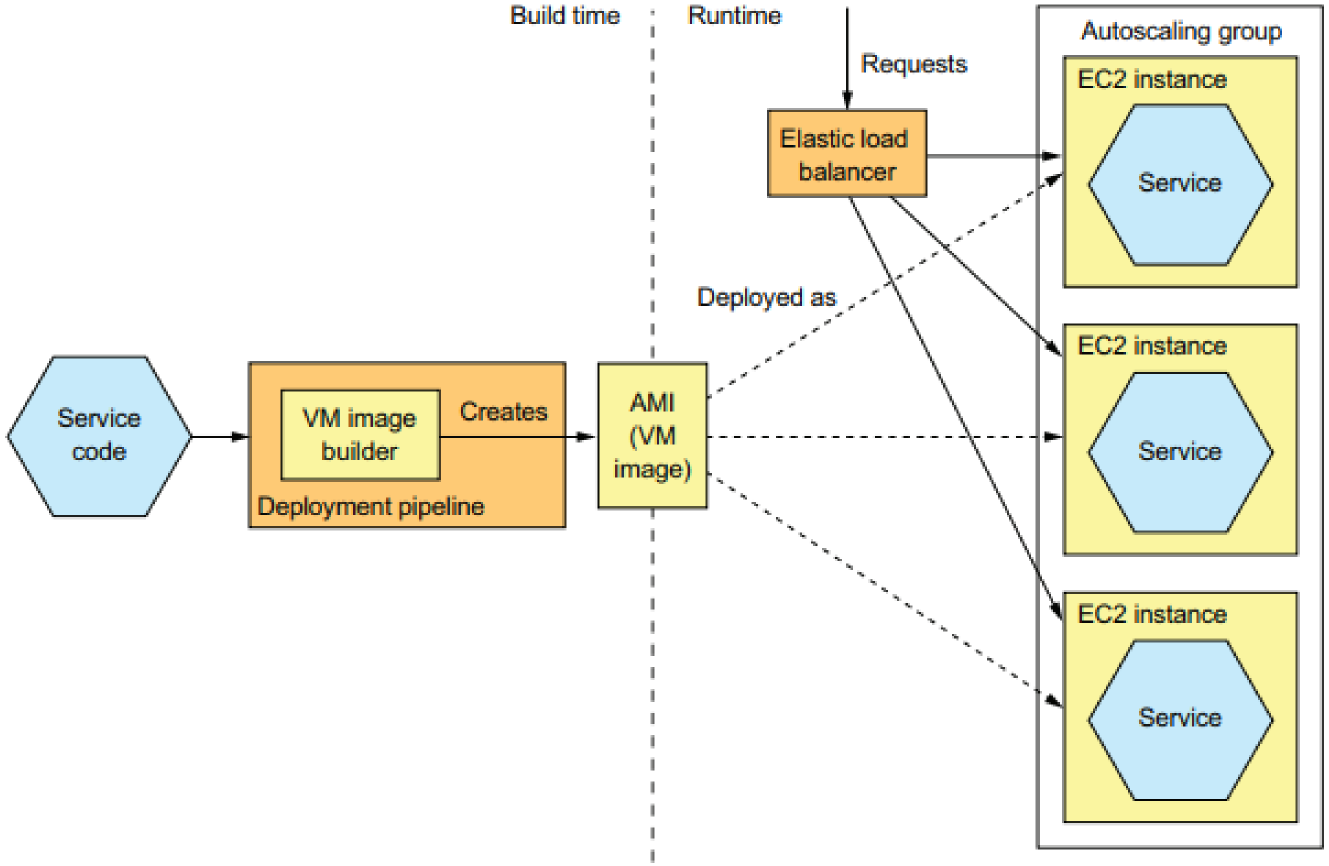
- **Hard to constrain the resources consumed by a service instance**: A process can potentially consume all of a machine's CPU or memory, starving other service instances and OS of resources
- **Manually determine the resources assigned to service instances**: Each machine has a fixed set of resources, CPU, memory, and so on, and each service instance needs some amount of resources.



Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 2: VIRTUAL MACHINE

- Deploy the restaurant service on AWS EC2
- The deployment pipeline now creates a VM image (AMI, Amazon Machine Image), which contains the service' code and whatever software is required to run it (e.g., JDK)
- Each service instance is an EC2 instance created from the AMI
- All EC2 instances are managed by AWS Auto Scaling Group

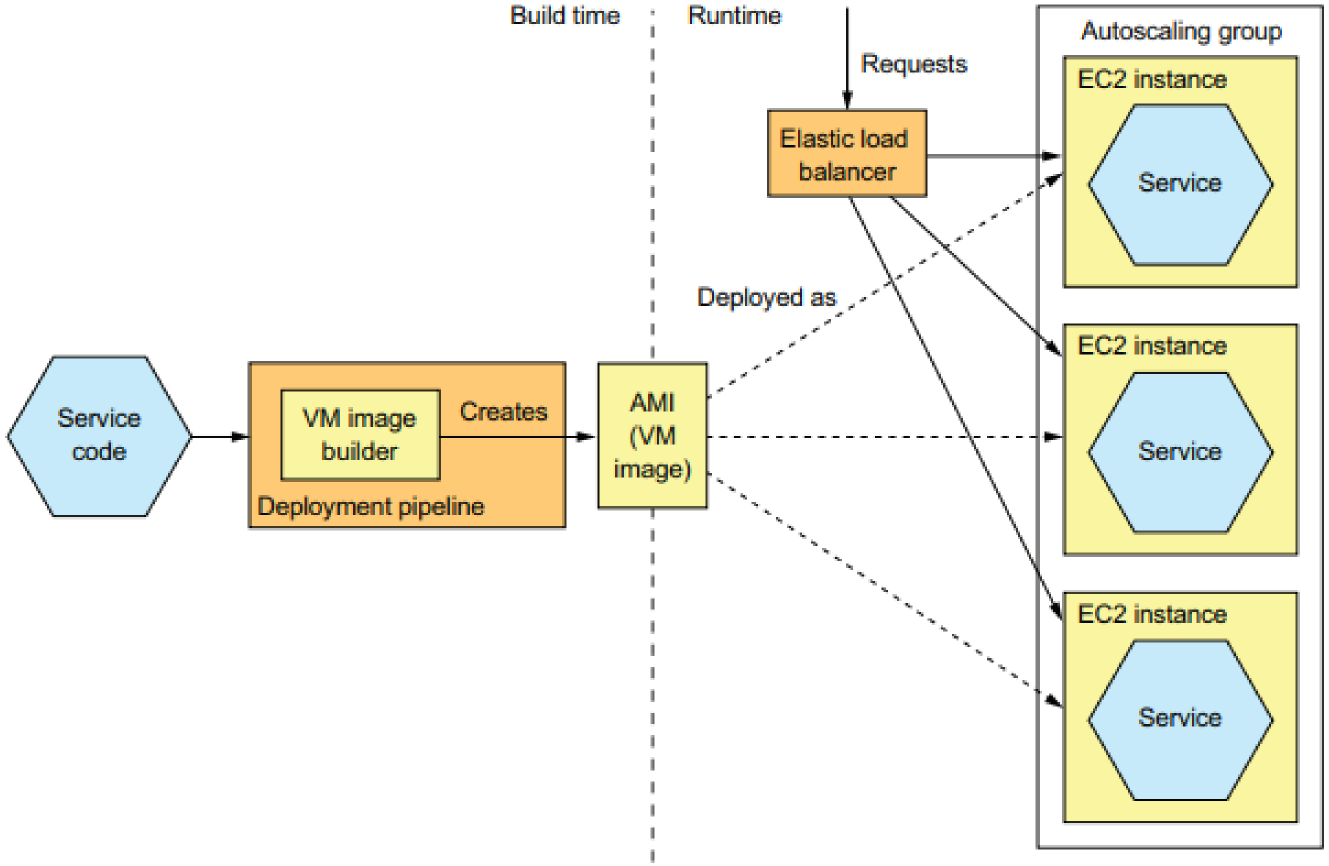


Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 2: VIRTUAL MACHINE

Drawbacks

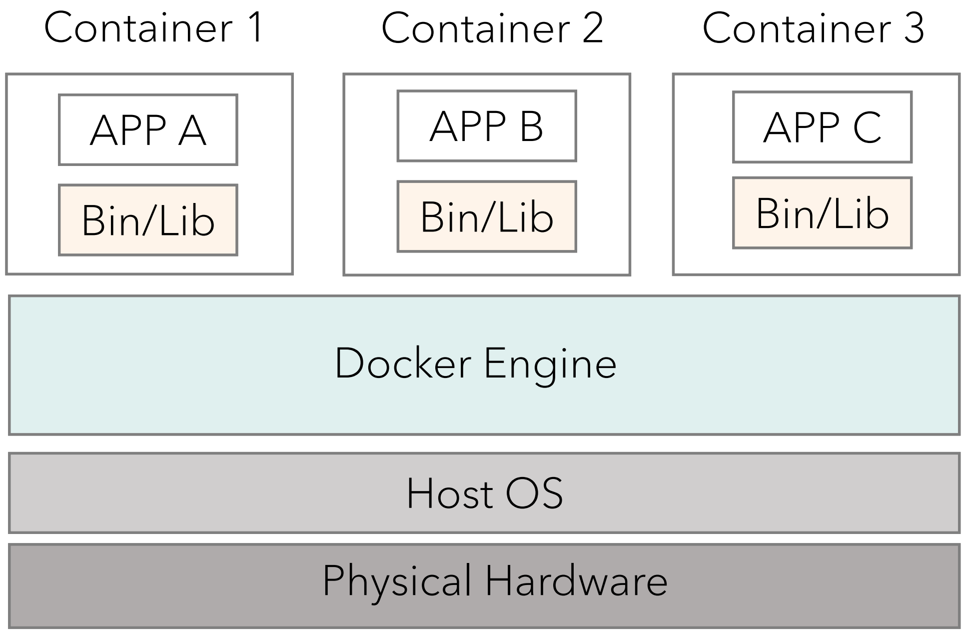
- Less-efficient resource utilization: Each service instance has the overhead of an entire virtual machine, including its OS
- Slow deployment:
  - Building a VM takes minutes.
  - The VM image is large, slow to be moved over the network
  - Slow to boot the OS running inside the VM



Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

DEPLOY PATTERN 3: CONTAINERS

A container is a lightweight and portable unit of software that packages an application and its dependencies together in a single environment.

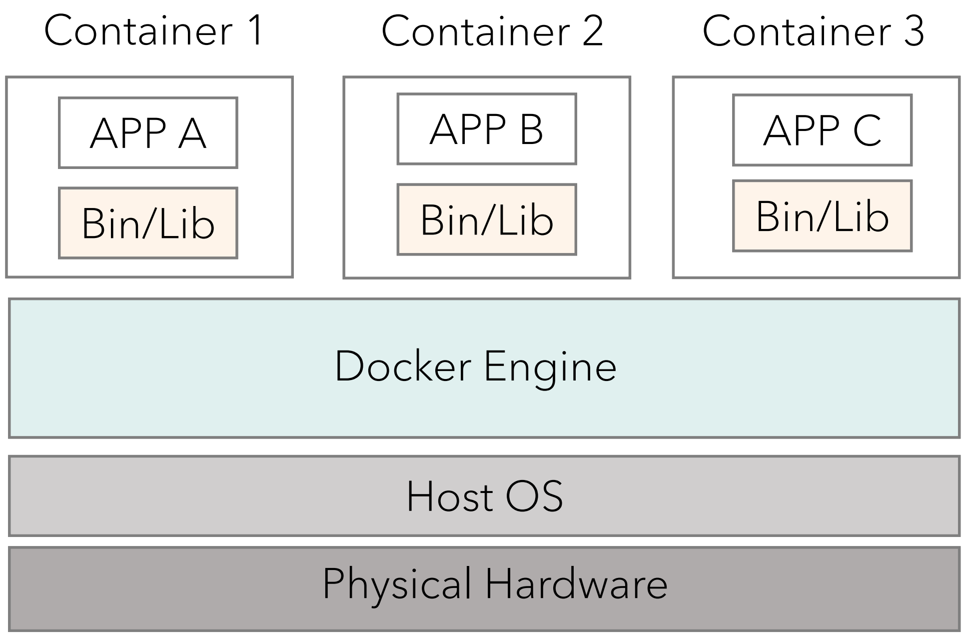


- A container consists of all the dependencies required to run an application, and isolates these dependencies from other containers on the same machine

- From the perspective of a process running in a container, it's as if it's running on its own machine. Each container also has its own root filesystem.

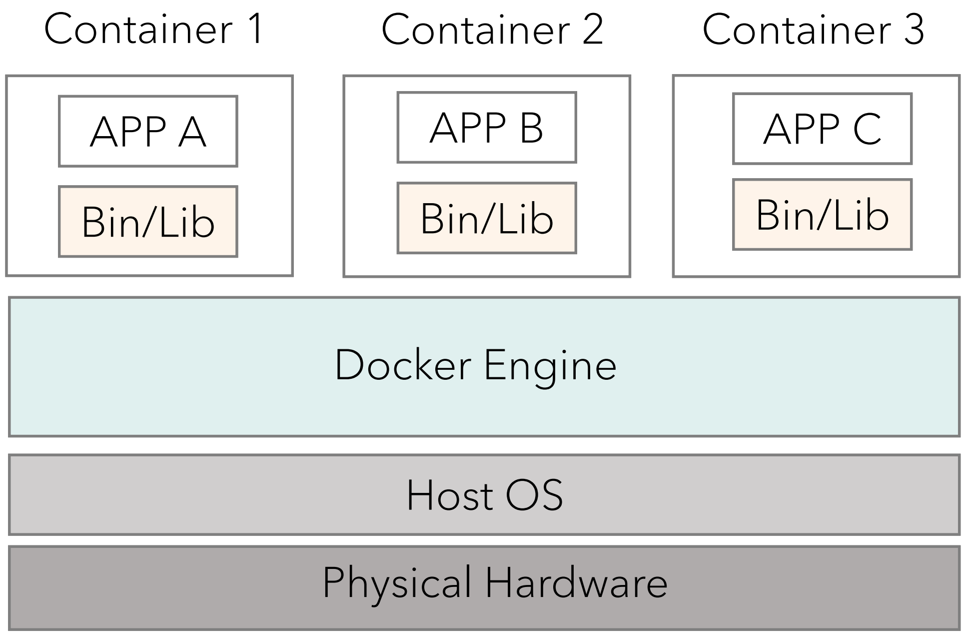
DEPLOY PATTERN 3: CONTAINERS

A container is a lightweight and portable unit of software that packages an application and its dependencies together in a single environment.



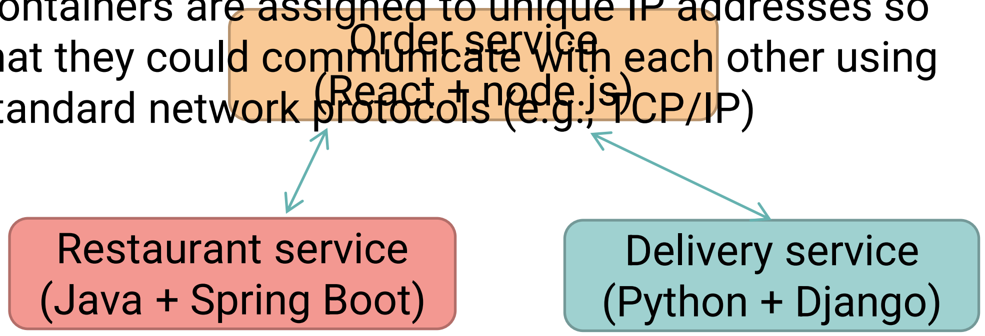
- When you create a container, you can specify its CPU, memory resources, and I/O resources. The container runtime enforces these limits and prevents a container from hogging the resources of its machine.
- Developers can easily create, deploy, and run applications in a consistent and predictable manner across different servers or cloud platforms.

DEPLOY PATTERN 3: CONTAINERS

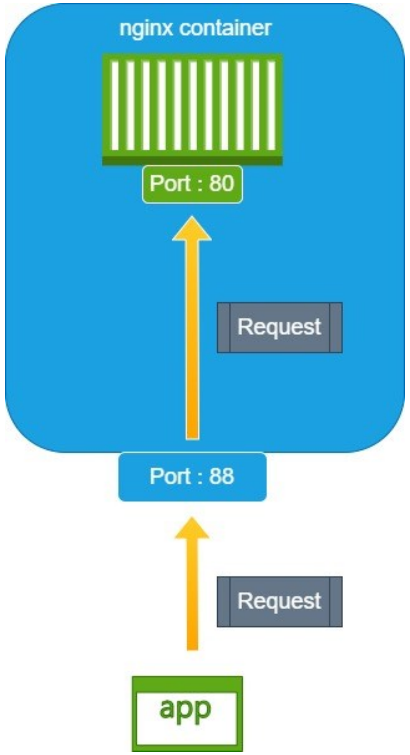


Online Food Ordering System

Container Communication
Containers are assigned to unique IP addresses so that they could communicate with each other using standard network protocols (e.g., TCP/IP)



DEPLOY PATTERN 3: CONTAINERS

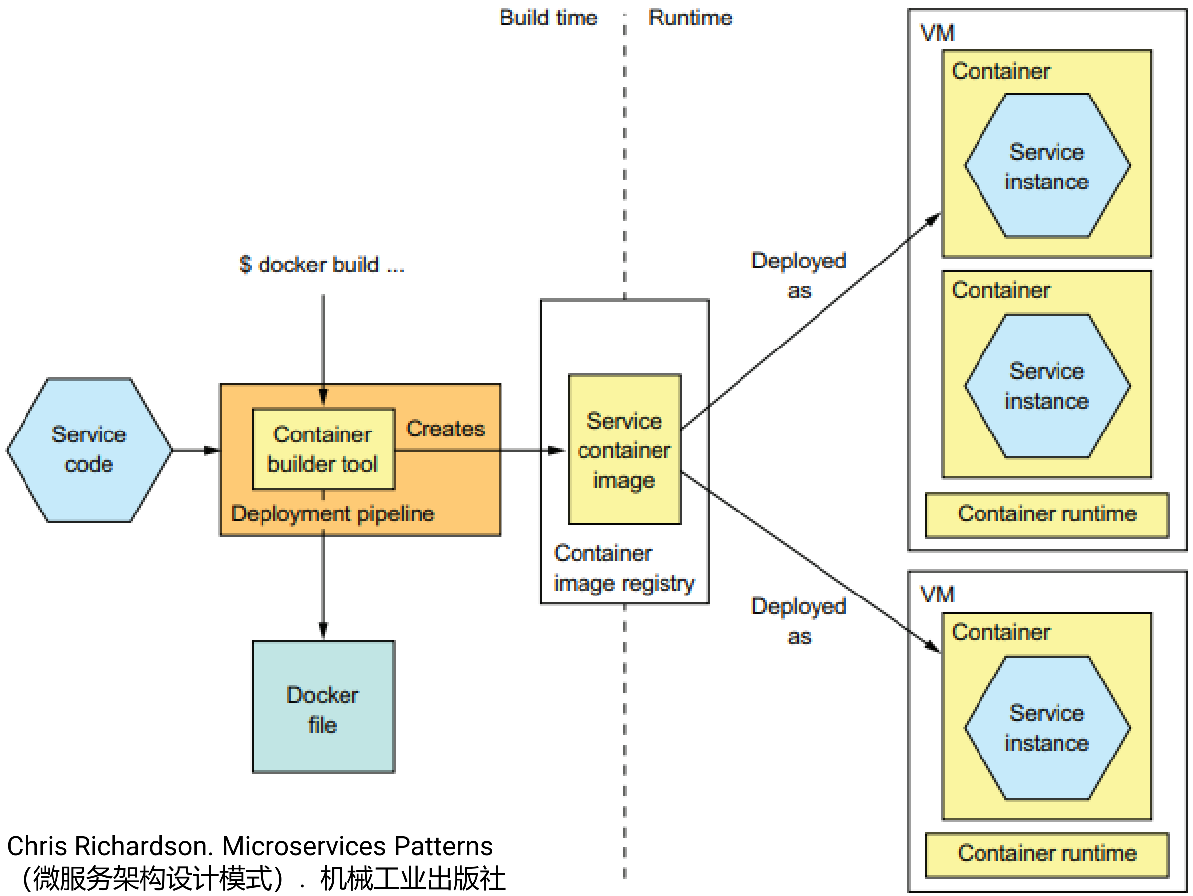


Container Communication

- All Java processes can, for example, listen on port 8080.

- Containers could expose **services** to the outside world by mapping container's port to the port on the host machine (port forwarding)

DEPLOY PATTERN 3: CONTAINERS



Chris Richardson. Microservices Patterns（微服务架构设计模式）. 机械工业出版社

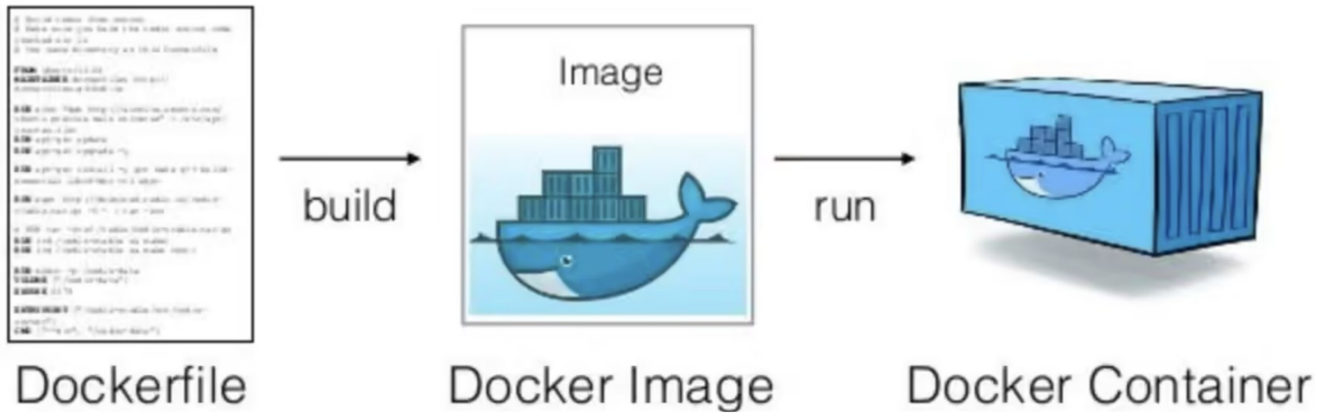
- At build-time, the deployment pipeline uses a container image-building tool, which reads the service's code and a description of the image, to create the container image and stores it in a registry
- At runtime, the container image is pulled from the registry. The service consists of multiple containers instantiated from that image.
- Containers typically run on virtual machines. A single VM will usually run multiple containers.

DOCKER

Docker is one of the most popular platforms for building, deploying, and managing containers

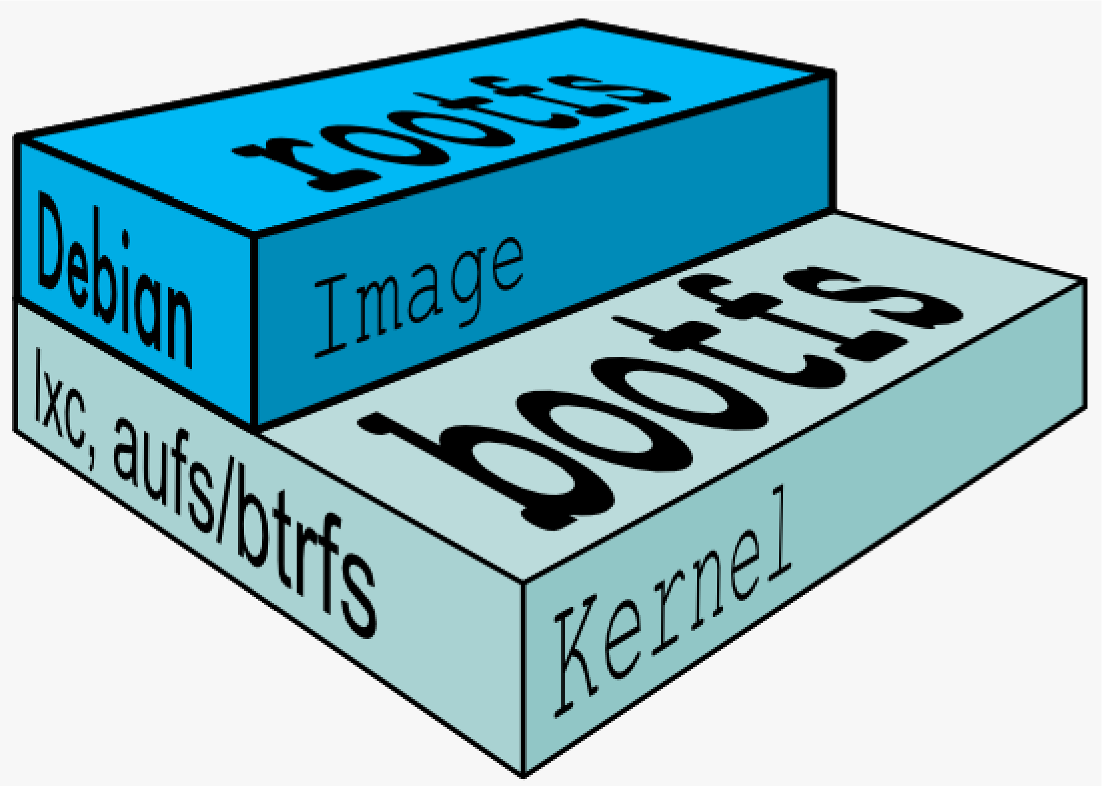
Docker image (镜像): a read-only template that acts as a set of instructions to create a container.

Dockerfile: a script that defines the instructions for building the image



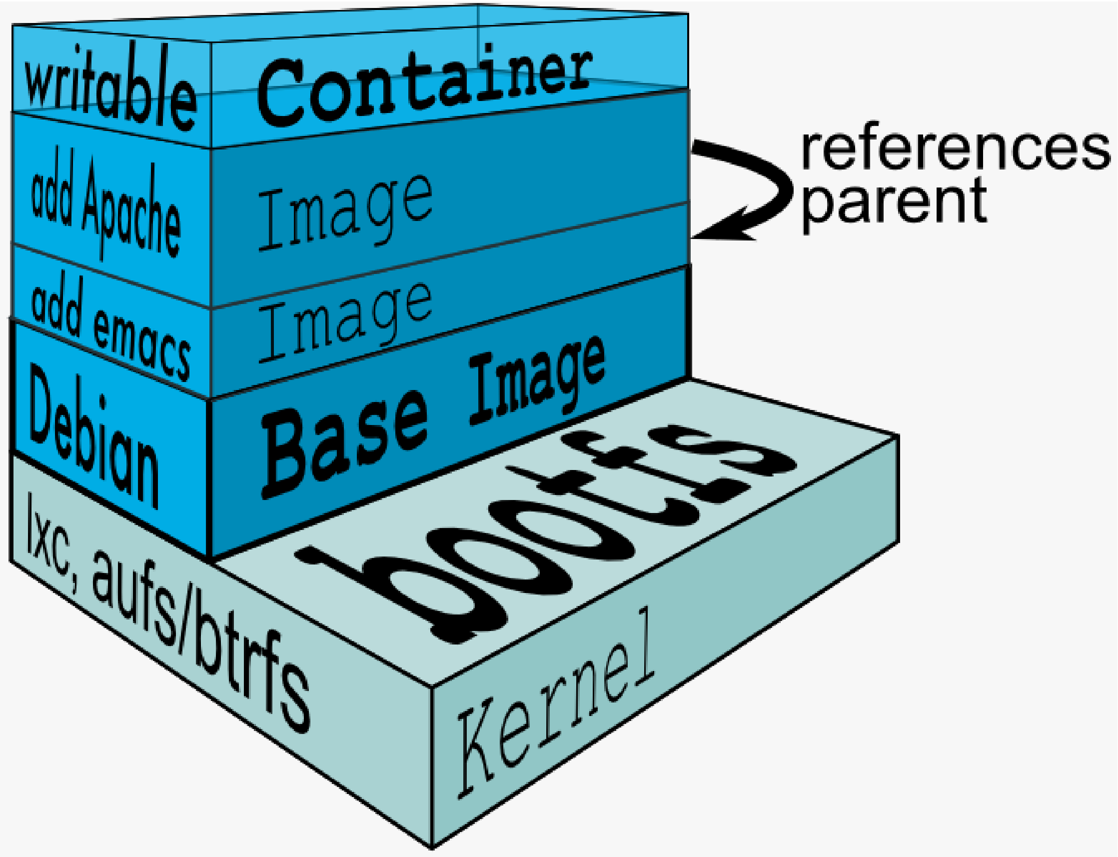
Docker container: a running instance of a Docker image

DOCKER IMAGES



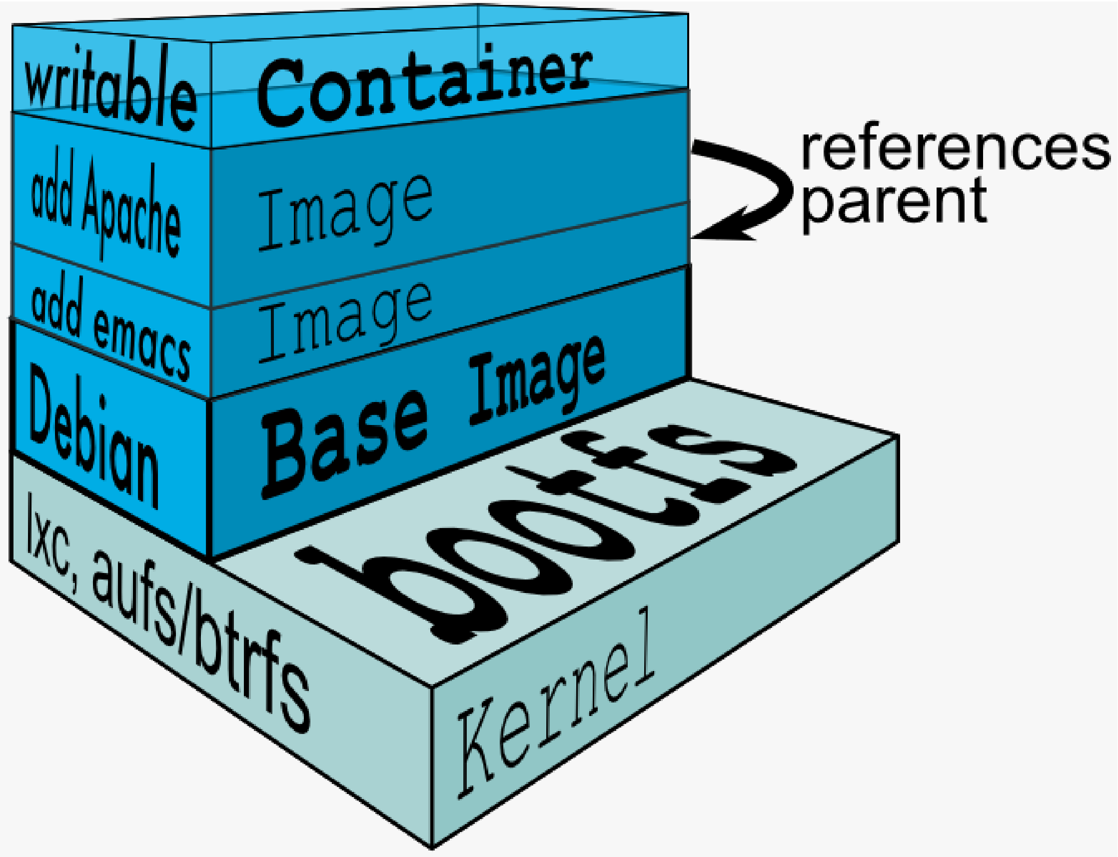
- Docker images consist of many layers. Each layer is built on top of another layer to form a series of intermediate images
- Kernel: Host machine’s kernel
- Base image:
  - Different Linux distribution, e.g., Debian, Ubuntu
  - Various pre-installed software images, e.g., OpenJDK
  - Or a blank image with nothing included

DOCKER IMAGES



- Images (read-only) can be layered on top of one another, specified by Dockerfile
- Each time Docker launches a container from an image, it adds a thin writable layer, known as the container layer, which stores all changes to the container throughout its runtime.

DOCKER IMAGES



- The top layer has **read-write** permissions, and all the remaining layers have **read-only** permissions
- Several containers may share access to the same underlying level of a Docker image, but write the changes locally and uniquely to each other

DOCKERFILE

Listing 12.1 The Dockerfile used to build Restaurant Service

```
FROM openjdk:8u171-jre-alpine
RUN apk --no-cache add curl
CMD java ${JAVA_OPTS} -jar ftgo-restaurant-service.jar
HEALTHCHECK --start-period=30s --
     interval=5s CMD curl http://localhost:8080/actuator/health || exit 1
COPY build/libs/ftgo-restaurant-service.jar .
```

The base image

Install curl for use by the health check.

Configure Docker to run java -jar .. when the container is started.

Configure Docker to invoke the health check endpoint.

Copies the JAR in Gradle's build directory into the image

The base image openjdk:8u171-jre-alpine is a minimal footprint Linux image containing the JRE

Chris Richardson. Microservices Pattern（微服务架构设计模式）. 机械工业出版社

USING DOCKERFILE TO BUILD DOCKER IMAGE

Once you've written the Dockerfile, you can then build the image.

Listing 12.2 The shell commands used to build the container image for Restaurant Service

```
cd ftgo-restaurant-service
../gradlew assemble
docker build -t ftgo-restaurant-service .
```

Change to the service's directory.

Build the service's JAR.

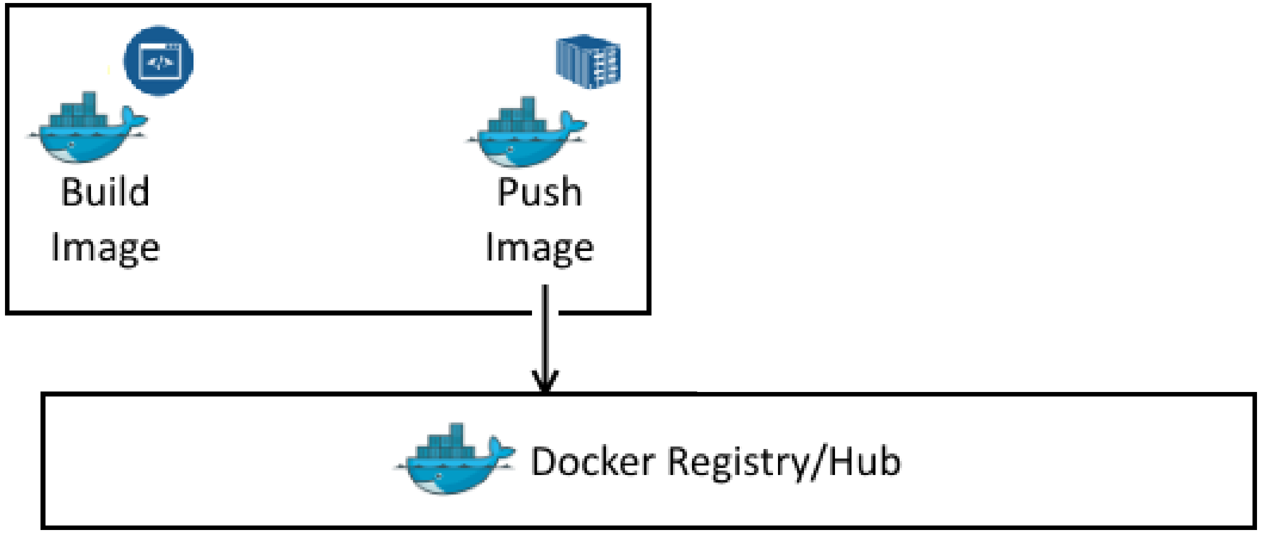
Build the image.

Chris Richardson. Microservices Pattern（微服务架构设计模式）. 机械工业出版社

PUSING A DOCKER IMAGE TO A REGISTRY

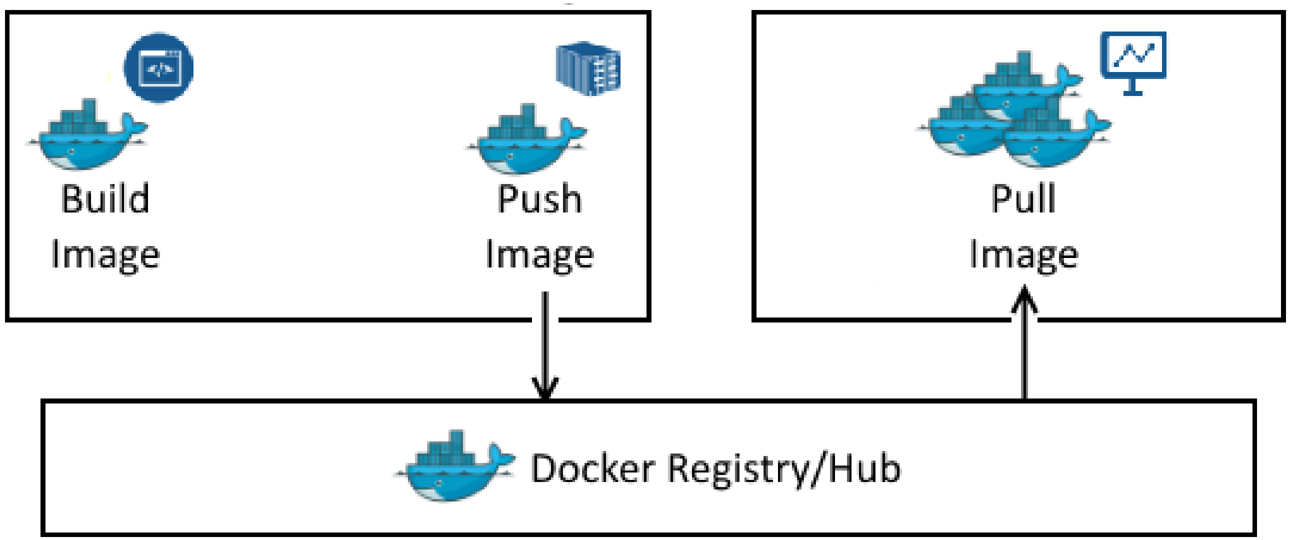
The final step of the build process is to push the newly built Docker image to a registry.
A Docker registry is the equivalent of a Java Maven repository for Java libraries

```
docker push registry.acme.com/ftgo-restaurant-service:1.0.0.RELEASE
```



PUSING A DOCKER IMAGE TO A REGISTRY

The final step of the build process is to push the newly built Docker image to a registry.
A Docker registry is the equivalent of a Java Maven repository for Java libraries



- The container infrastructure will pull the image from the registry onto a production server.
- It will then create one or more containers from that image. Each container is an instance of your service.

RUN A DOCKER CONTAINER

The `docker run` command pulls the image from the registry if necessary. It then creates and starts the container, which runs the `java -jar` command specified in the `Dockerfile`.

Listing 12.3 Using `docker run` to run a containerized service

```
docker run \
  -d  \
  --name ftgo-restaurant-service  \
  -p 8082:8080  \
  -e SPRING_DATASOURCE_URL=... -e SPRING_DATASOURCE_USERNAME=...  \
  -e SPRING_DATASOURCE_PASSWORD=... \
  registry.acme.com/ftgo-restaurant-service:1.0.0.RELEASE
```

Runs it as a background daemon

The name of the container

Binds port 8080 of the container to port 8082 of the host machine

Environment variables

Image to run

Chris Richardson. Microservices Pattern（微服务架构设计模式）. 机械工业出版社

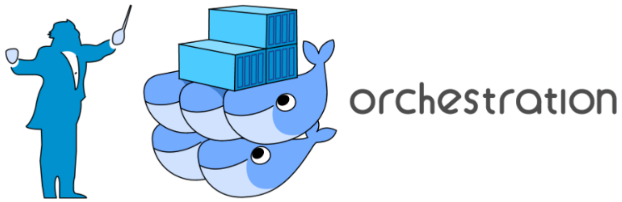
DEPLOY PATTERN 3: CONTAINERS

Drawbacks

- Managing container images
- Managing the container infrastructures

To deploy services reliably, you must use a **Docker orchestration framework**

CONTAINER ORCHESTRATION



Use container orchestration (编排) to **automate and manage** tasks such as:

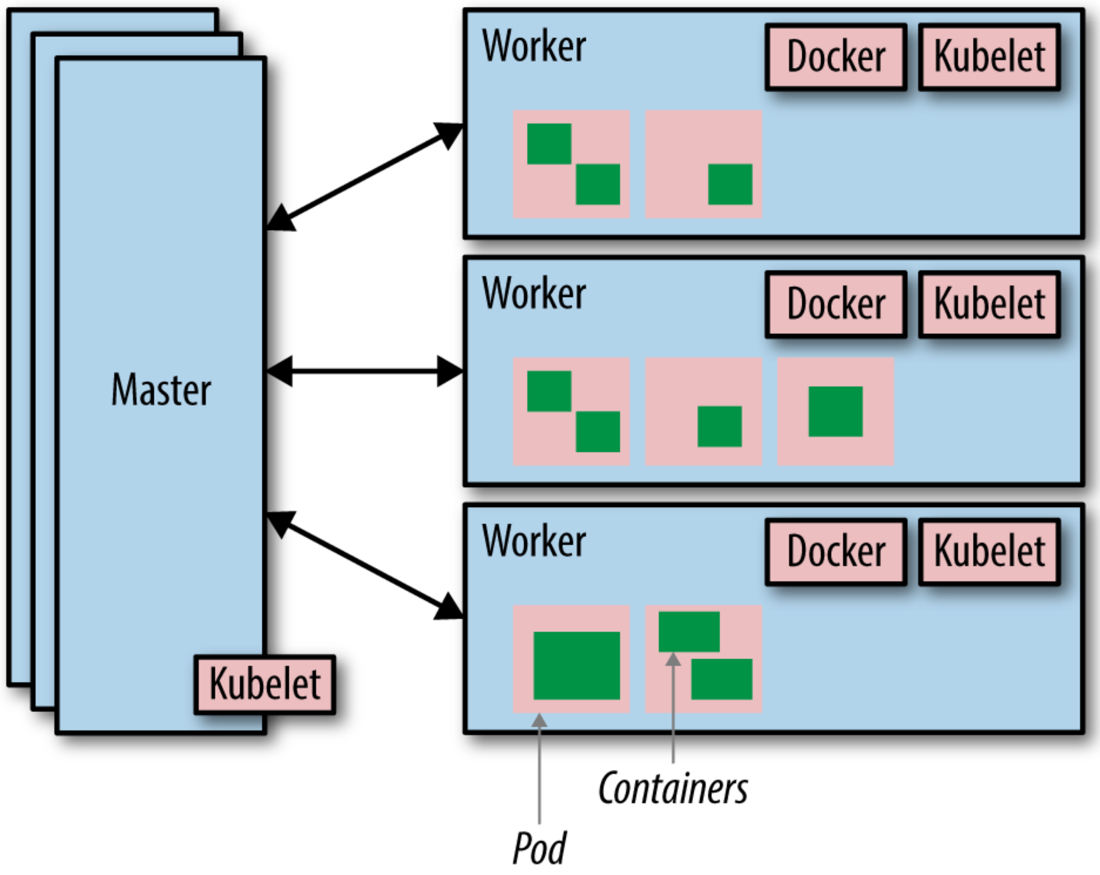
- Deployment
- Configuration
- Scheduling
- Resource allocation
- Container availability
- Load balancing and traffic routing
- Scaling or removing containers based on balancing workloads across your infrastructure
- Monitoring container health
- Keeping interactions between containers secure

CONTAINER ORCHESTRATION TOOLS

Docker Swarm and Kubernetes (K8s) are both container orchestration tools that allow for the management and deployment of containerized applications **at scale**

KUBERNETES ARCHITECTURE

https://enabling-cloud.github.io/kubernetes-learning/

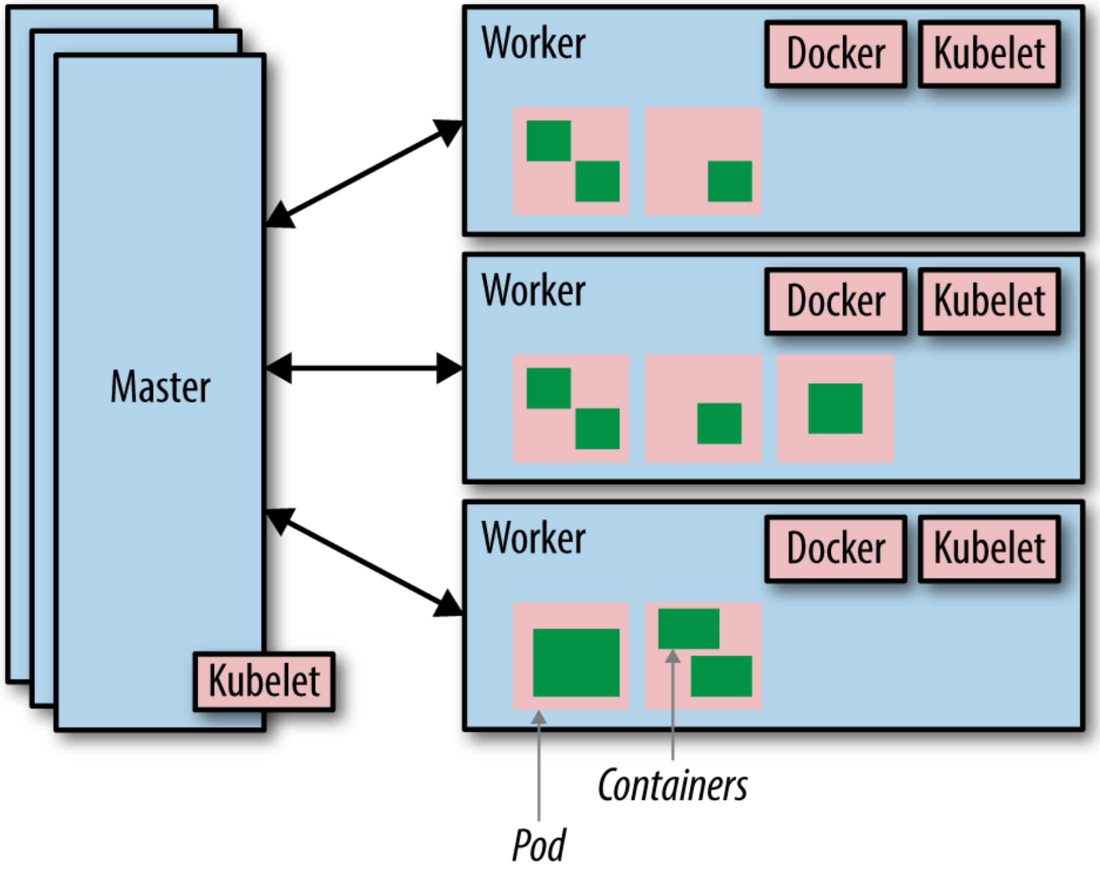


Cluster: A control plane and one or more compute machines, or worker nodes.

- **Control plane (master)**: The collection of processes that control k8s nodes. This is where all task assignments originate.
- **Worker nodes**: Worker nodes within the k8s cluster are used to run containerized applications

KUBERNETES ARCHITECTURE

https://enabling-cloud.github.io/kubernetes-learning/

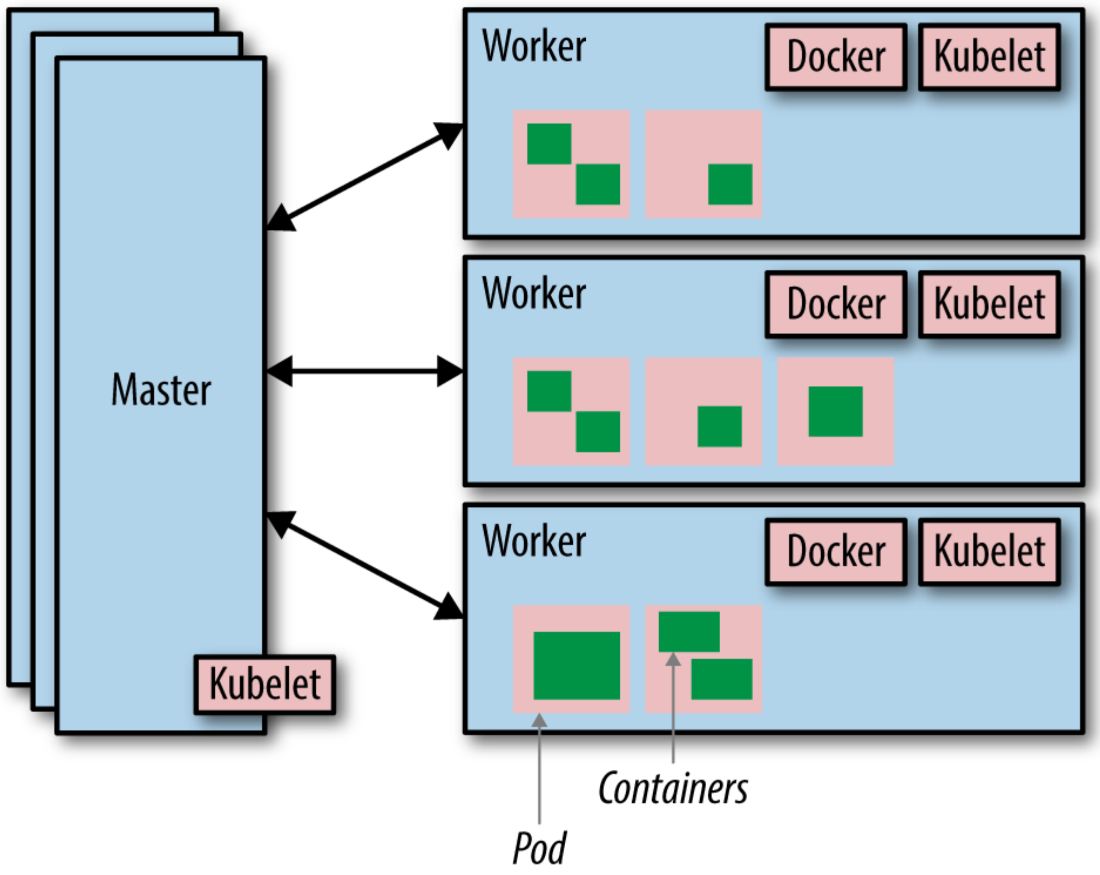


Worker node

- **Pod**: smallest deployable unit in k8s.
  - A pod hosts one or more containers
  - A pod provides shared storage and networking for its containers
- **Kubelet**: a system service (daemon) that runs on a worker node
  - Managing and monitoring the state of the pods and containers on that node.
  - Communicate with the master to receive instructions and report status updates.

KUBERNETES ARCHITECTURE

https://enabling-cloud.github.io/kubernetes-learning/



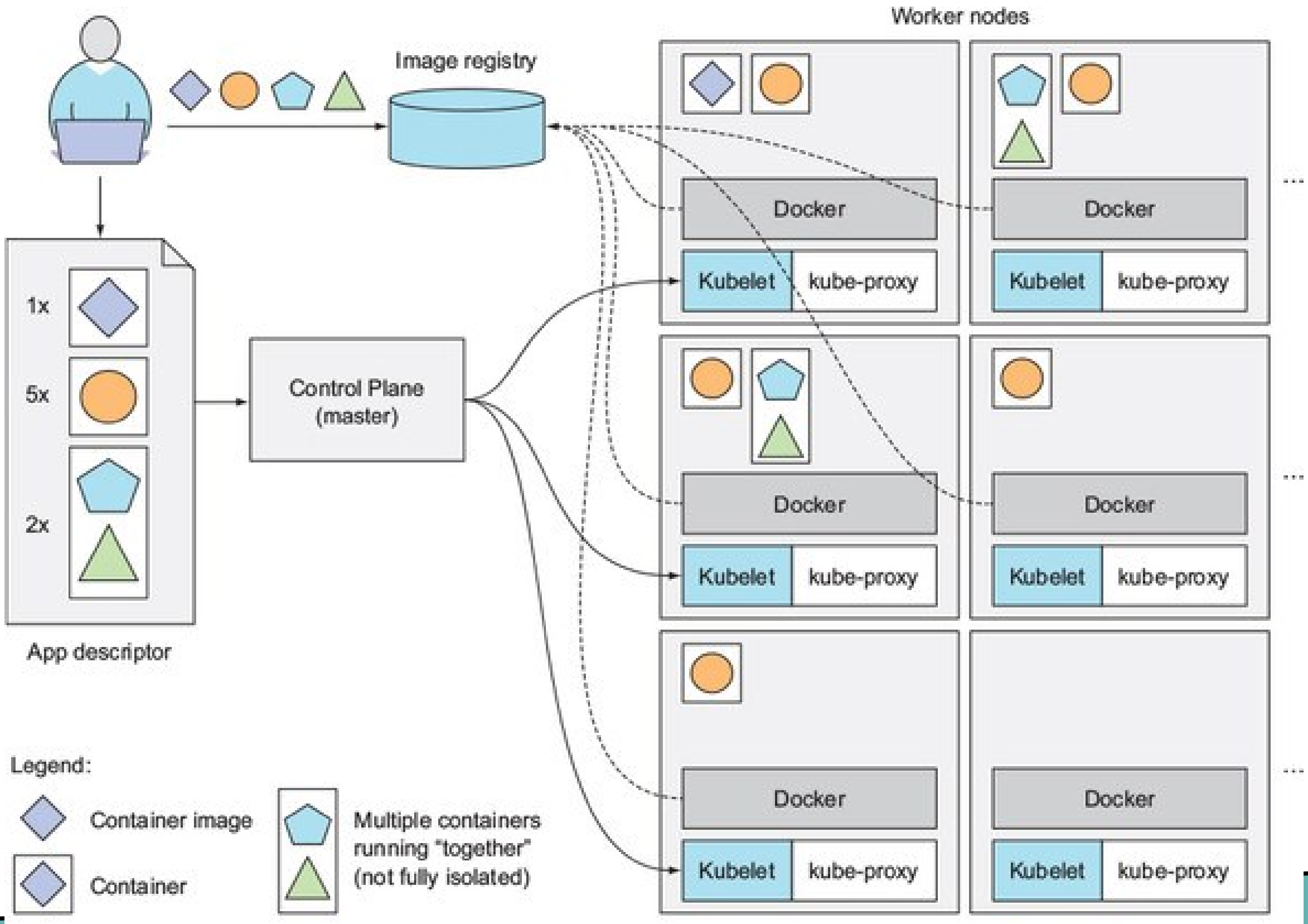
Worker node

- **Container runtime** (Docker): runs the containers on worker nodes
  - Pulling container images from registry
  - Starting and stopping containers
  - Managing container resources
- **Kube-proxy**: network proxy on workers
  - Routing traffic to the correct pods
  - Load balancing

KUBERNETES ARCHITECTURE

https://enabling-cloud.github.io/kubernetes-learning/

You tell k8s to run N instances of your service, and it handles the rest



KUBERNETES YAML FILE

- A k8s deployment YAML file is a configuration file written in YAML that defines the desired state of a Kubernetes Deployment.
- This YAML file is used to create, update, or delete Deployments in a k8s cluster.
- It contains a set of key-value pairs that specify various attributes and settings for the Deployment, such as the number of replicas, pod template specifications, labels, and more

Listing 12.4 Kubernetes `Deployment` for `ftgo-restaurant-service`

```
apiVersion: extensions/v1beta1
kind: Deployment
 metadata:
  name: ftgo-restaurant-service
 spec:
  replicas: 2
   template:
    metadata:
      labels:
        app: ftgo-restaurant-service
     spec:
       containers:
       - name: ftgo-restaurant-service
         image: msapatterns/ftgo-restaurant-service:latest
         imagePullPolicy: Always
         ports:
         - containerPort: 8080
           name: httpport
         env:
           - name: JAVA_OPTS
             value: "-Dsun.net.inetaddr.ttl=30"
           - name: SPRING_DATASOURCE_URL
             value: jdbc:mysql://ftgo-mysql/eventuate
           - name: SPRING_DATASOURCE_USERNAME
```

Specifies that this is an object of type Deployment

The name of the deployment

Number of pod replicas

Gives each pod a label called app whose value is ftgo-restaurant-service

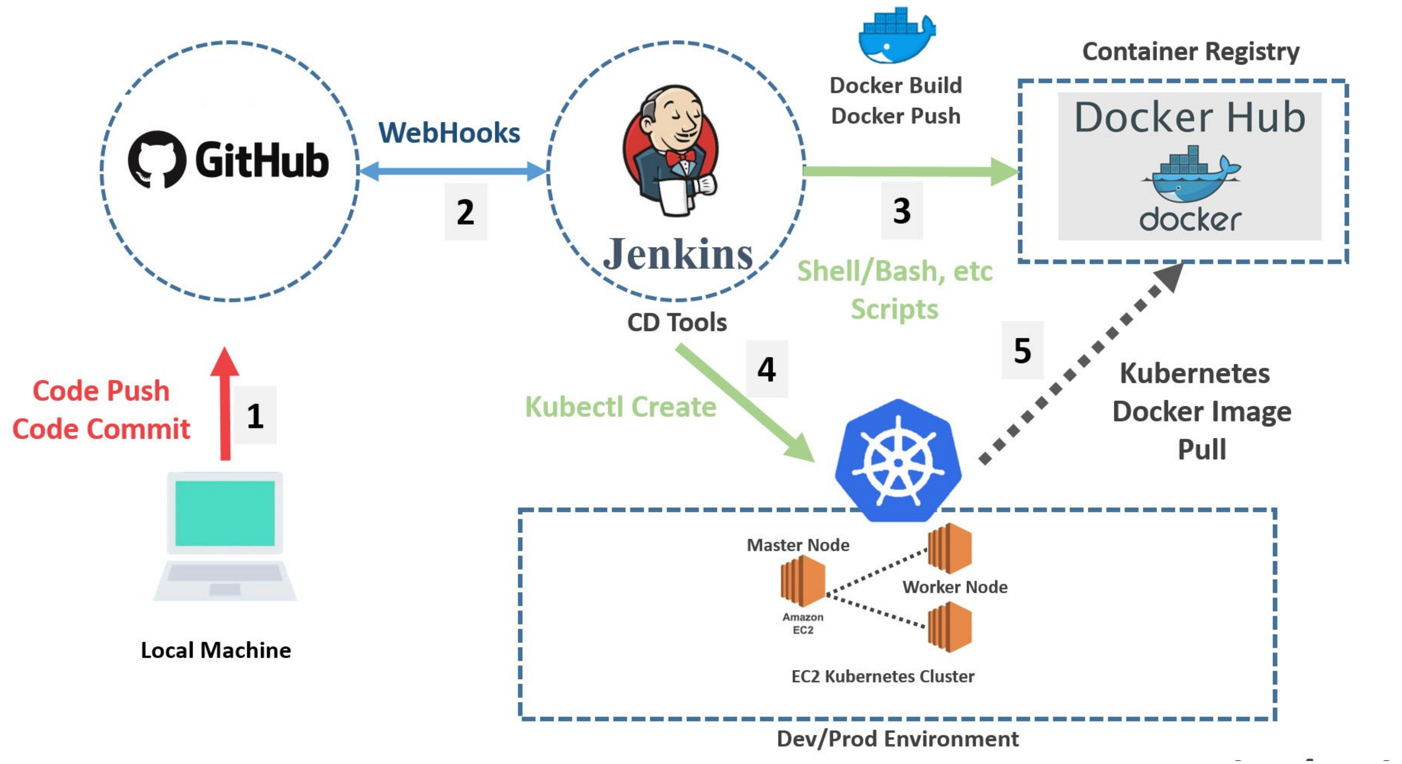
The specification of the pod, which defines just one container

The container's port

The container's environment variables, which are read by Spring Boot

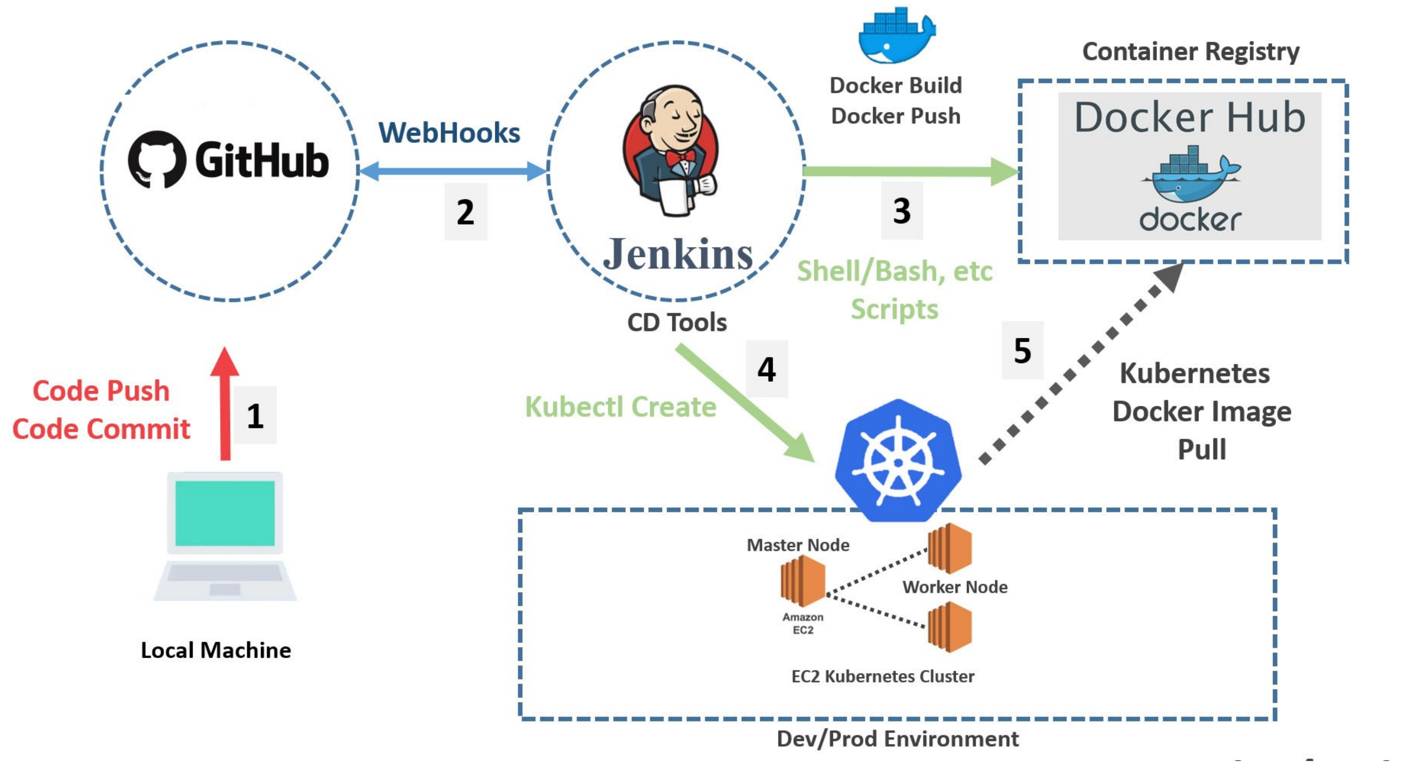
Chris Richardson. Microservices Pattern（微服务架构设计模式）. 机械工业出版社

A TYPICAL JENKINS PIPELINE



Step 0: Configure a **webhook** on GitHub, which allows external services like Jenkins to be notified when certain events like push happen.

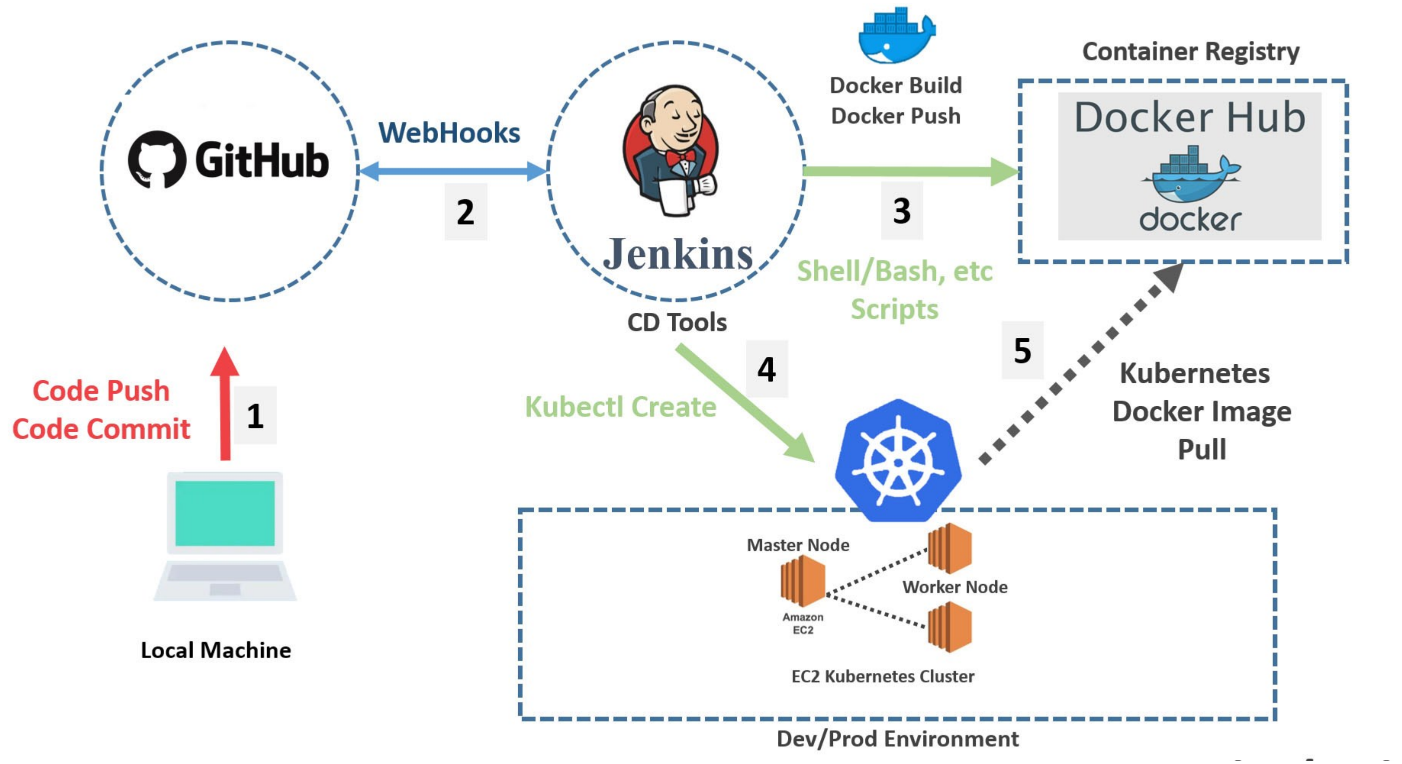
A TYPICAL JENKINS PIPELINE



Step 1: Dev team push local commits to GitHub

Step 2: The push triggers the webhook, and Jenkins will automatically download all the updates from GitHub

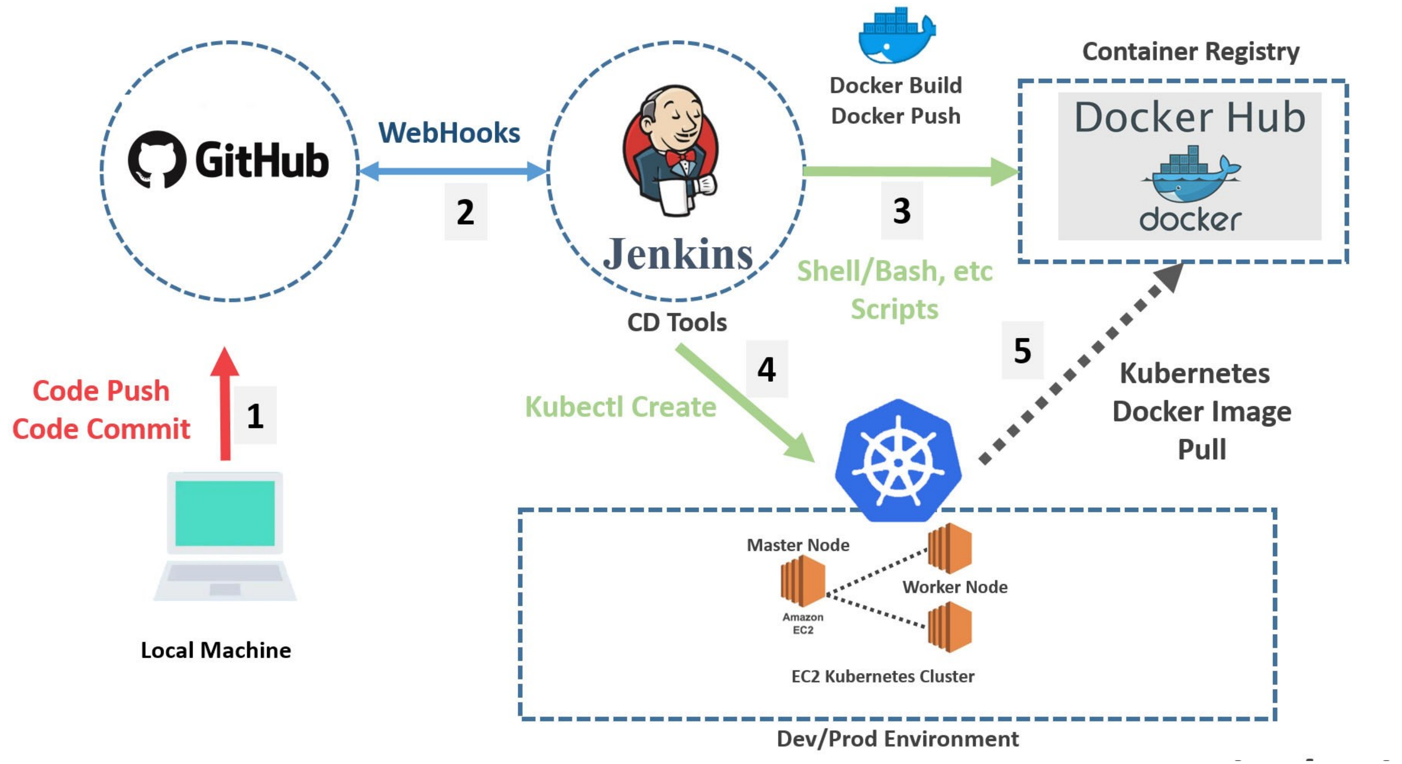
A TYPICAL JENKINS PIPELINE



Step 3: Jenkins could be configured to automatically execute a series of tasks/stages when triggered, for example:

- Build the project with Maven
- Test the project with JUnit
- Build the docker image
- Push the docker image to Docker Hub

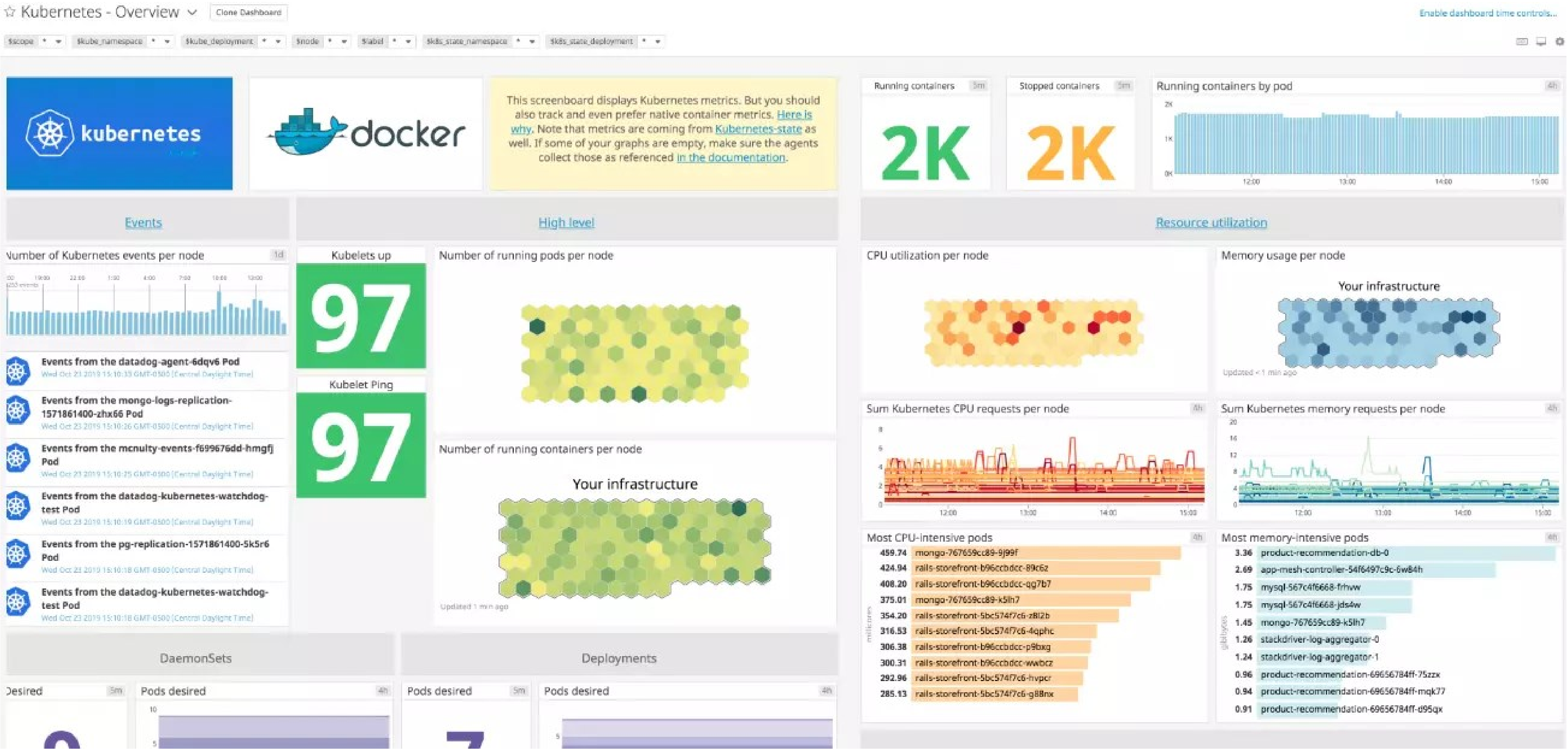
A TYPICAL JENKINS PIPELINE



Step 4: Jenkins could also be configured for automatic deployment, e.g., to deploy the software to a K8s cluster

Step 5: K8s pull docker images from the docker registry

MONITORING THE K8S CLUSTER

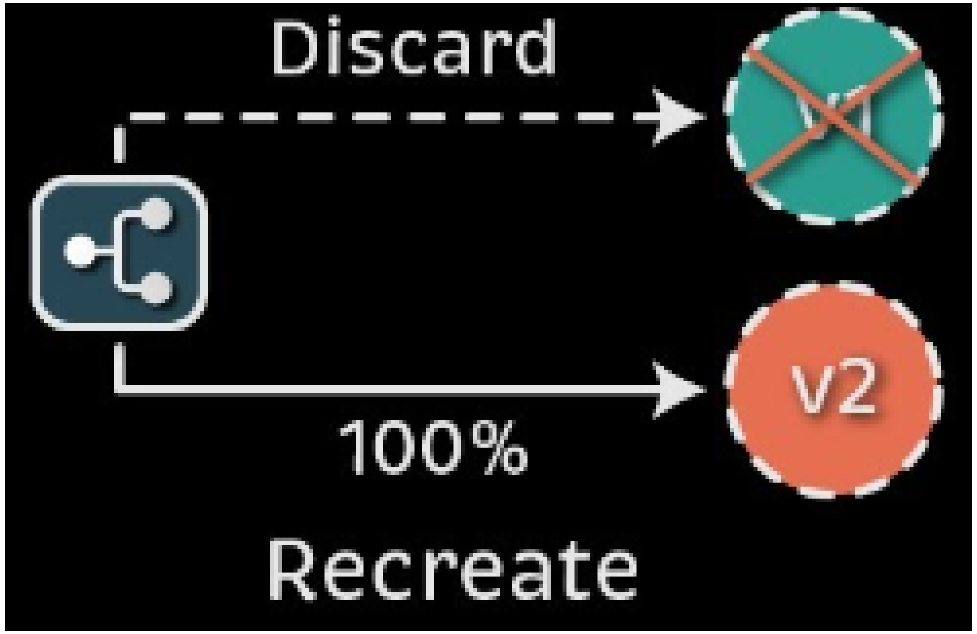


https://www.cloudzero.com/blog/kubernetes-monitoring

K8S DEPLOYMENT STRATEGY

Recreate: terminate all the running instances then recreate them with the newer version.

```yaml
spec:
  replicas: 3
  strategy:
    type: Recreate
```

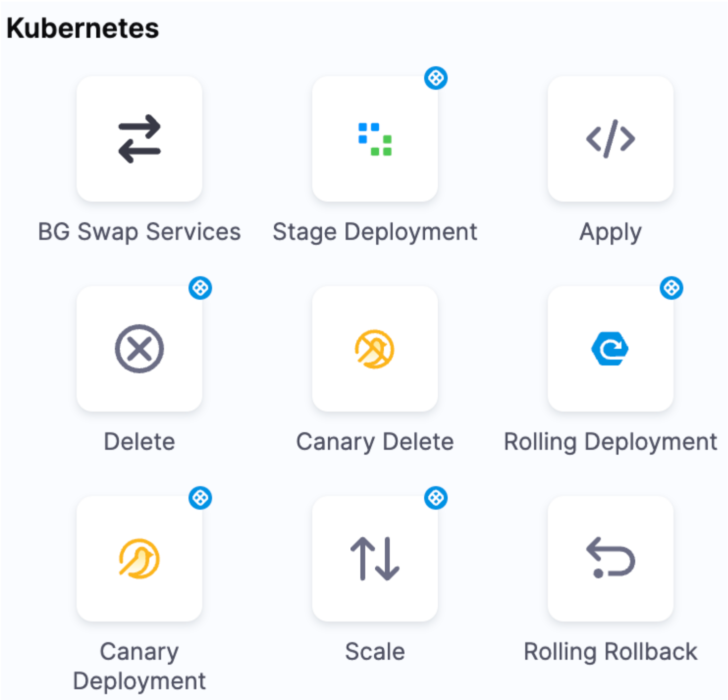


K8S DEPLOYMENT STRATEGY

Rolling strategy


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  namespace: default
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 25%
  replicas: 4
```

K8S DEPLOYMENT STRATEGY

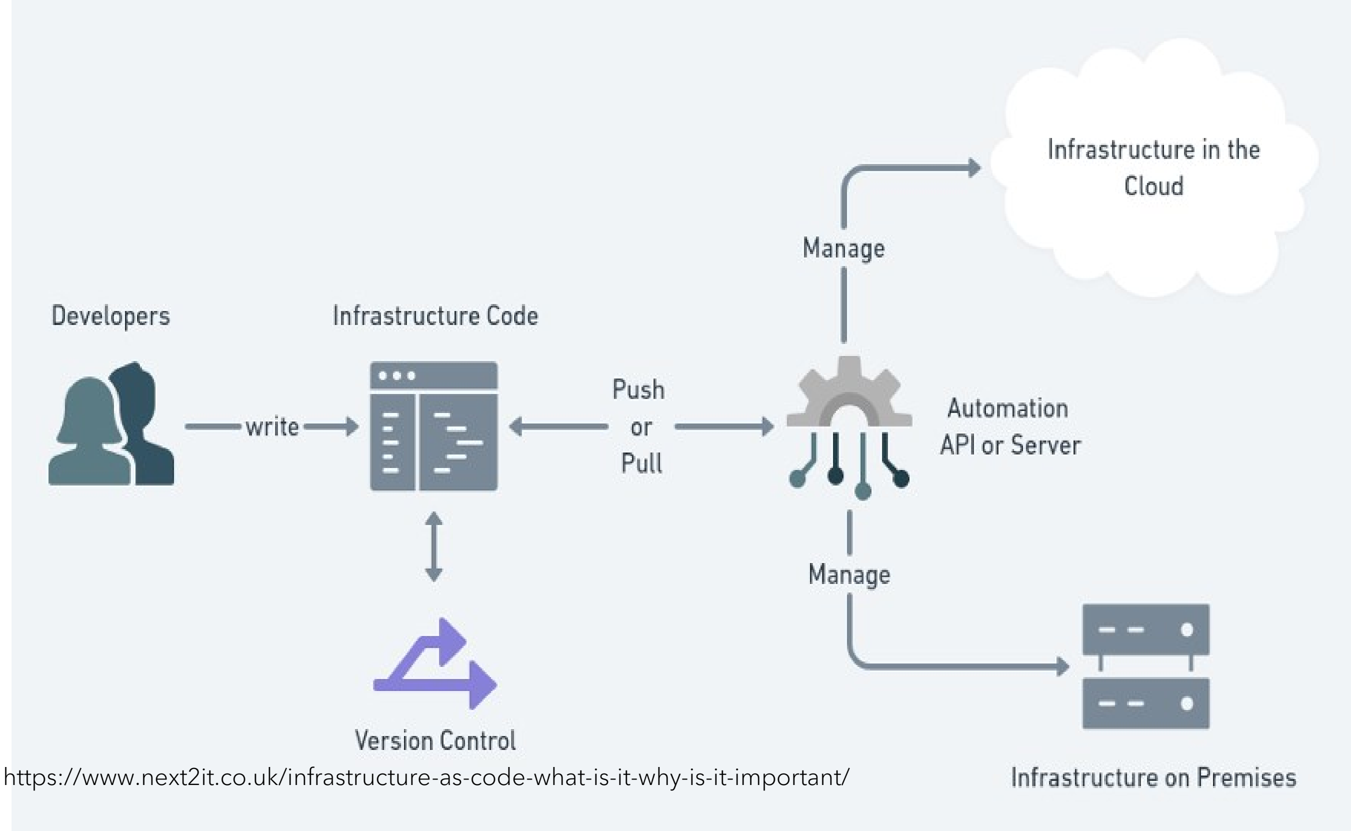


Deployment Strategy	Available in K8s out of the box?	Pros
Recreate	Yes	Fast and consistent.
Rolling	Yes	Minimizes downtime, provides security guarantees.
Blue/Green	No	No downtime and low risk, easy to switch traffic back to the current working version in case of issues.
Canary	No	Seamless to users, makes it possible to evaluate a new version and get user inputs with low risk.

https://codefresh.io/learn/kubernetes-deployment/top-6-kubernetes-deployment-strategies-and-how-to-choose/

INFRASTRUCTURE AS CODE (基础设施即代码)

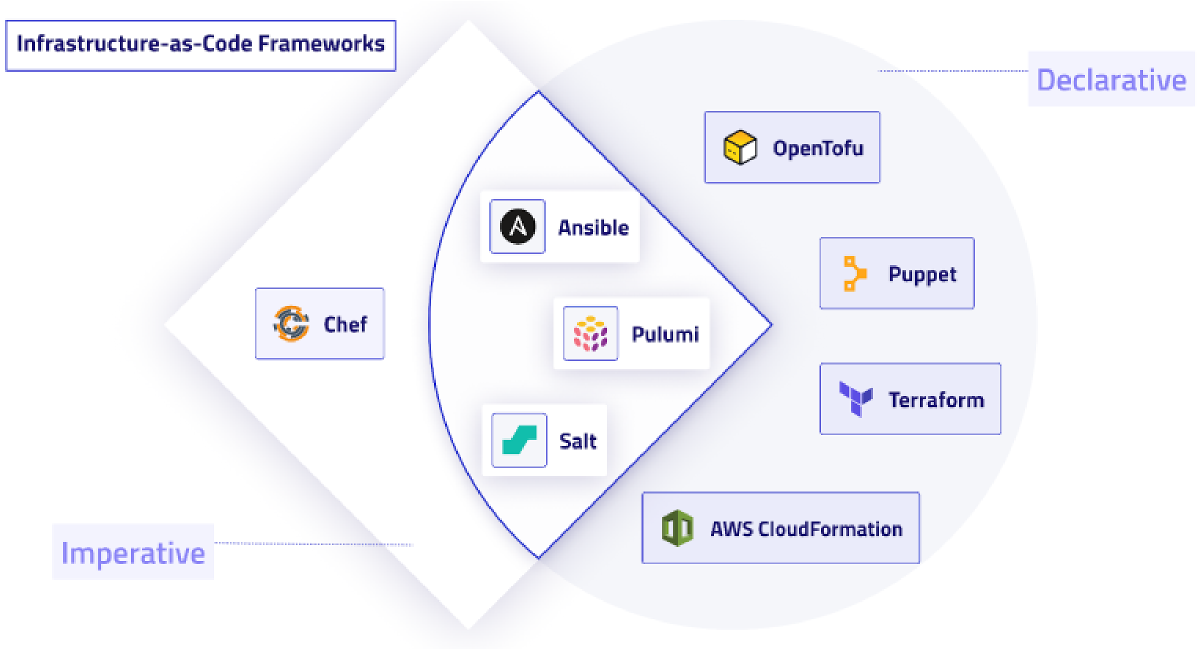
- Infrastructure as code (IaC) is DevOps practice that involves defining and managing infrastructure resources such as servers, networks, storage **using code instead of manual processes**.
- IaC code is stored in version control systems such as Git, and then executed using automation tools.
- Teams can define and maintain their infrastructure as code, just like software applications, and apply best practices such as testing, and CI/CD.



https://www.next2it.co.uk/infrastructure-as-code-what-is-it-why-is-it-important/

INFRASTRUCTURE AS CODE: APPROACHES

- **Declarative approach**: focuses on what the eventual target configuration should be. The declarative approach defines the desired state and the system executes what needs to happen to achieve that desired state.
- **Imperative approach**: Imperative defines specific commands that need to be executed in the appropriate order to end with the desired state; focuses on how the infrastructure is to be changed.



https://www.env0.com/blog/infrastructure-as-code-101

INFRASTRUCTURE AS CODE

Configurations

- **Environment config:** IP, port, OS, version, etc.
- **Application config:** versions and configs for dependent software or services, caches, token, etc.
- **Business config:** discount, feature toggles, etc.

Benefits

- **Scale and agility:** automation
- **Self-documenting**
- **Auditability**: everything is trackable

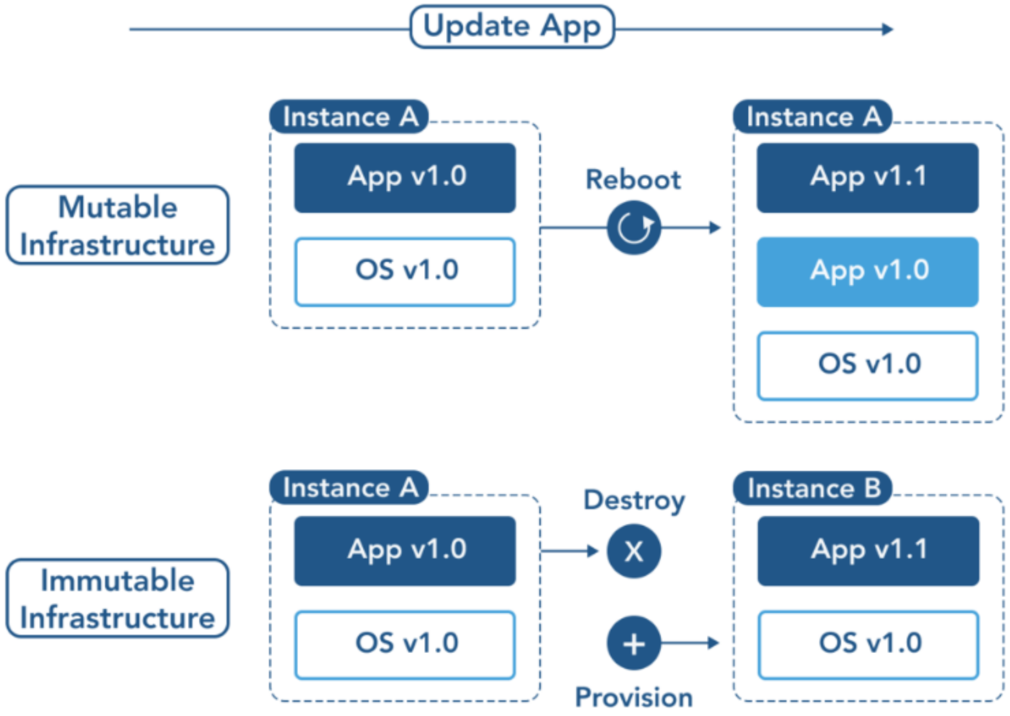
Ansible config code example

```yaml
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest

  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest

  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

IMMUTABLE INFRASTRUCTURE (不可变基础设施):

- Once a specific instance (such as a container or virtual machine) is started, its configuration should never change.
- Instead of upgrading or re-configuring the underlying infrastructure of that instance, you should simply replace it entirely with a new instance that has all of your required changes.

https://www.opsramp.com/guides/why-kubernetes/infrastructure-as-code/

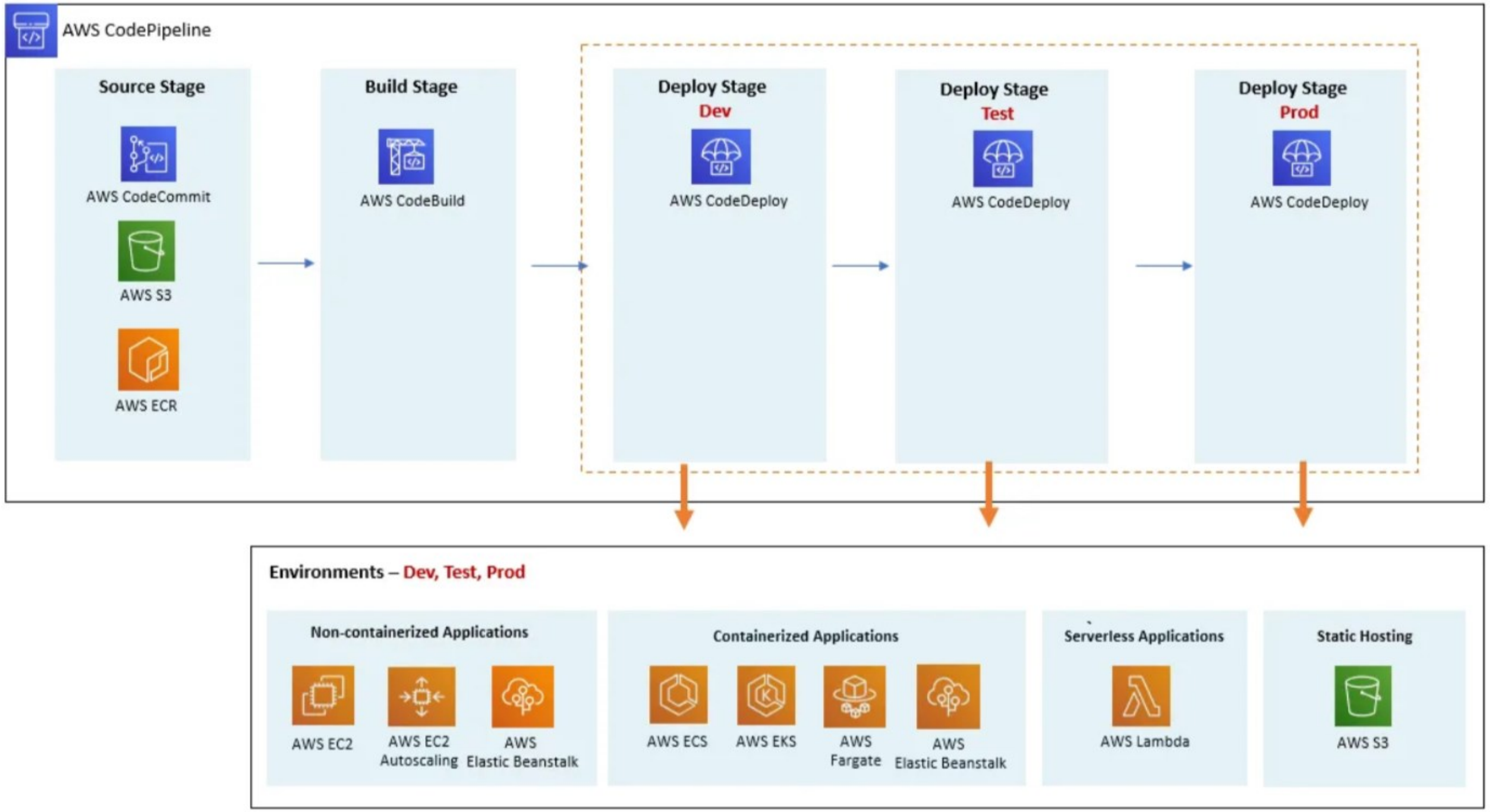


WHAT IS CLOUD-NATIVE (云原生)?

By 2024, 50% of Organizations Will Use Applications Built on Cloud-Native Technologies to Enable Consistency in Running in Any and Many Locations

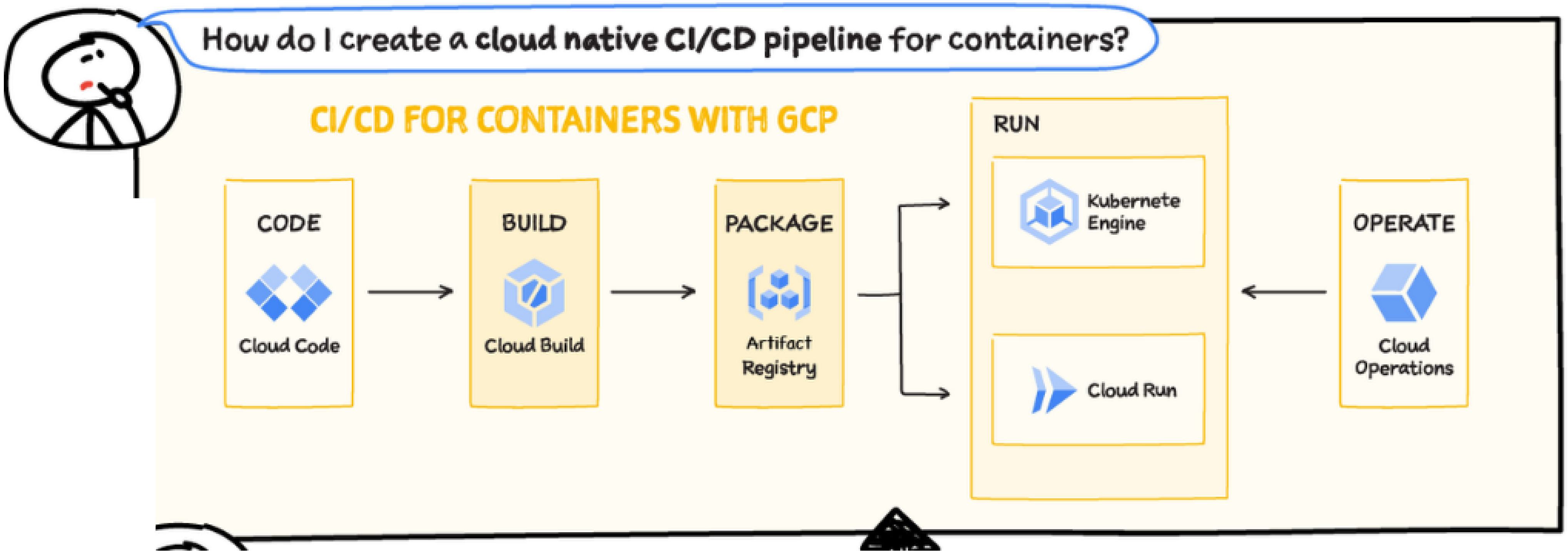
-- IDC FutureScape: Worldwide Cloud 2022 Predictions

CLOUD-NATIVE CI/CD PIPELINE ON AWS



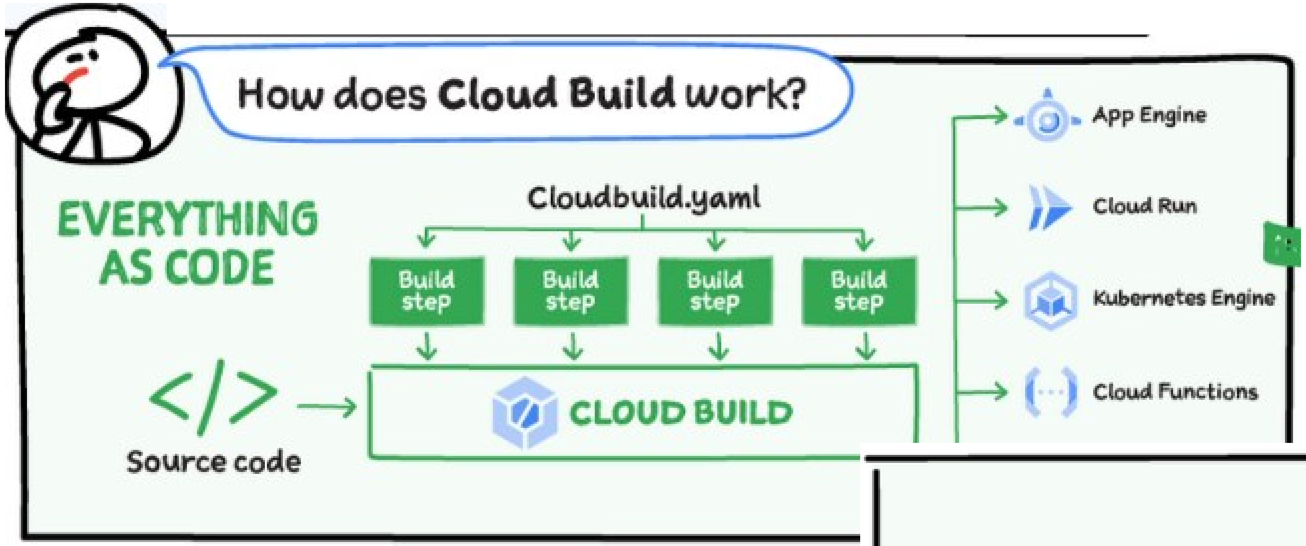
https://medium.com/humans-of-devops/enabling-cloud-native-ci-cd-workflows-in-aws-d424a43b0500

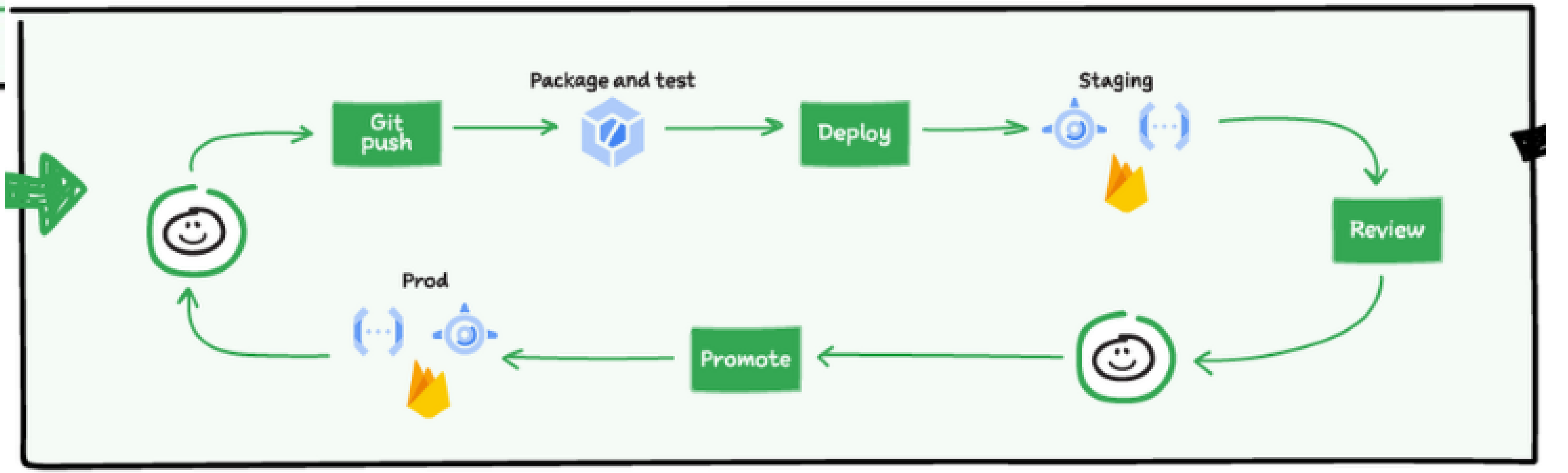
CLOUD-NATIVE CI/CD PIPELINE ON GOOGLE CLOUD



https://cloud.google.com/blog/topics/developers-practitioners/devops-and-cicd-google-cloud-explained

CLOUD-NATIVE CI/CD PIPELINE ON GOOGLE CLOUD

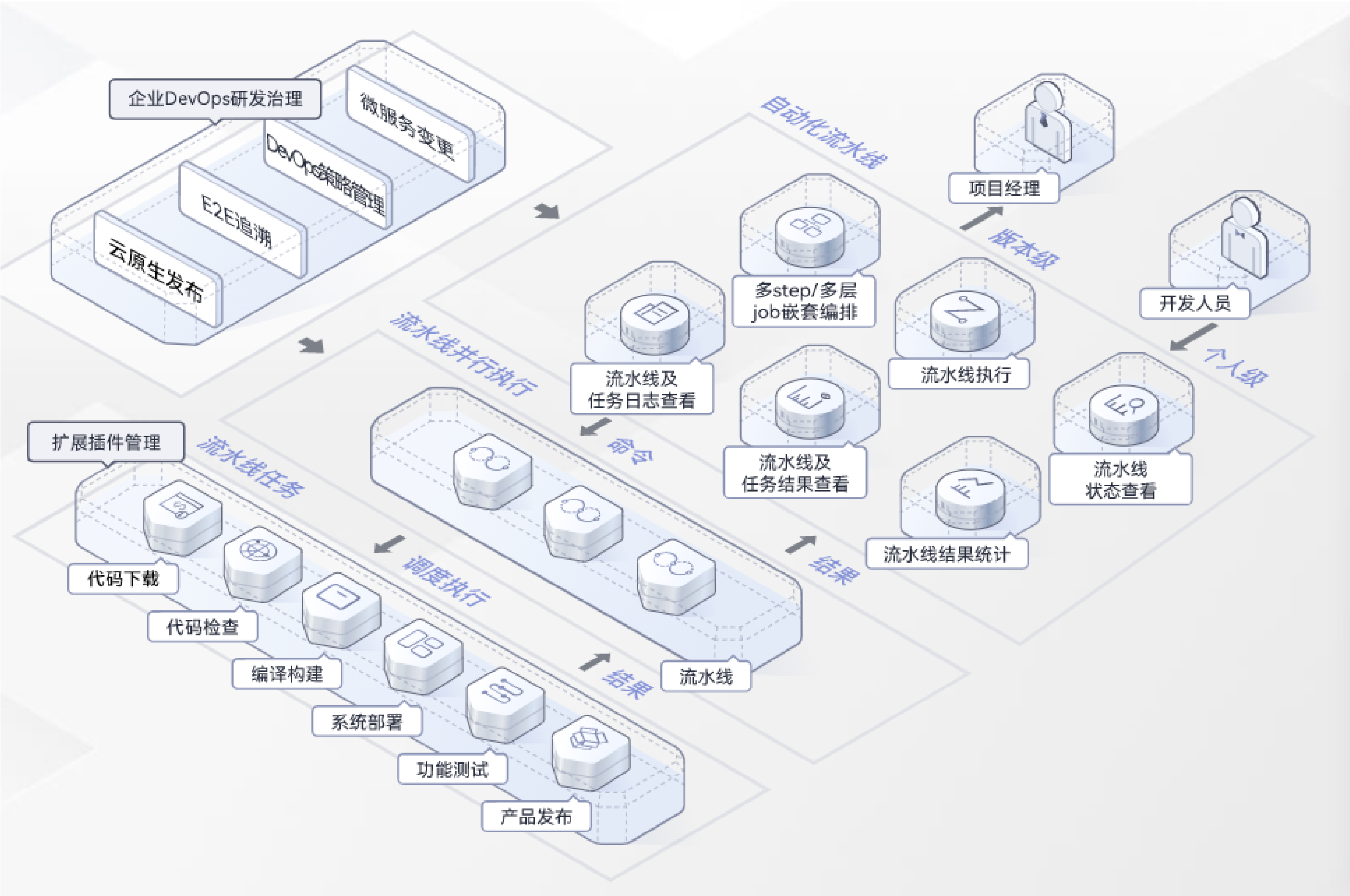




https://cloud.google.com/blog/topics/developers-practitioners/devops-and-cicd-google-cloud-explained

华为DevCloud/CodeArts

https://www.huaweicloud.com/product/cloudpipeline.html



服务优势

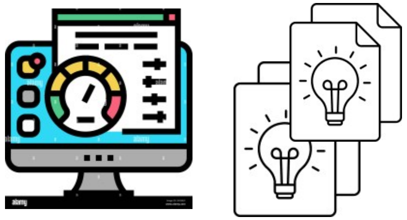
- **灵活编排、高效调度**
  - 多step/多层job嵌套编排，代码事件、定时、手工、变更、子流水线等灵活的执行策略
  - 百万级任务并发执行，满足大规模构建、代码检查、测试并发执行要求
- **开放流水线插件**
  - 架构可扩展，支持第三方插件快速集成
  - 可视化插件研发，支持低代码生成UI
- **内置企业DevOps研发治理模型**
  - 基于微服务DevOps变更模式，特性可按需发布
  - 蓝绿升级/滚动升级等云原生发布管理，版本可一键回滚
  - 全流程E2E可追溯
  - DevOps研发策略治理模型

SUMMARY

	Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
~ 1980 ~ 1990	Waterfall	Monolithic	Physical Server	Datacenter
~ 2000	Agile	N-Tie	Virtual Servers	Hosted
~ 2010	DevOps	Microservices	Containers	Cloud

OUR TEAM PROJECT

1. Metrics
2. Documentation
3. Tests
4. Build
5. Deploy
6. Demo



Week 1
Team up

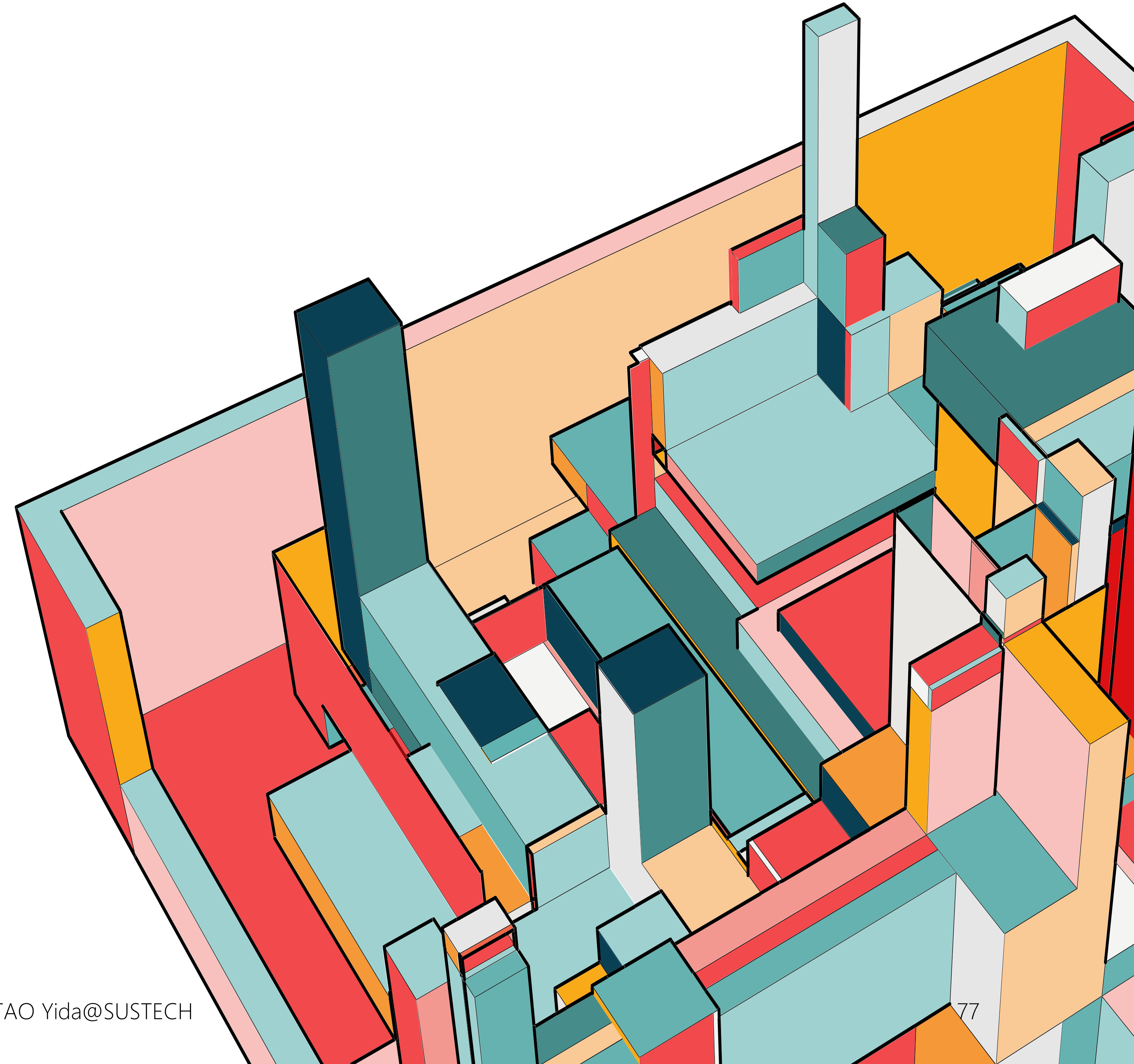
Week 5
Proposal

Week 9
Sprint 1

Week 16
Sprint 2

READINGS

- Microservice Patterns with Examples in Java. Chris Richardson.
- Chapter 23-24. Software Engineering at Google by Winters et al.
- 第7、10章 软件体系结构. 现代软件工程基础 by 彭鑫 et al.



NEXT

- Software maintenance
- Software evolution