CompSci 401: Cloud Computing

What Is Orchestration?

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Full automation of a service

- Many tasks related to offering a service

Full automation of a service

- Many tasks related to offering a service
  - Deployment

Full automation of a service

- Many tasks related to offering a service
  - Deployment
  - Configuration

Full automation of a service

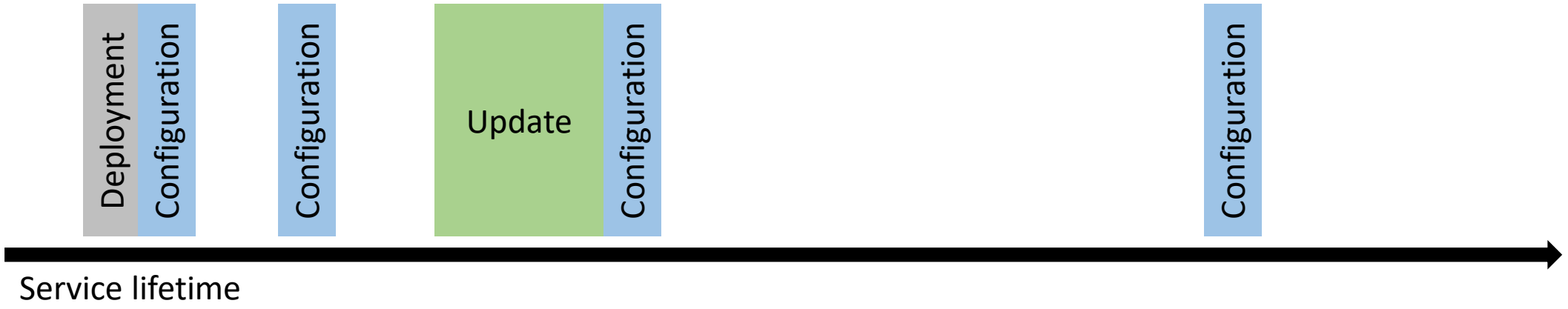
- Many tasks related to offering a service
  - Deployment
  - Configuration

Full automation of a service

- Many tasks related to offering a service
  - Deployment
  - Configuration
  - Updates

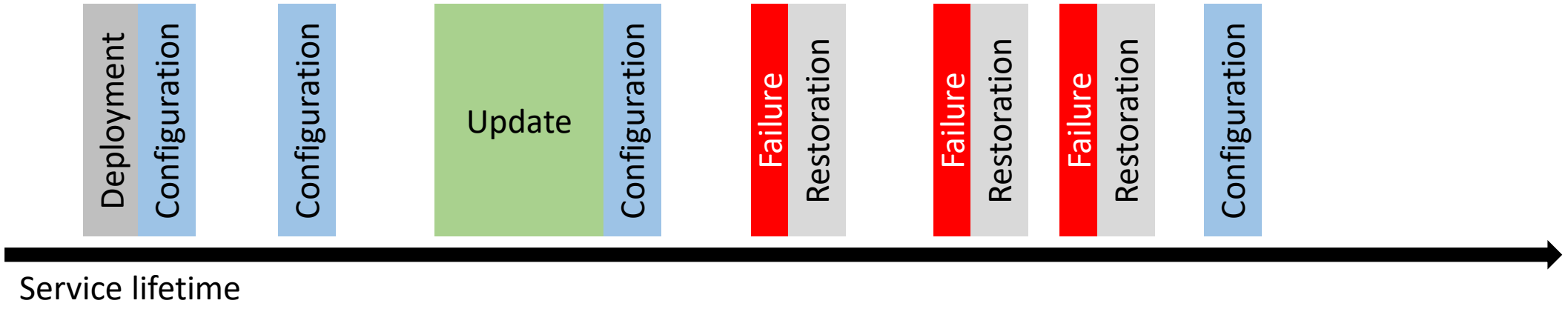
Full automation of a service

- Many tasks related to offering a service
  - Deployment
  - Configuration
  - Updates



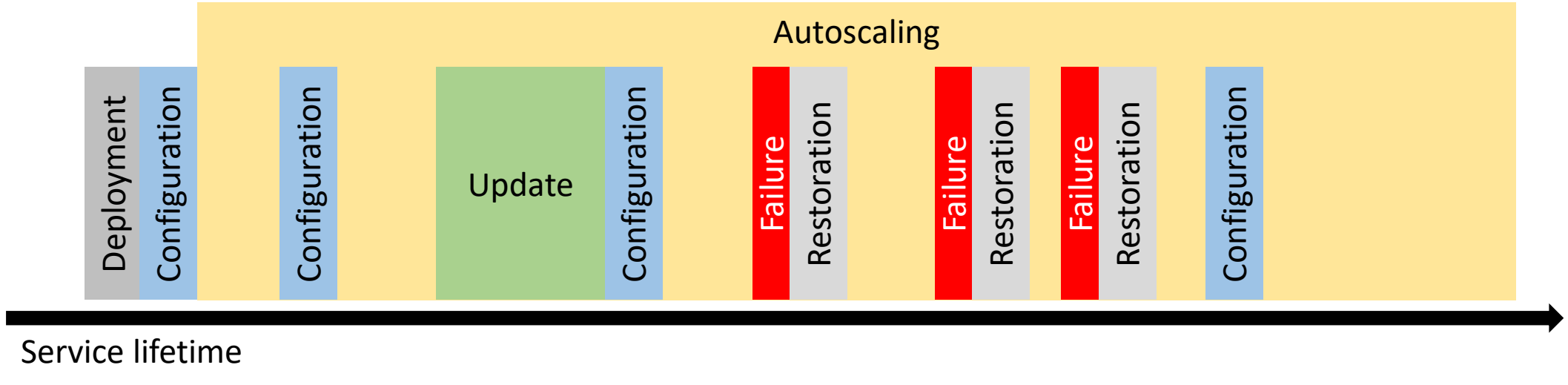
Full automation of a service

- Many tasks related to offering a service
  - Deployment
  - Configuration
  - Updates
  - Failure resilience



Full automation of a service

- Many tasks related to offering a service
  - Deployment
  - Configuration
  - Updates
  - Failure resilience
  - Autoscaling



Specify the building blocks of a service

- Orchestration assembles the building blocks
  - Support for diverse goals

Specify the building blocks of a service

- Orchestration assembles the building blocks
  - Support for diverse goals

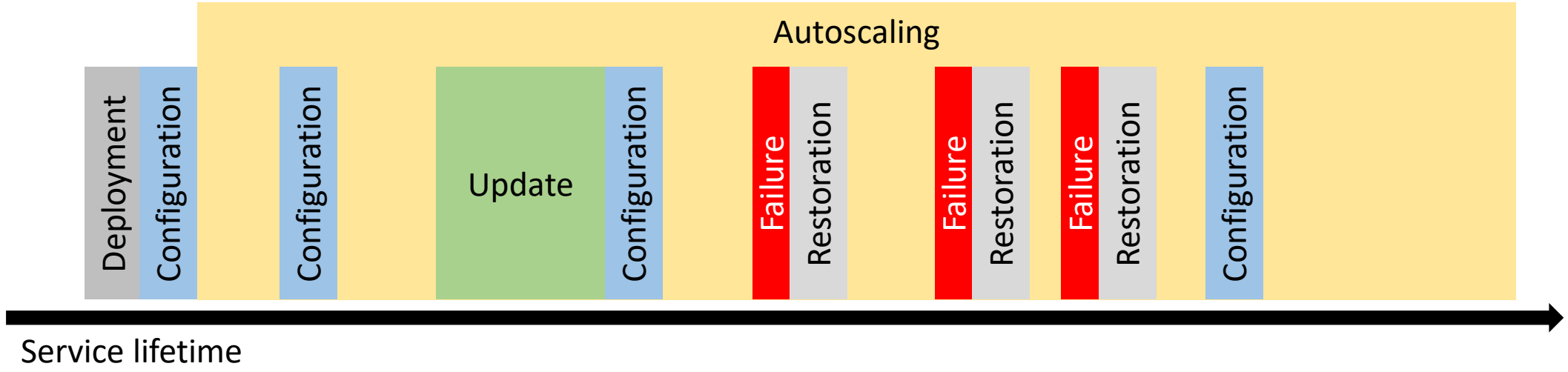
- Blocks: Map and Reduce
  - Orchestrator: Hadoop and Yarn
- Blocks: Containers
  - Orchestrator: Kubernetes
- Blocks: Virtual machines
  - Orchestrator: OpenStack

Properties of building blocks

- Ephemeral: Blocks are not permanent
  - Rapid creation
  - Short lifetime
- Can be replicated if more blocks are needed
- Can be replaced with other instances of that block
- Can be combined with other blocks

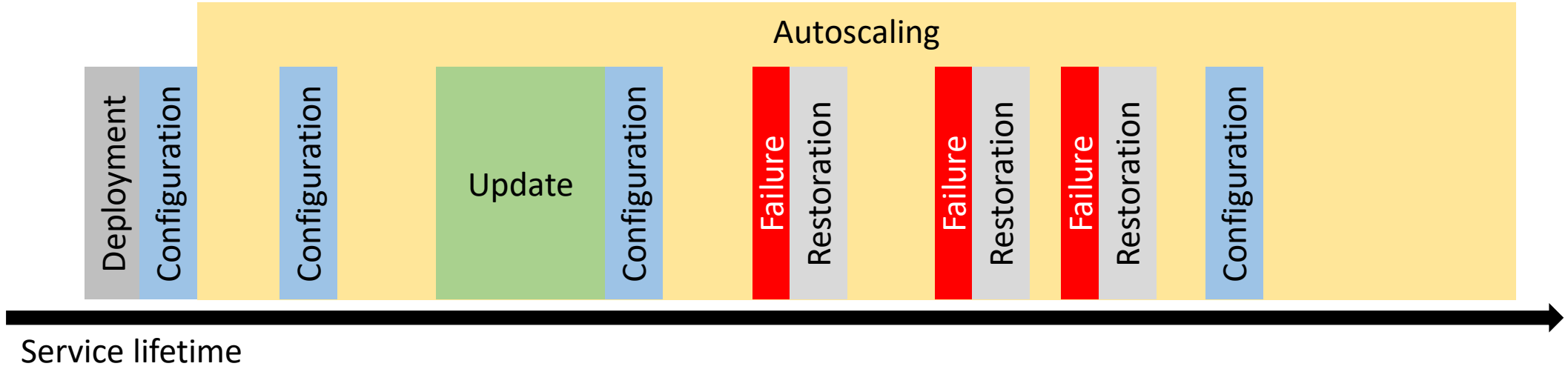
Properties of building blocks

- Ephemeral: Blocks are not permanent → Deployment, updates
- Can be replicated if more blocks are needed → Autoscaling
- Can be replaced with other instances of that block → Restoration
- Can be combined with other blocks



Properties of building blocks

- Ephemeral: Blocks are not permanent → Deployment, updates
- Can be replicated if more blocks are needed → Autoscaling
- Can be replaced with other instances of that block → Restoration
- Can be combined with other blocks → Complex services



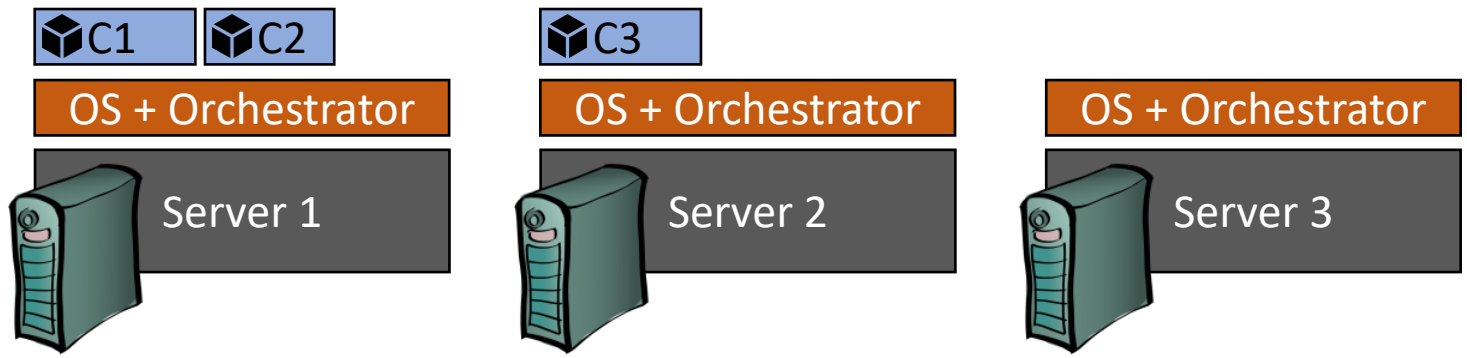
CompSci 401: Cloud Computing

Orchestrator Functionalities

Prof. Ítalo Cunha

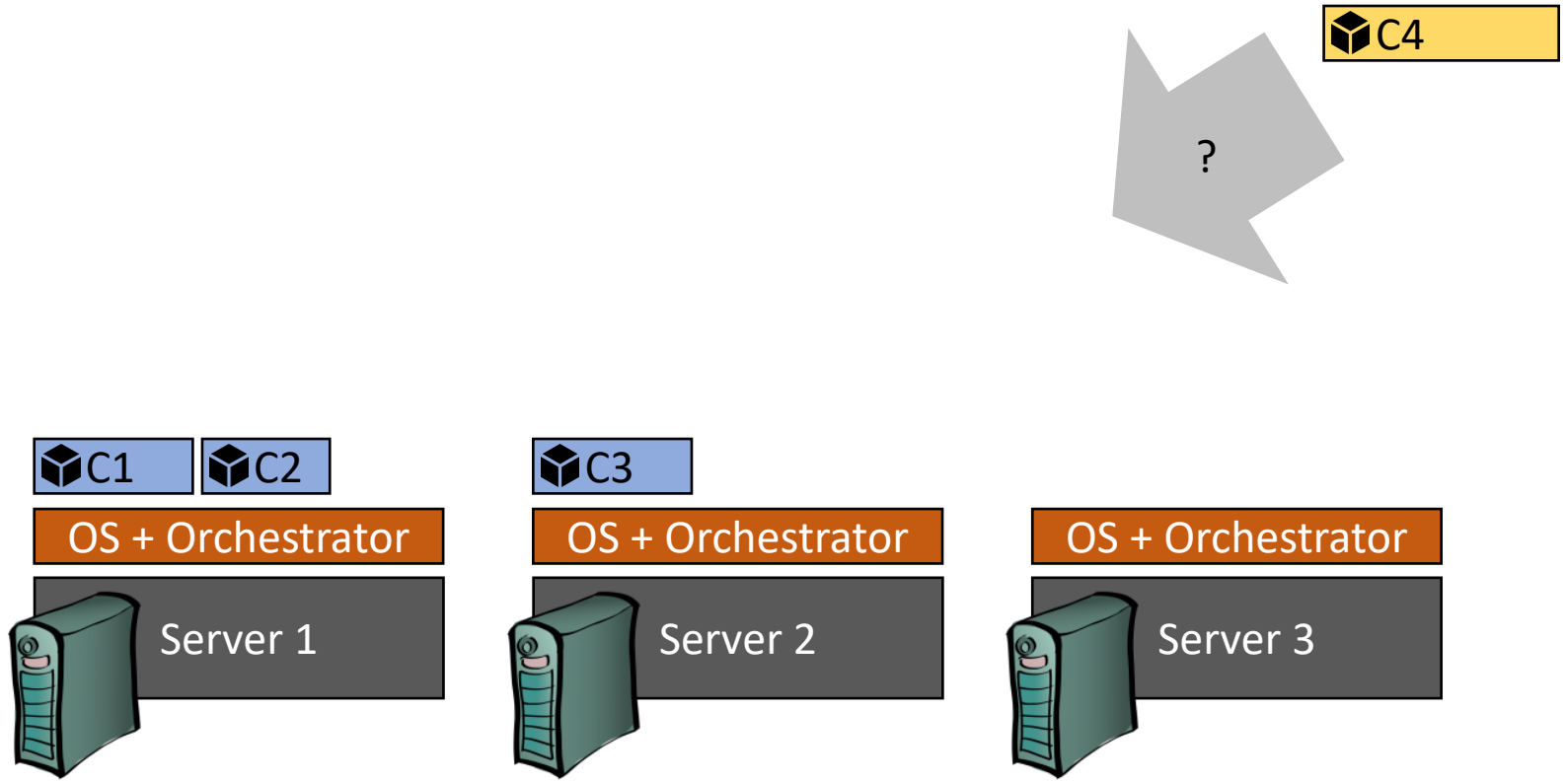
昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Optimized container placement



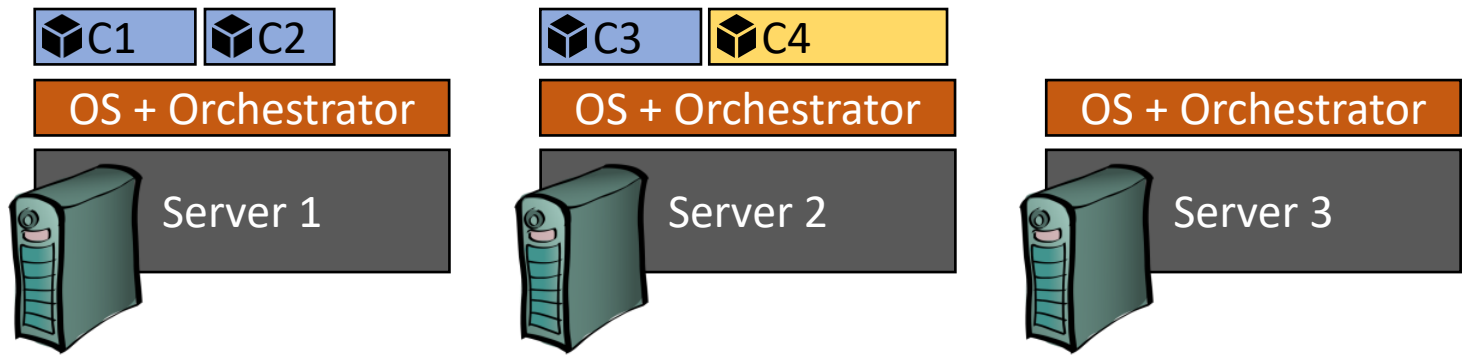
Optimized container placement

- Where to place container C4?



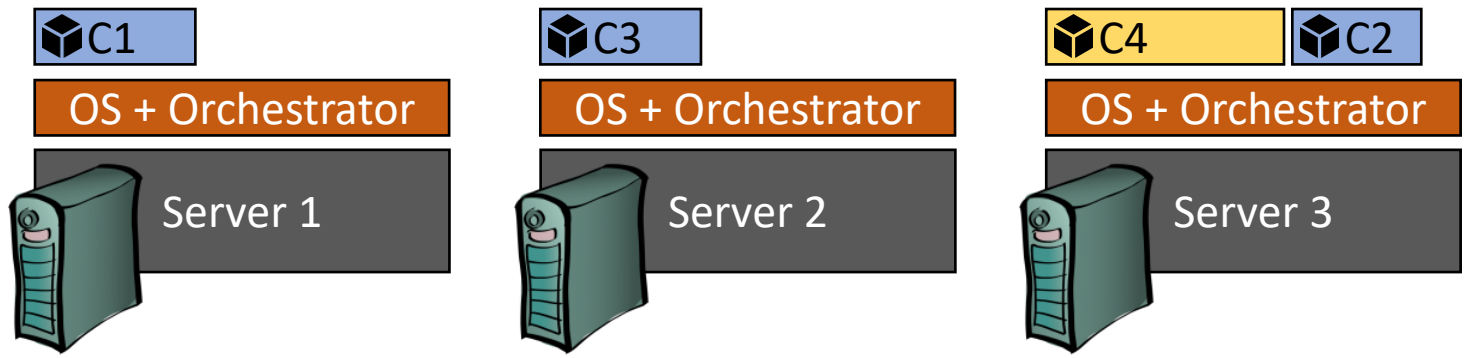
Optimized container placement

- Where to place container C4?
- Maximize server utilization
  - Place on server S2



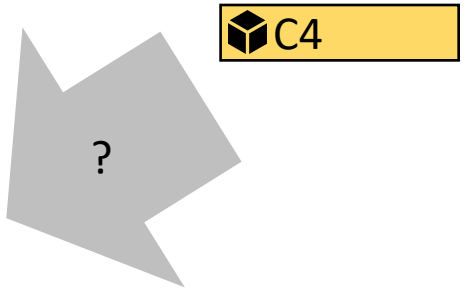
Optimized container placement

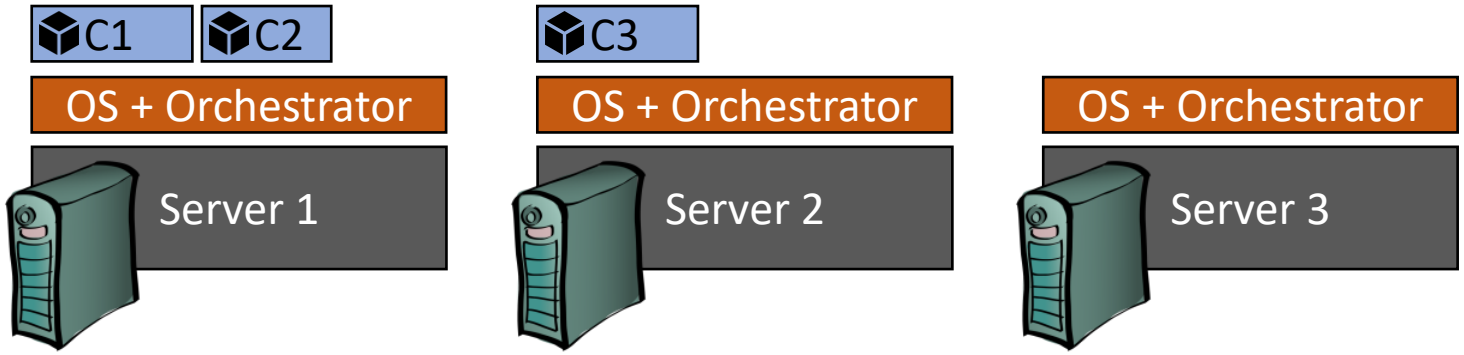
- Where to place container C4?
- Maximize server utilization
  - Place on server S2
- Make network traffic local
  - If C4 will talk to C2, co-locate on Server 3



Optimized container placement

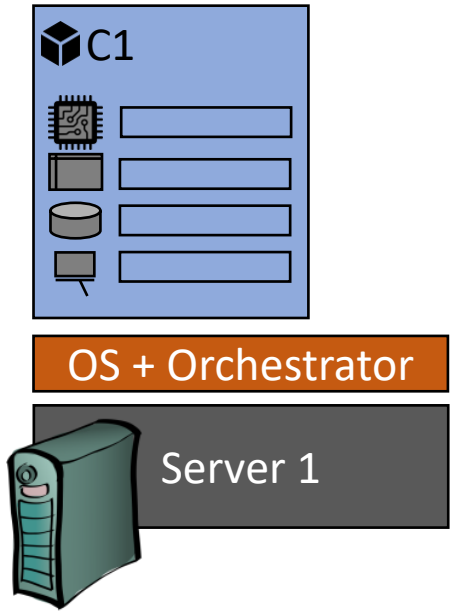
- Where to place container C4?
- Maximize server utilization
  - Place on server S2
- Make network traffic local
  - If C4 will talk to C2, co-locate on Server 3
- Satisfy container-specific demands





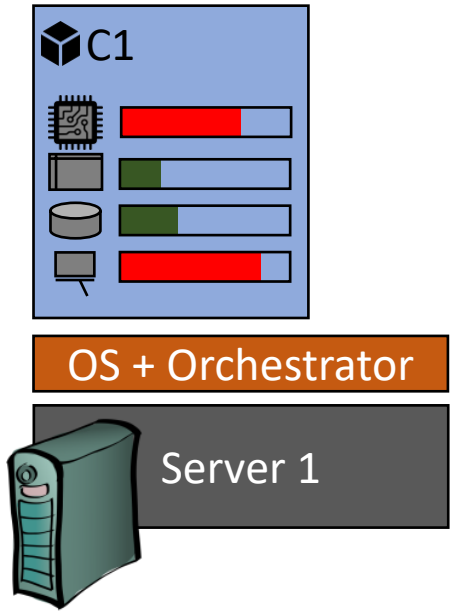
Resource utilization monitoring

- Developers and customers specify estimated resource use



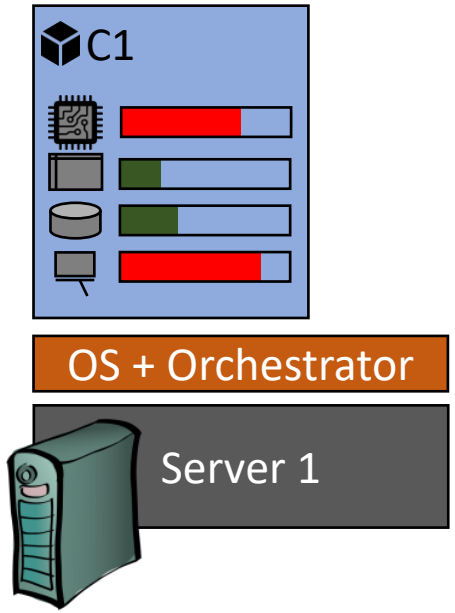
Resource utilization monitoring

- Developers and customers specify estimated resource use
- Orchestrators monitor resource use

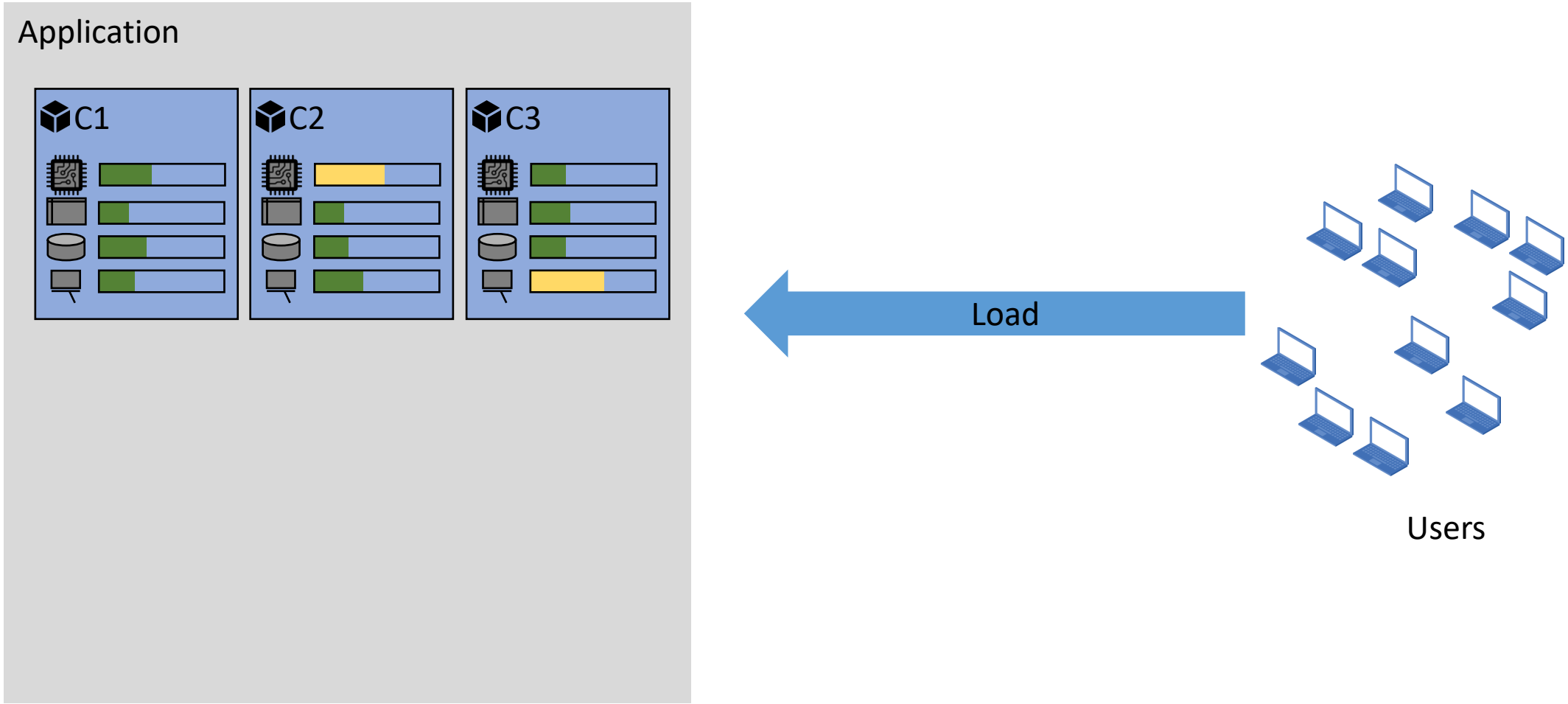


Resource utilization monitoring

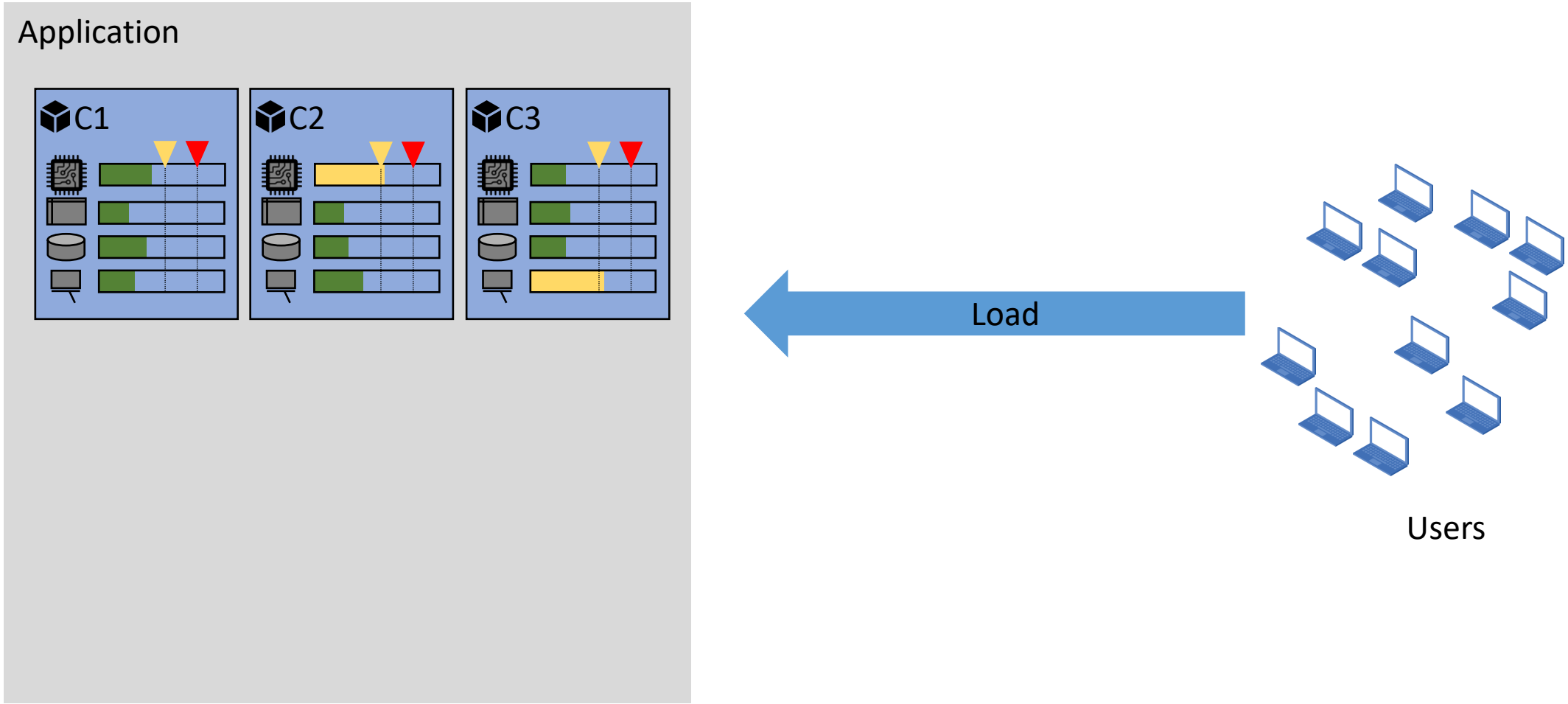
- Developers and customers specify estimated resource use
- Orchestrators monitor resource use
- Many different resources can be tracked



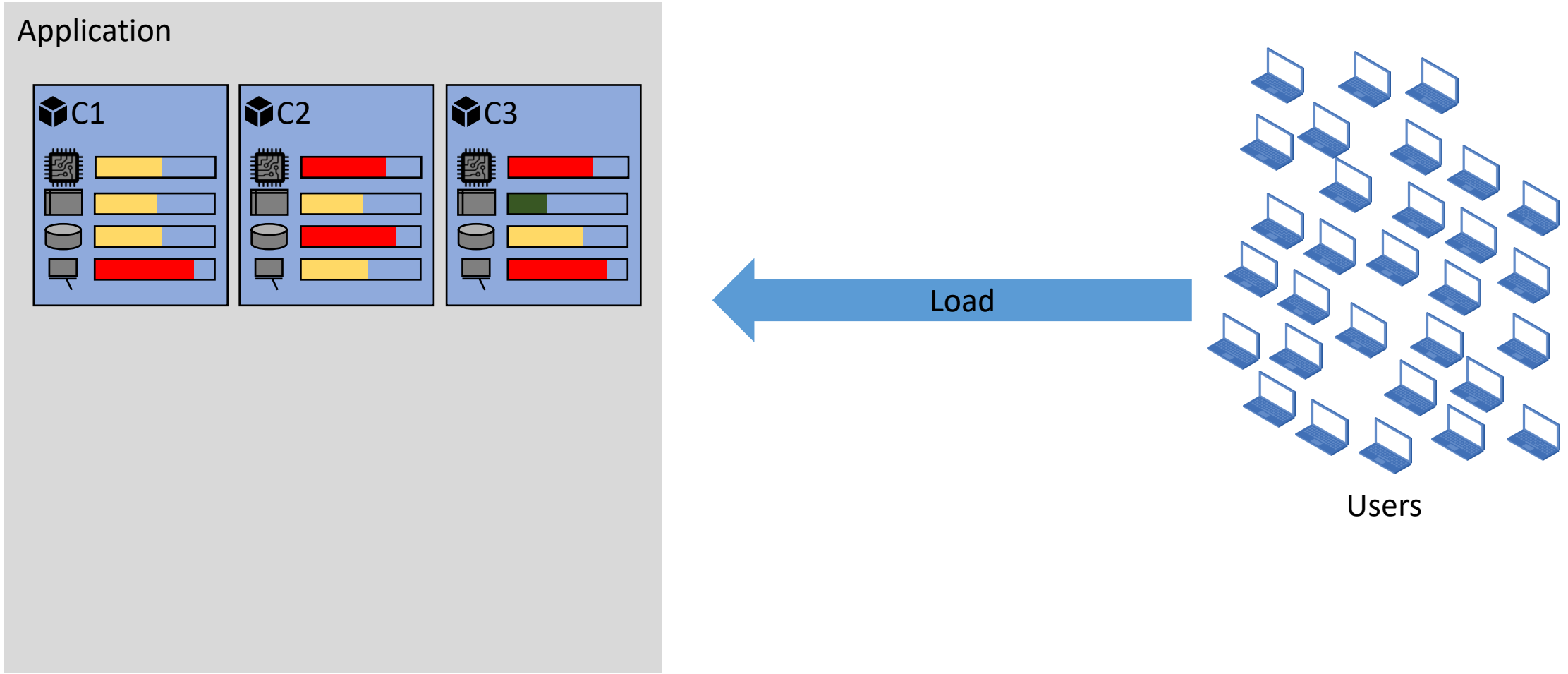
Autoscaling



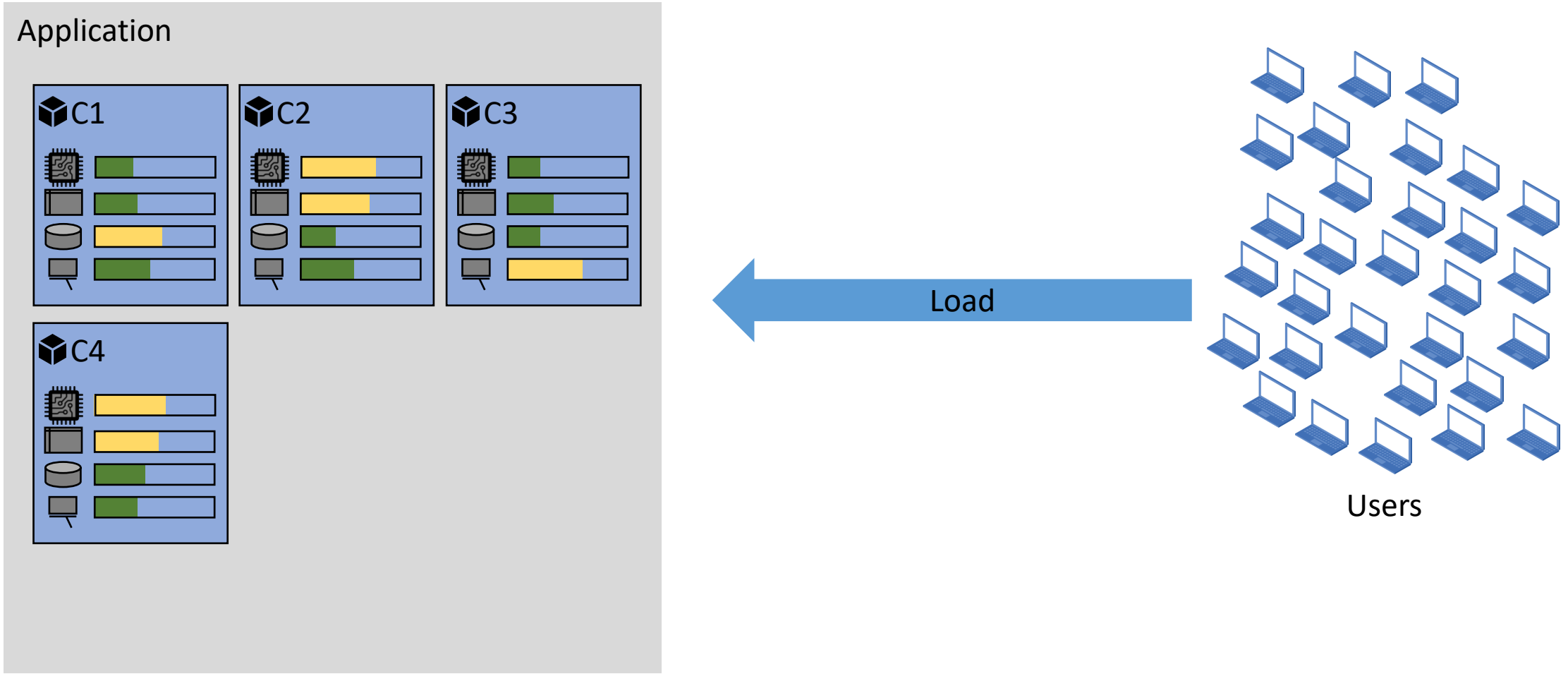
Autoscaling



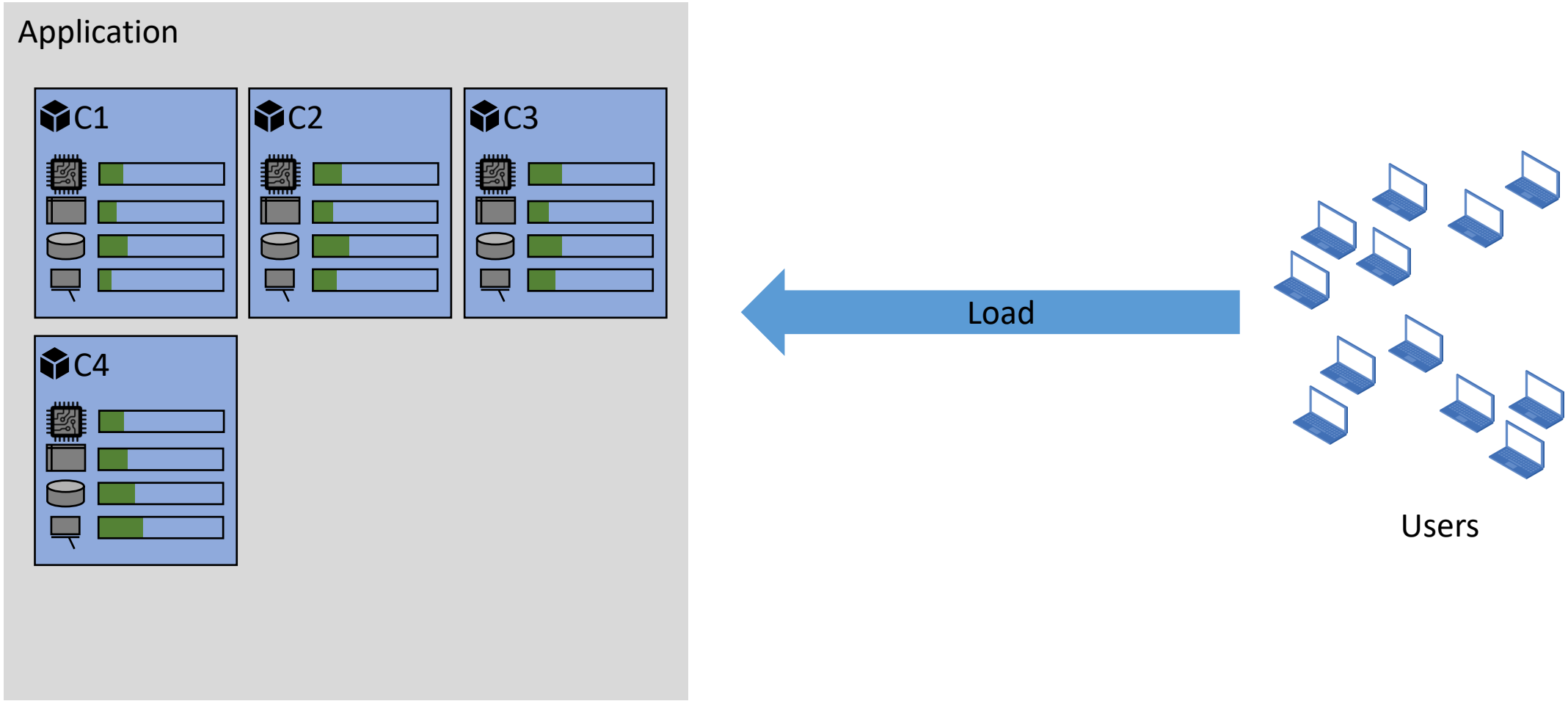
Autoscaling



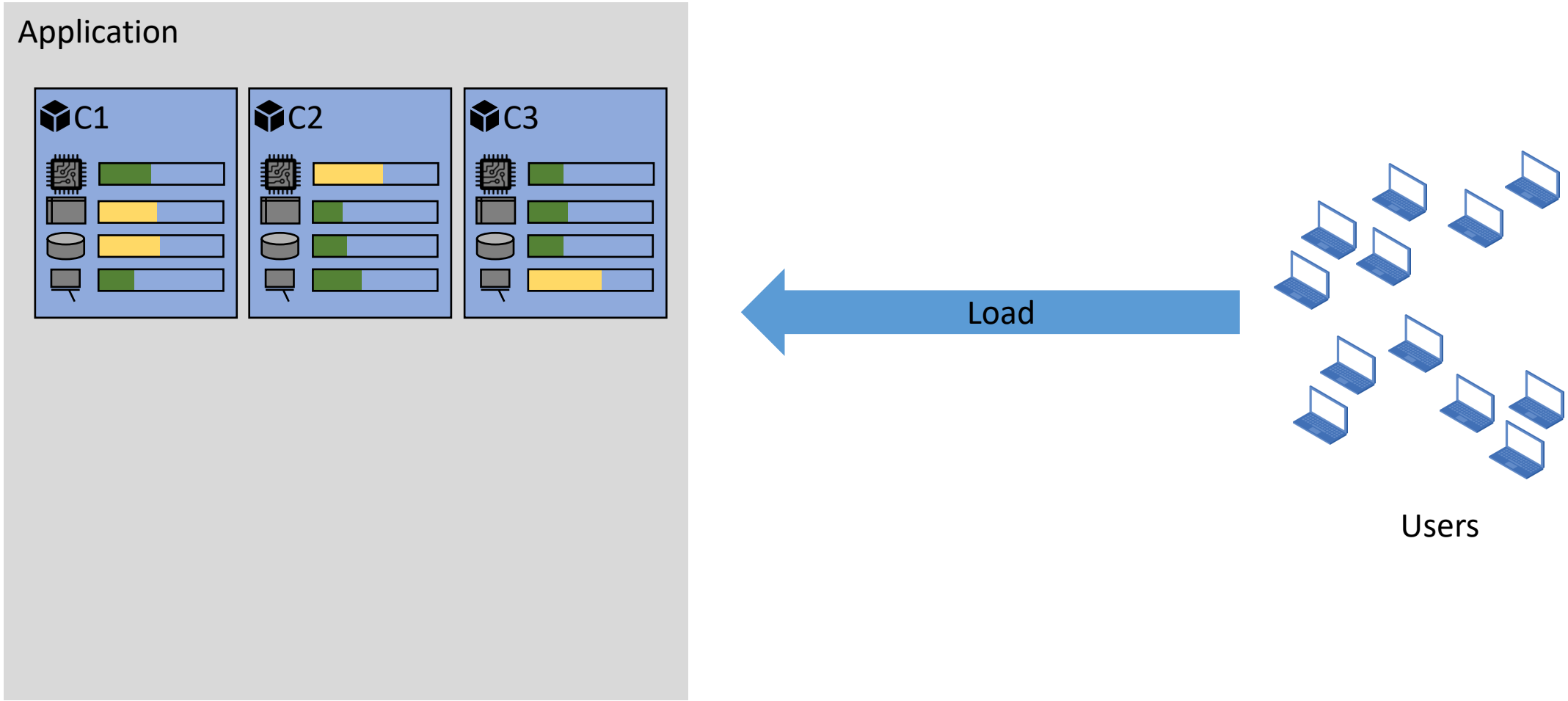
Autoscaling



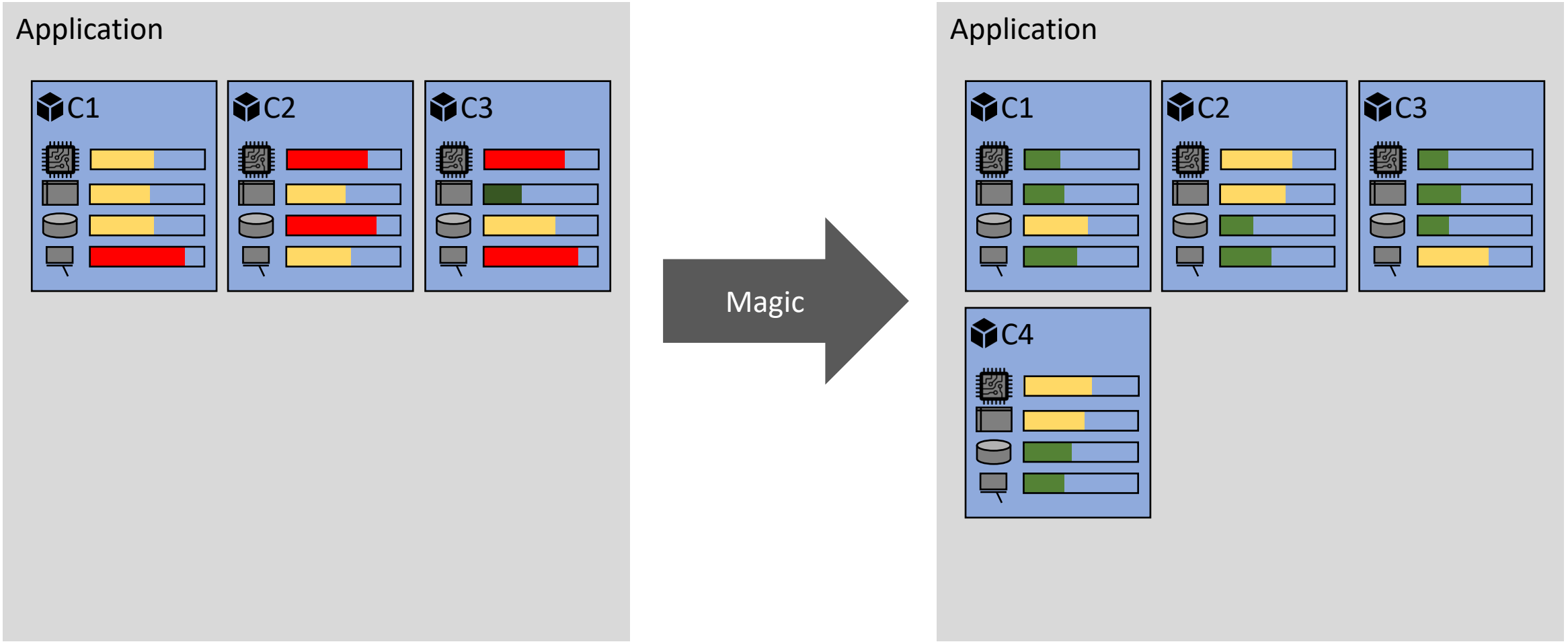
Autoscaling



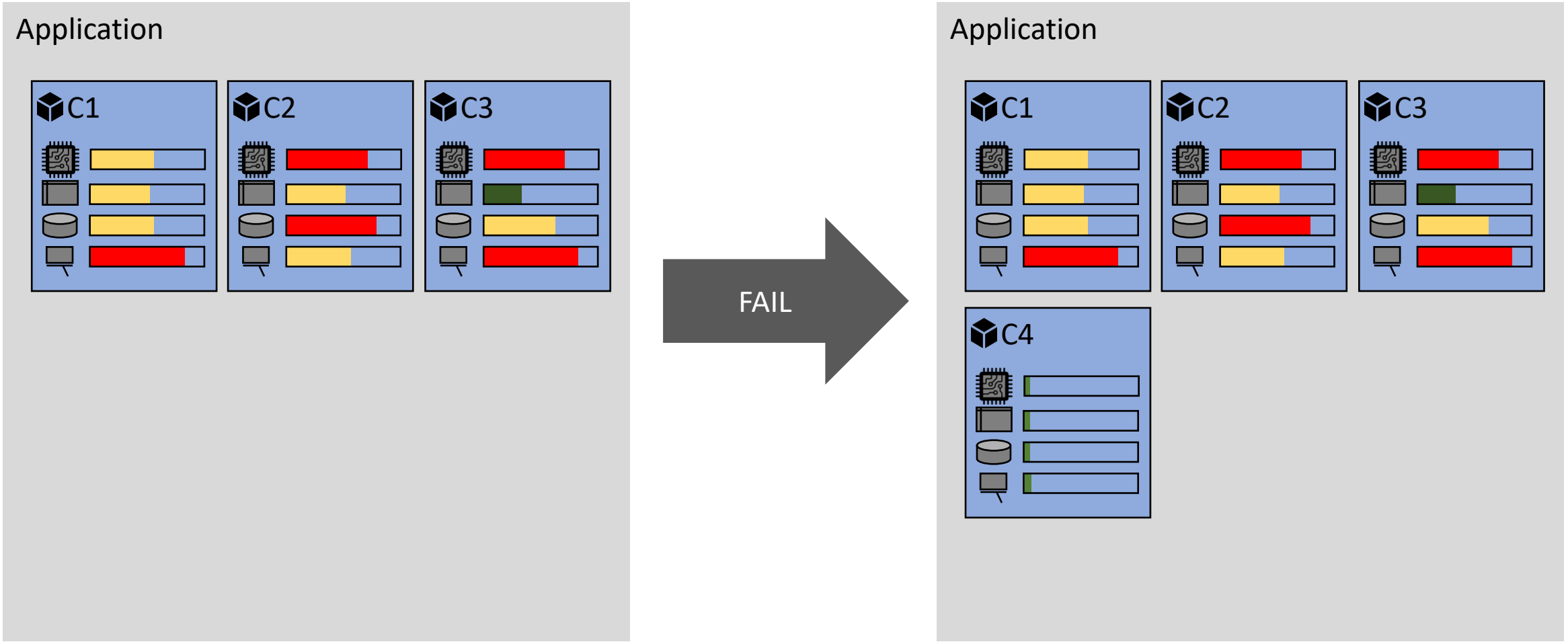
Autoscaling



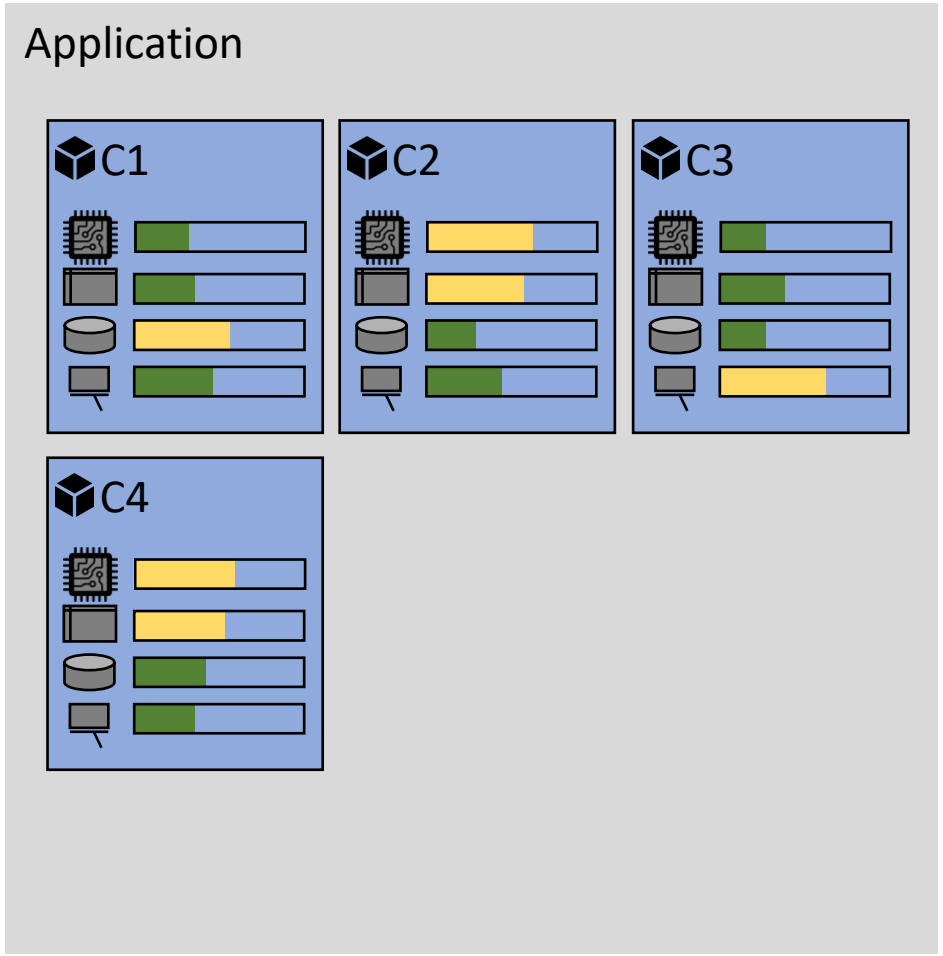
Load balancing

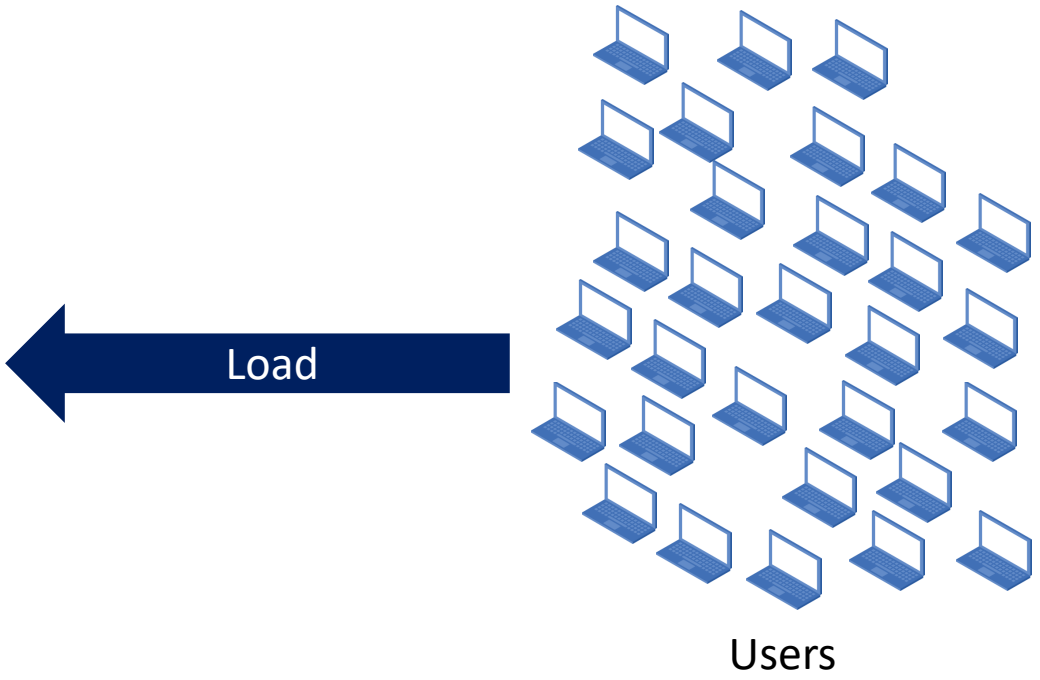


Load balancing

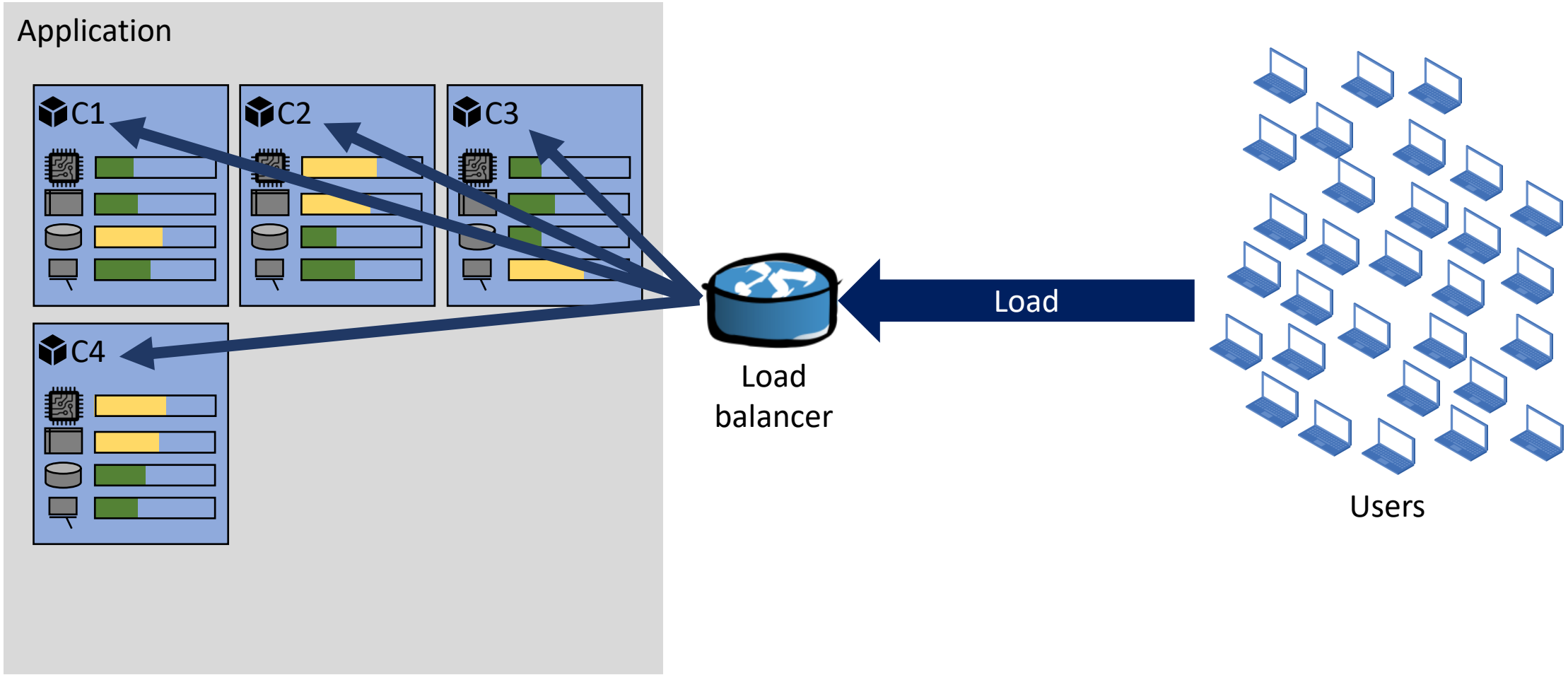


Load balancing

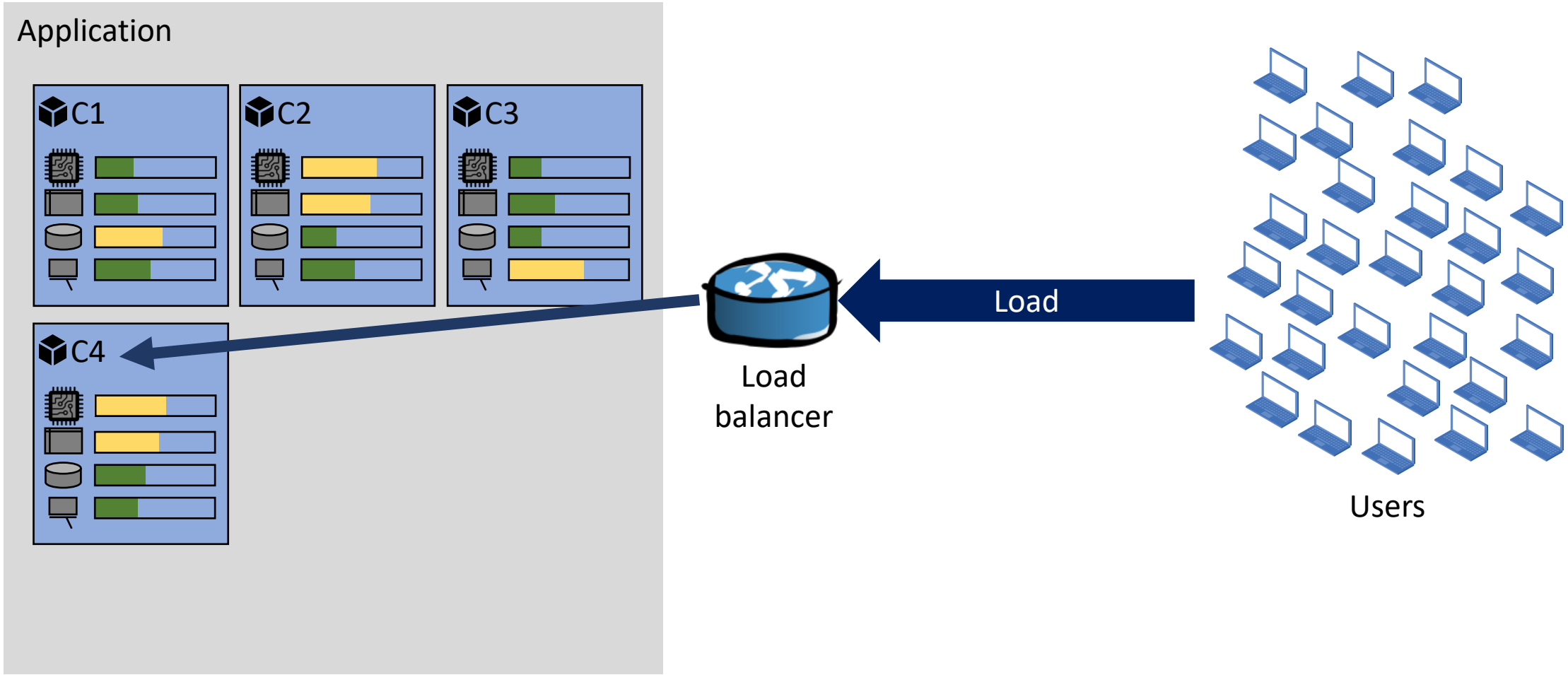




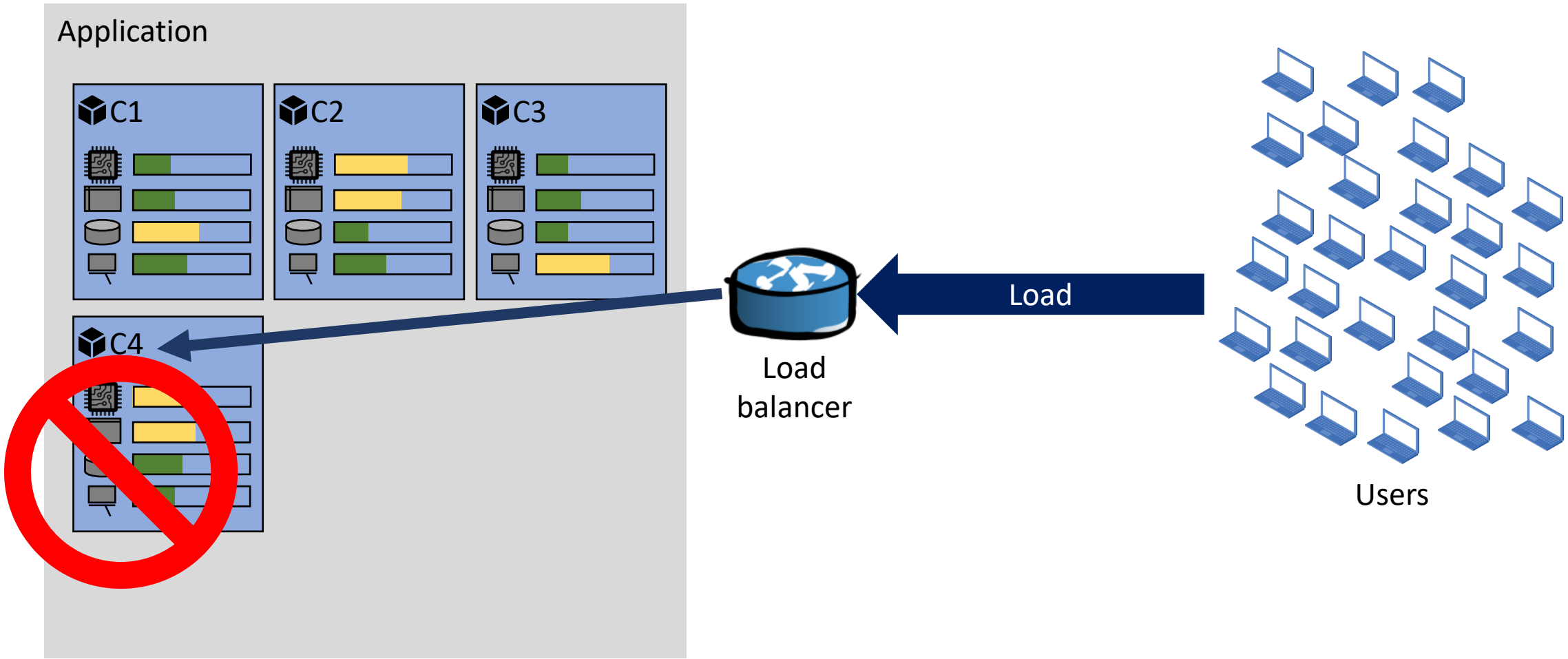
Load balancing



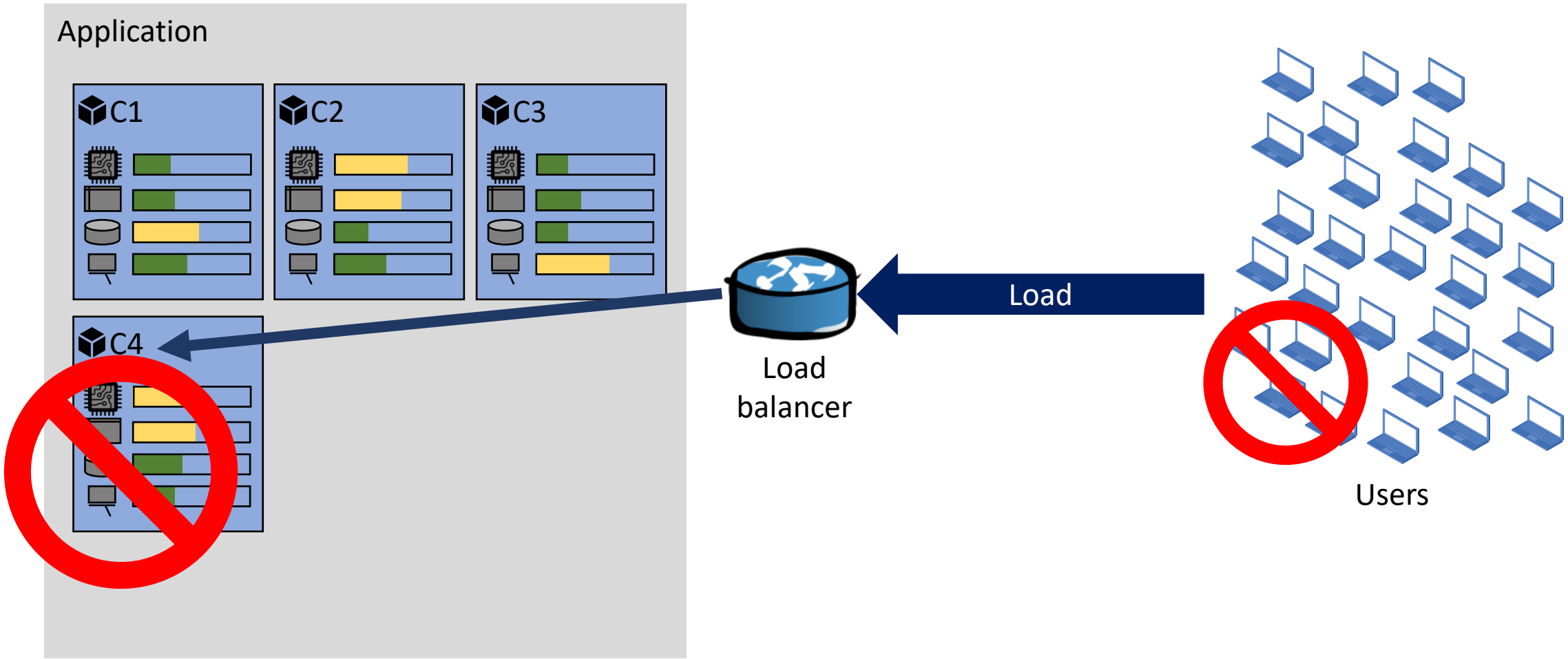
Failure recovery



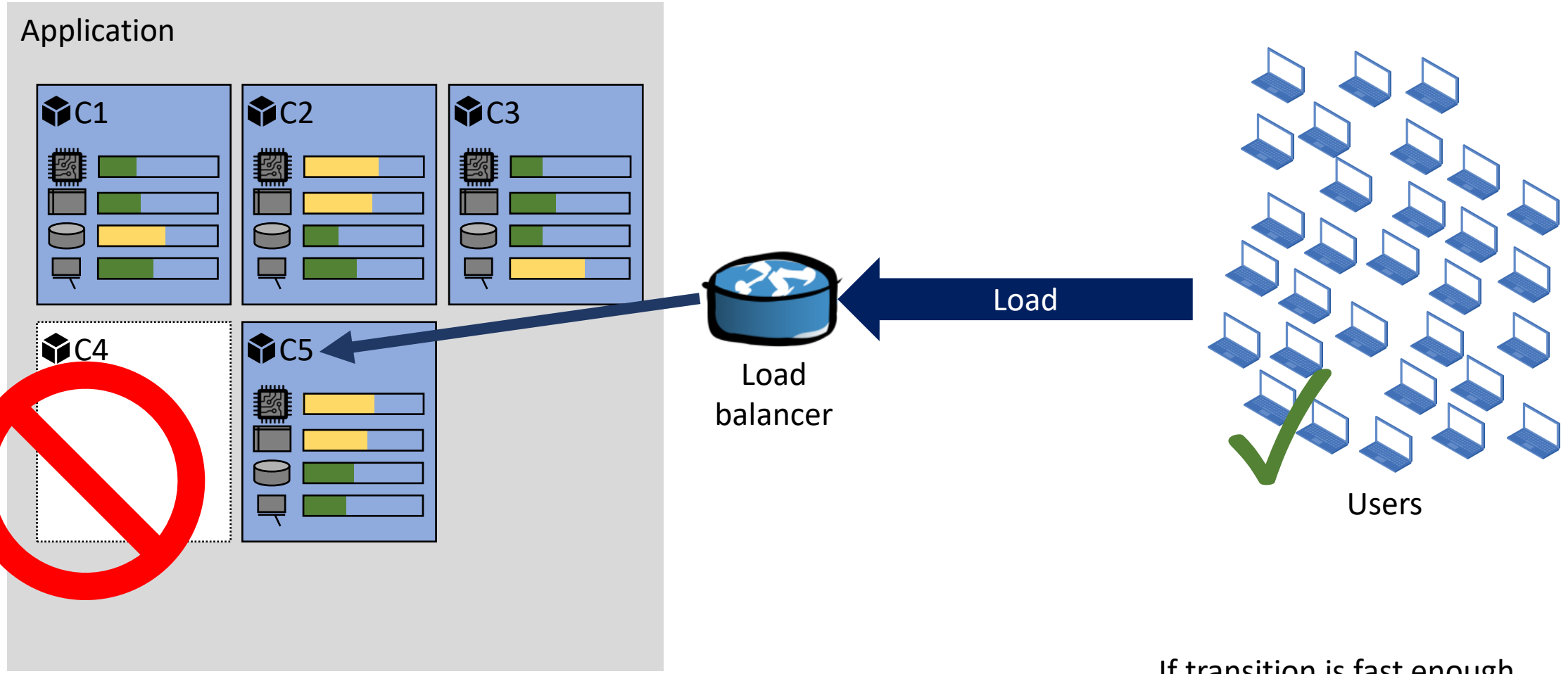
Failure recovery



Failure recovery

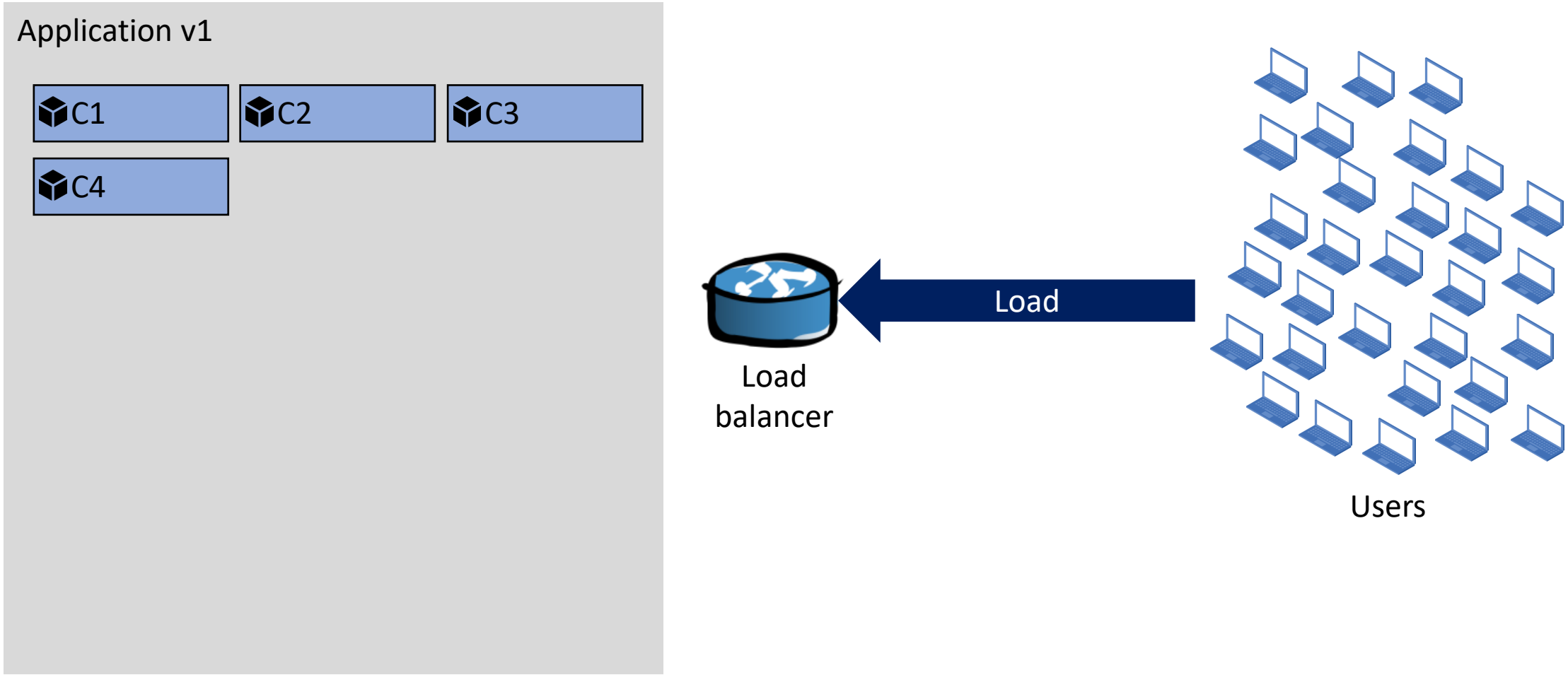


Failure recovery

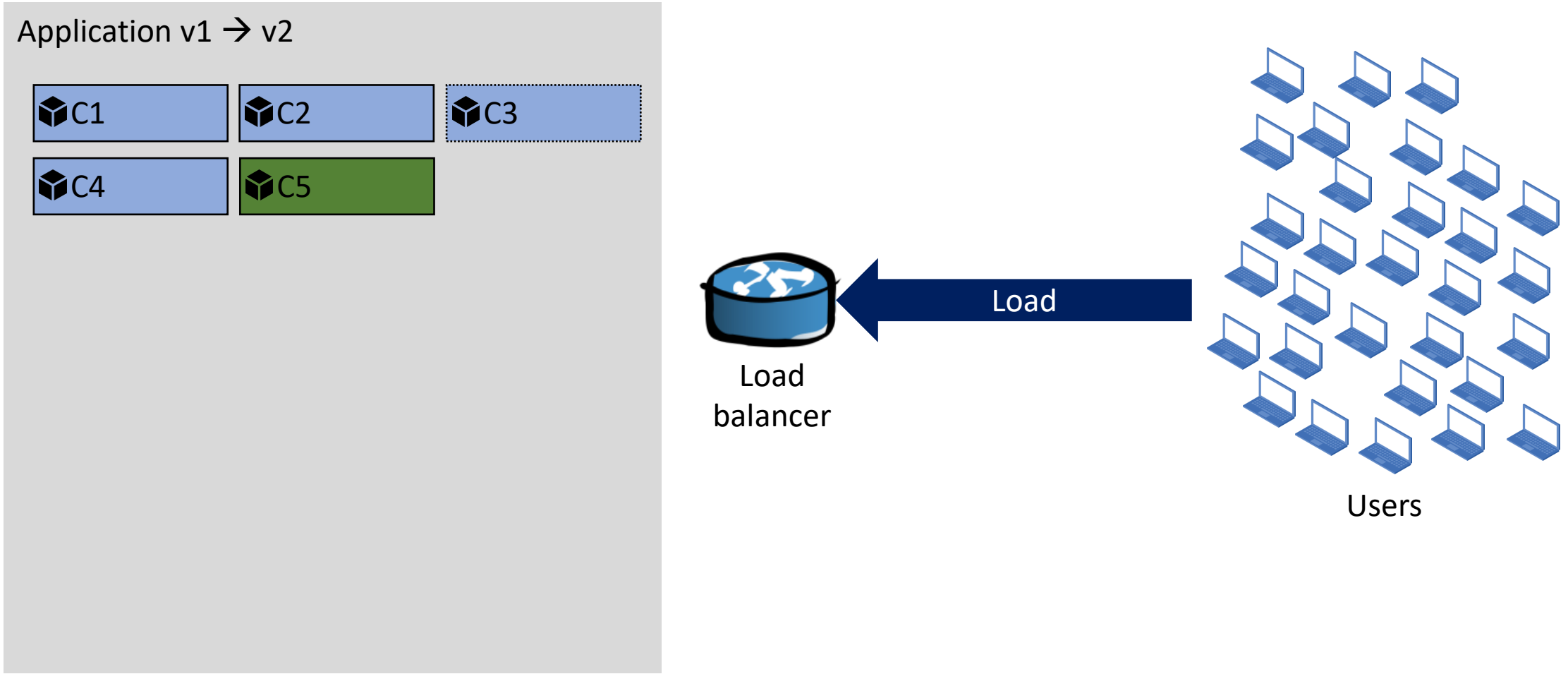


If transition is fast enough, clients might not perceive failure

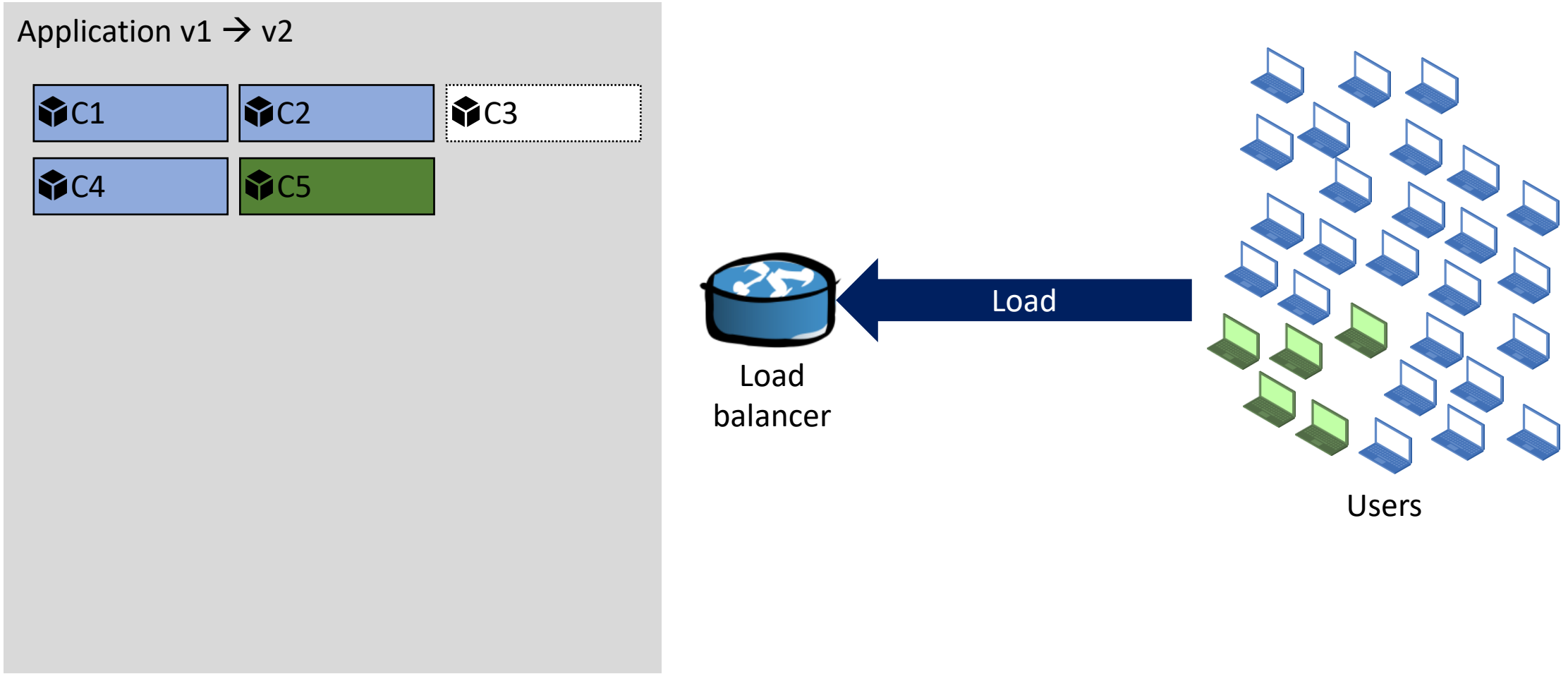
Update rollouts and rollbacks



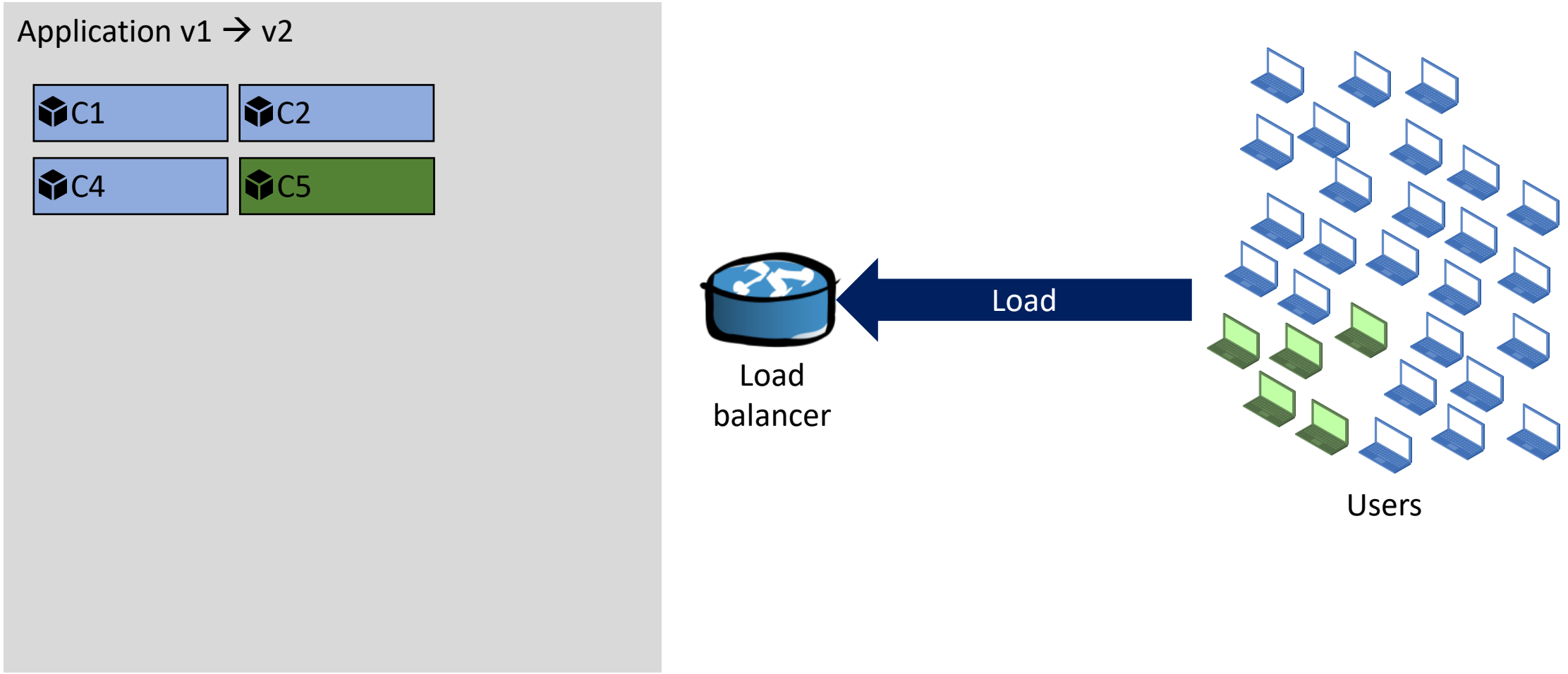
Update rollouts and rollbacks



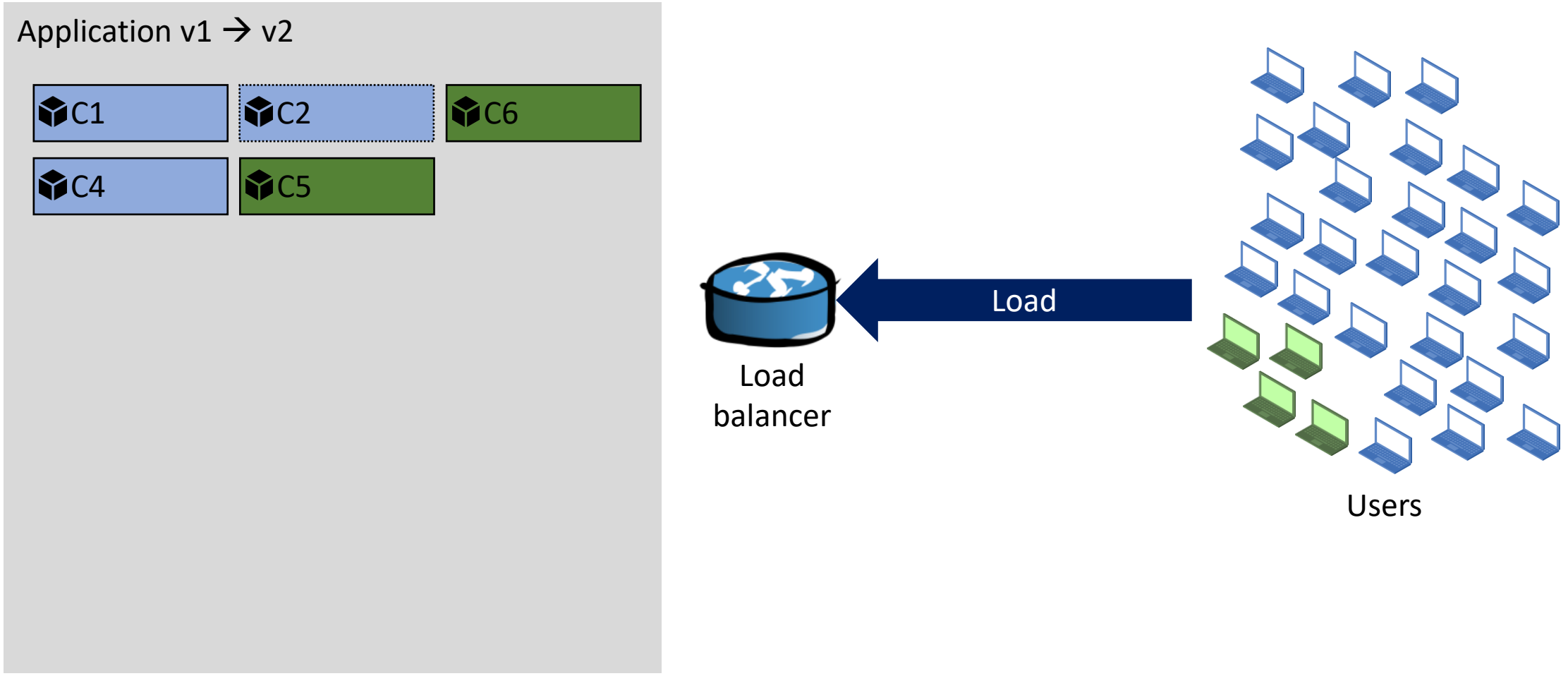
Update rollouts and rollbacks



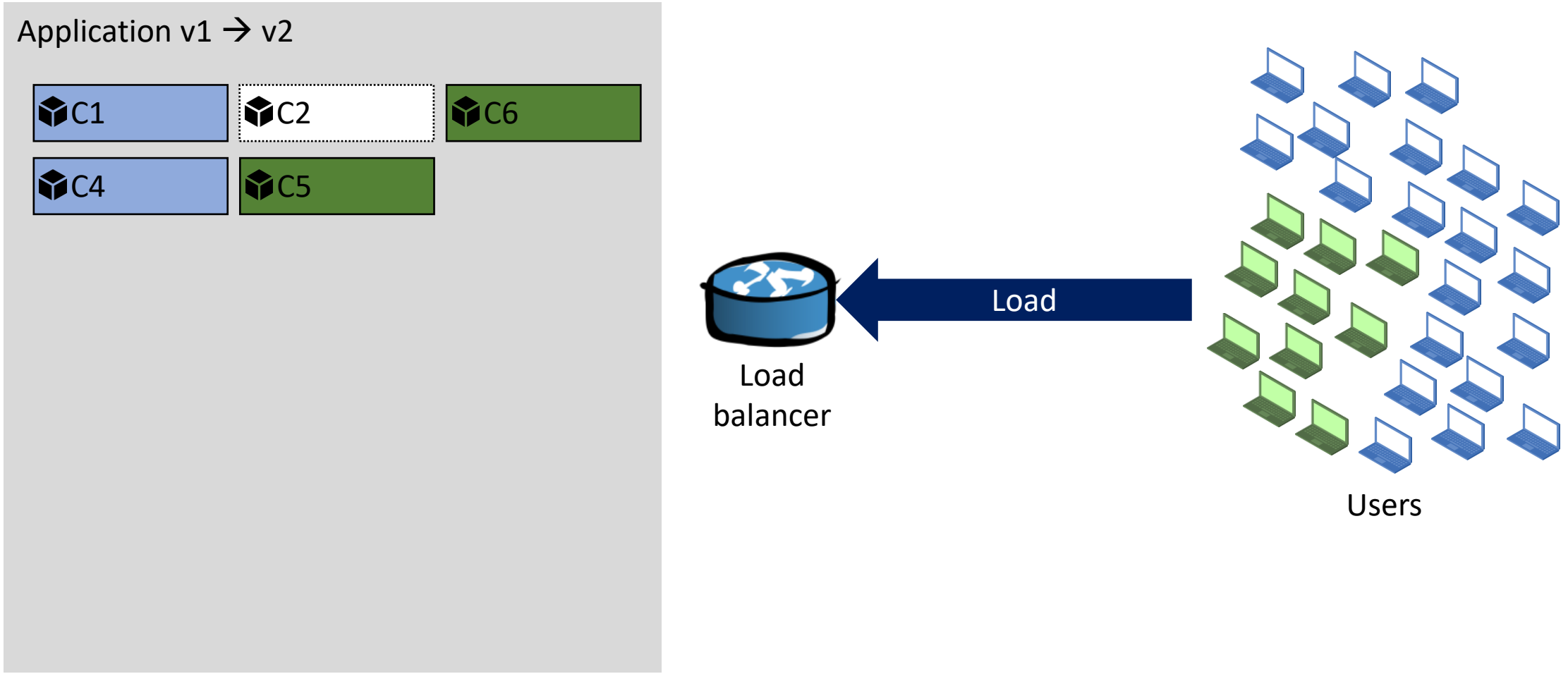
Update rollouts and rollbacks



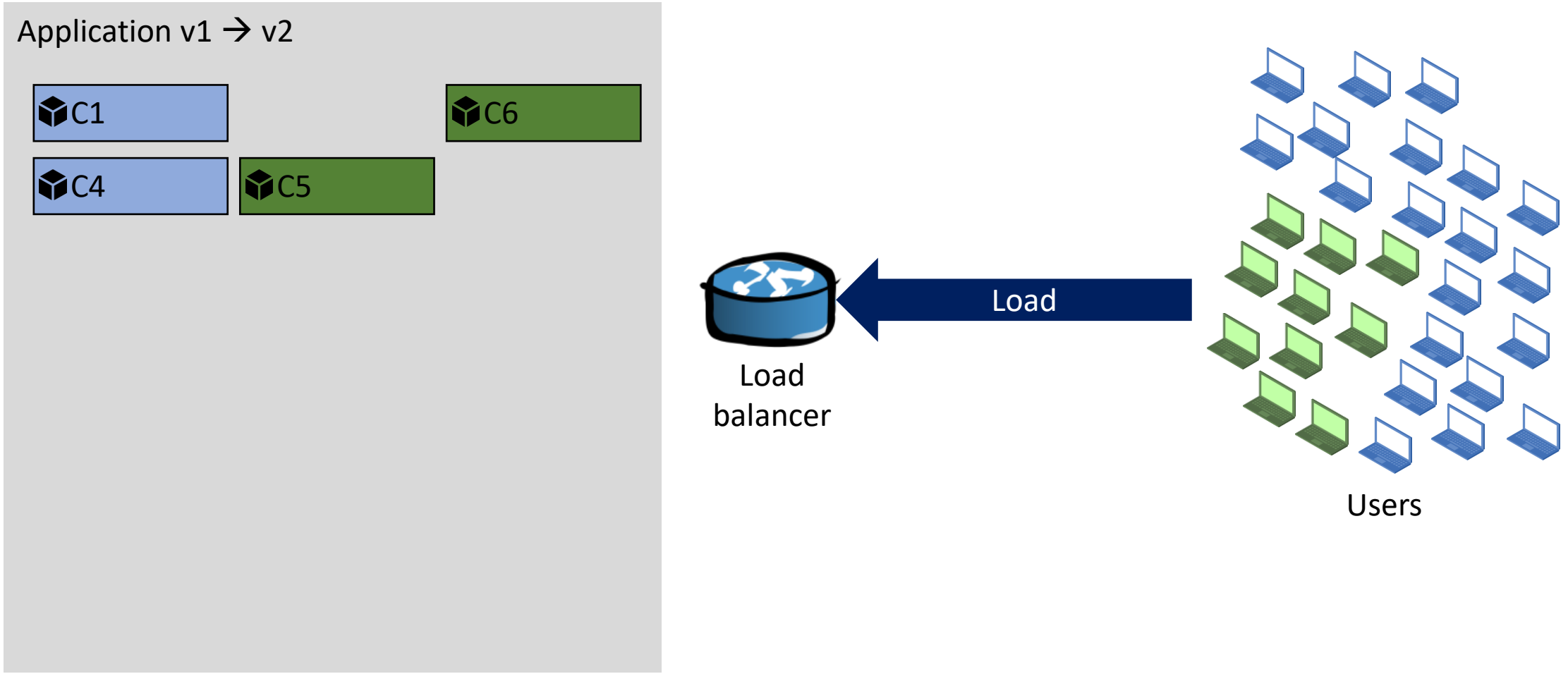
Update rollouts and rollbacks



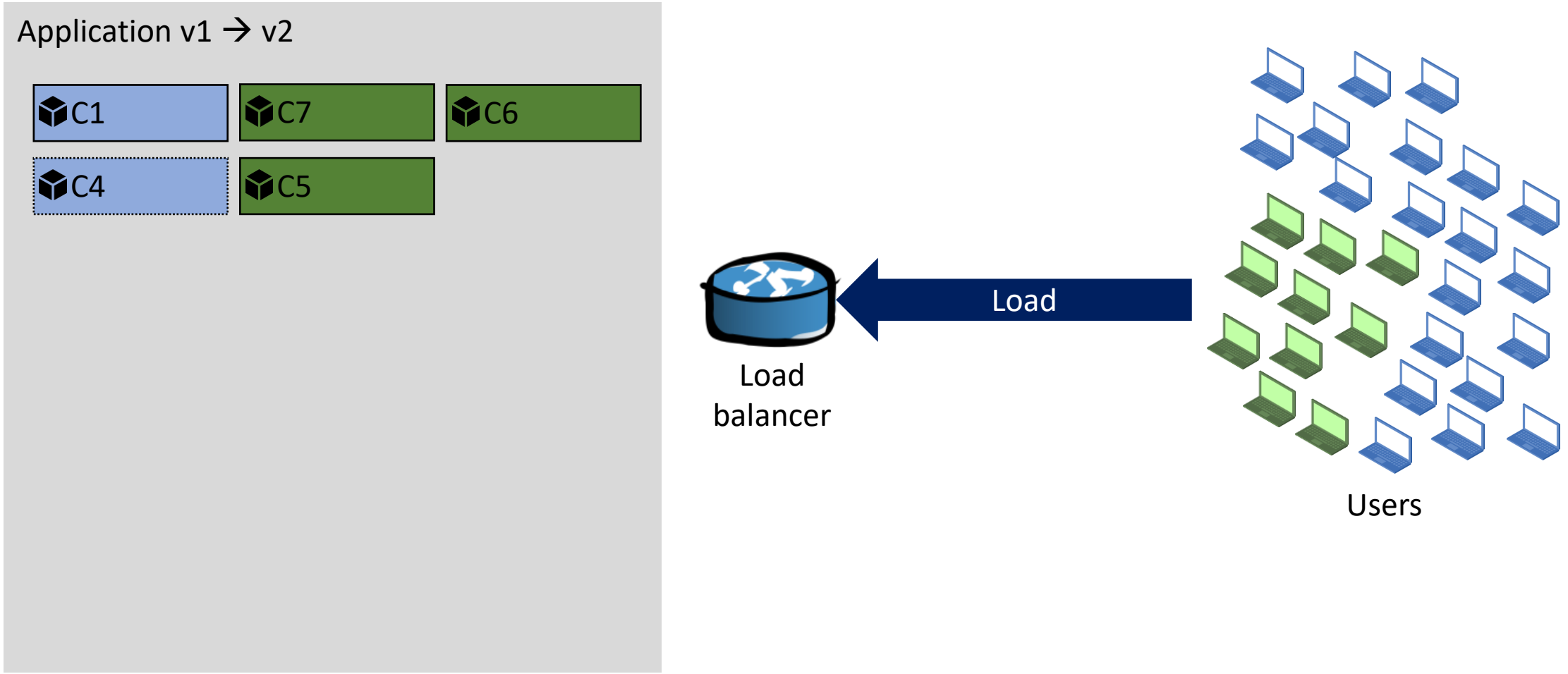
Update rollouts and rollbacks



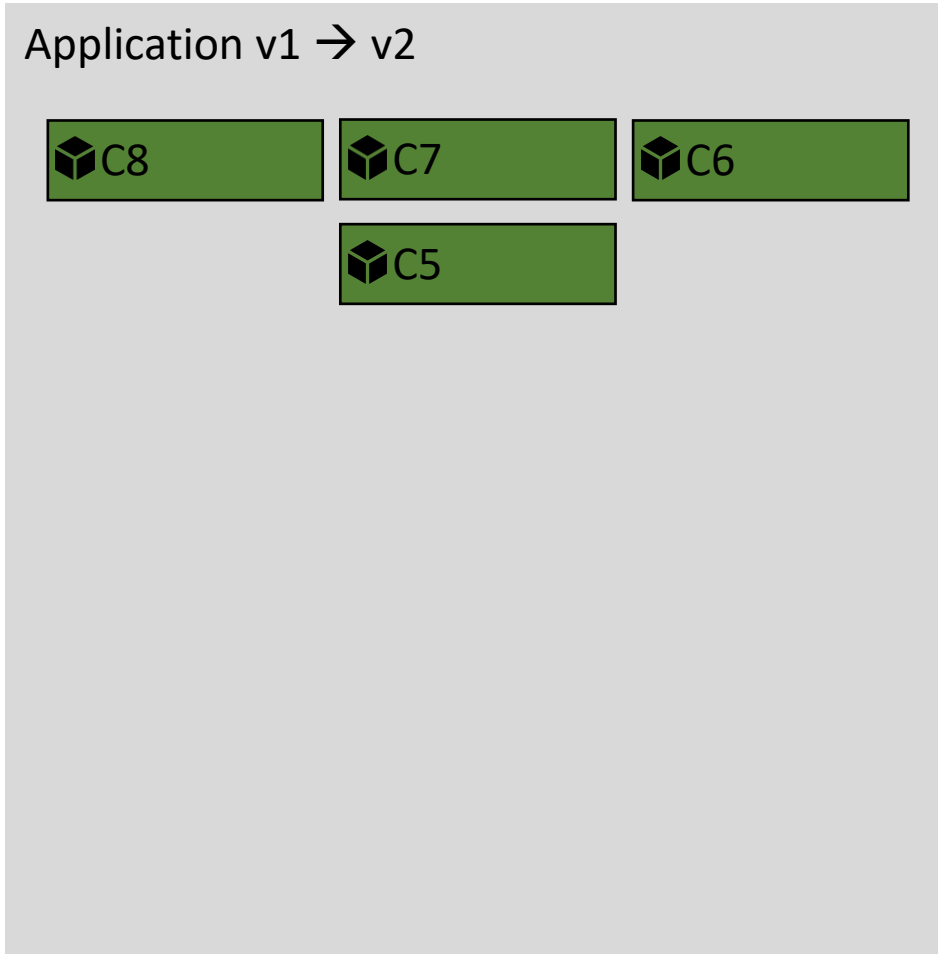
Update rollouts and rollbacks

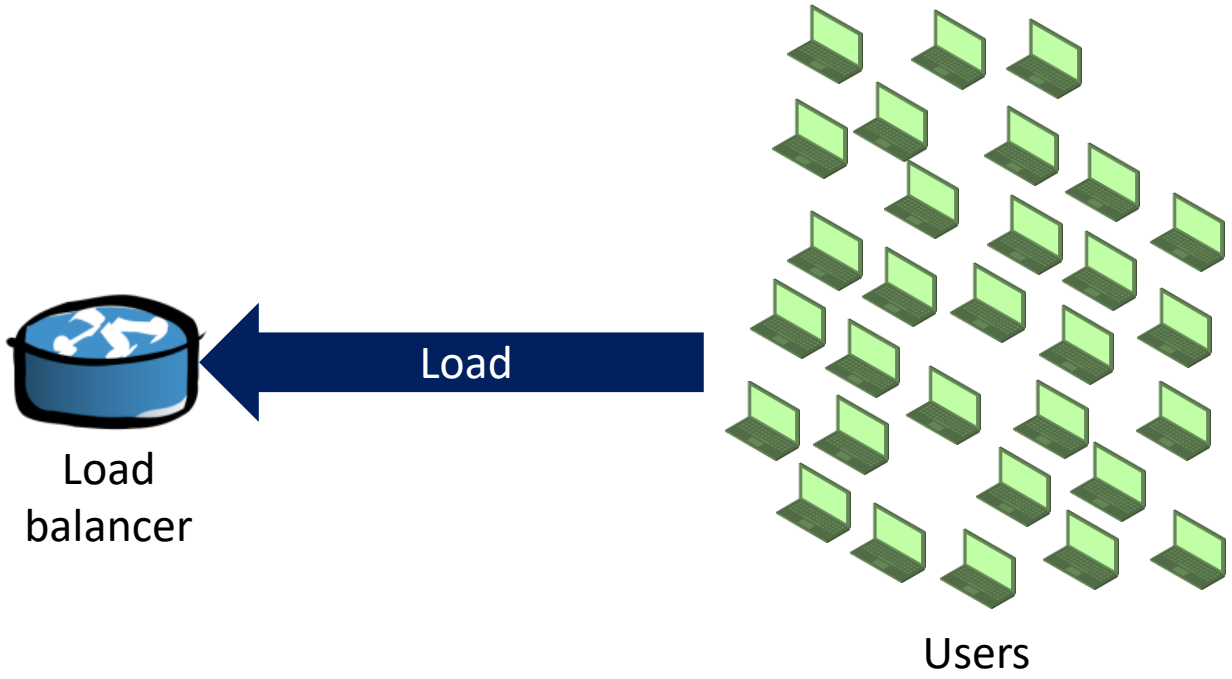


Update rollouts and rollbacks



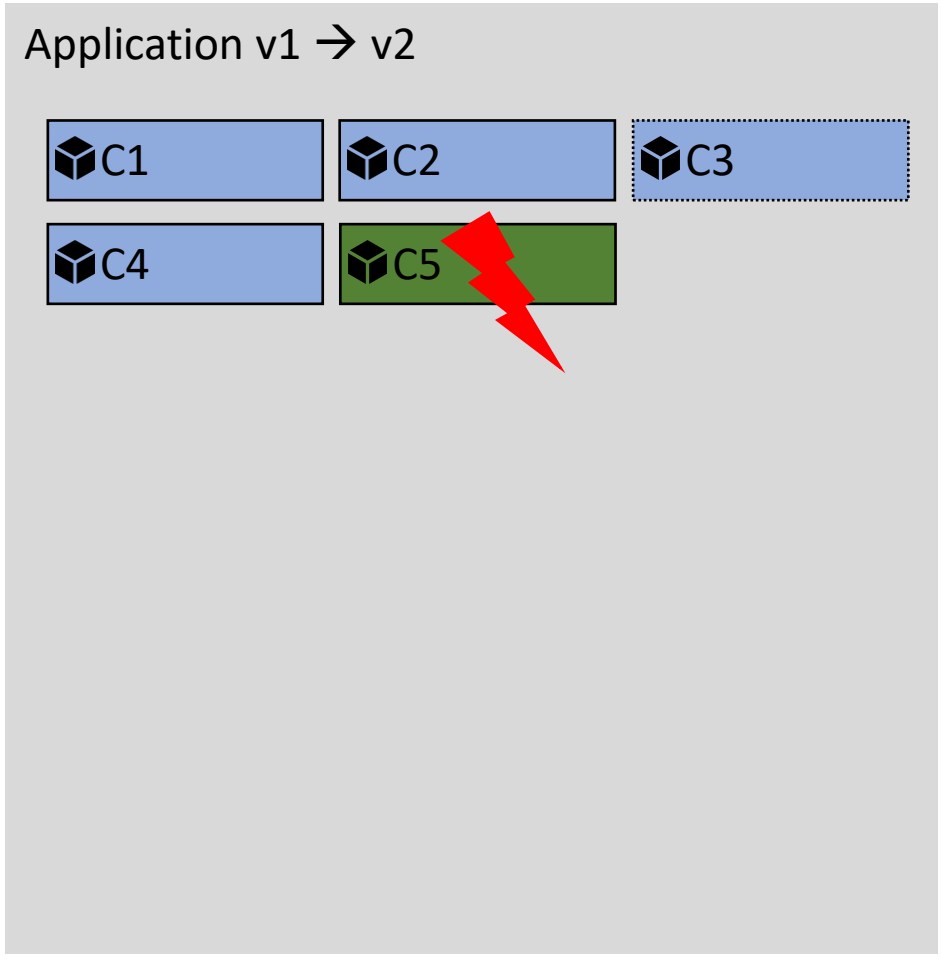
Update rollouts and rollbacks

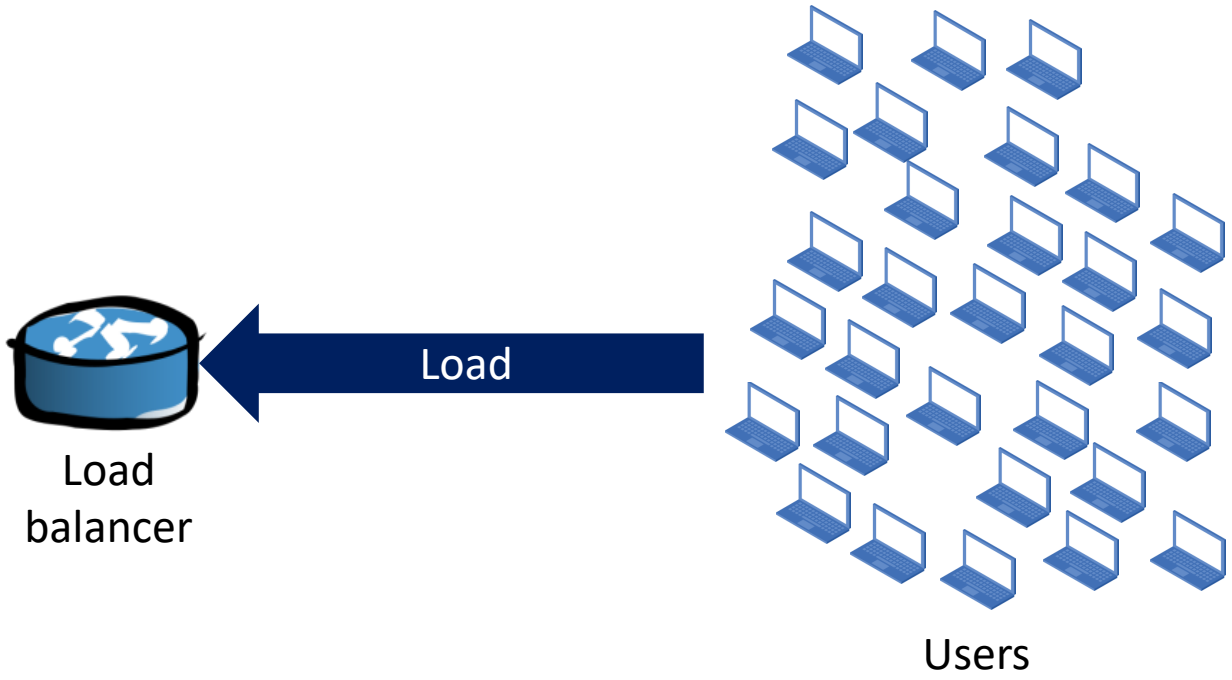




All clients running updated version, zero downtime

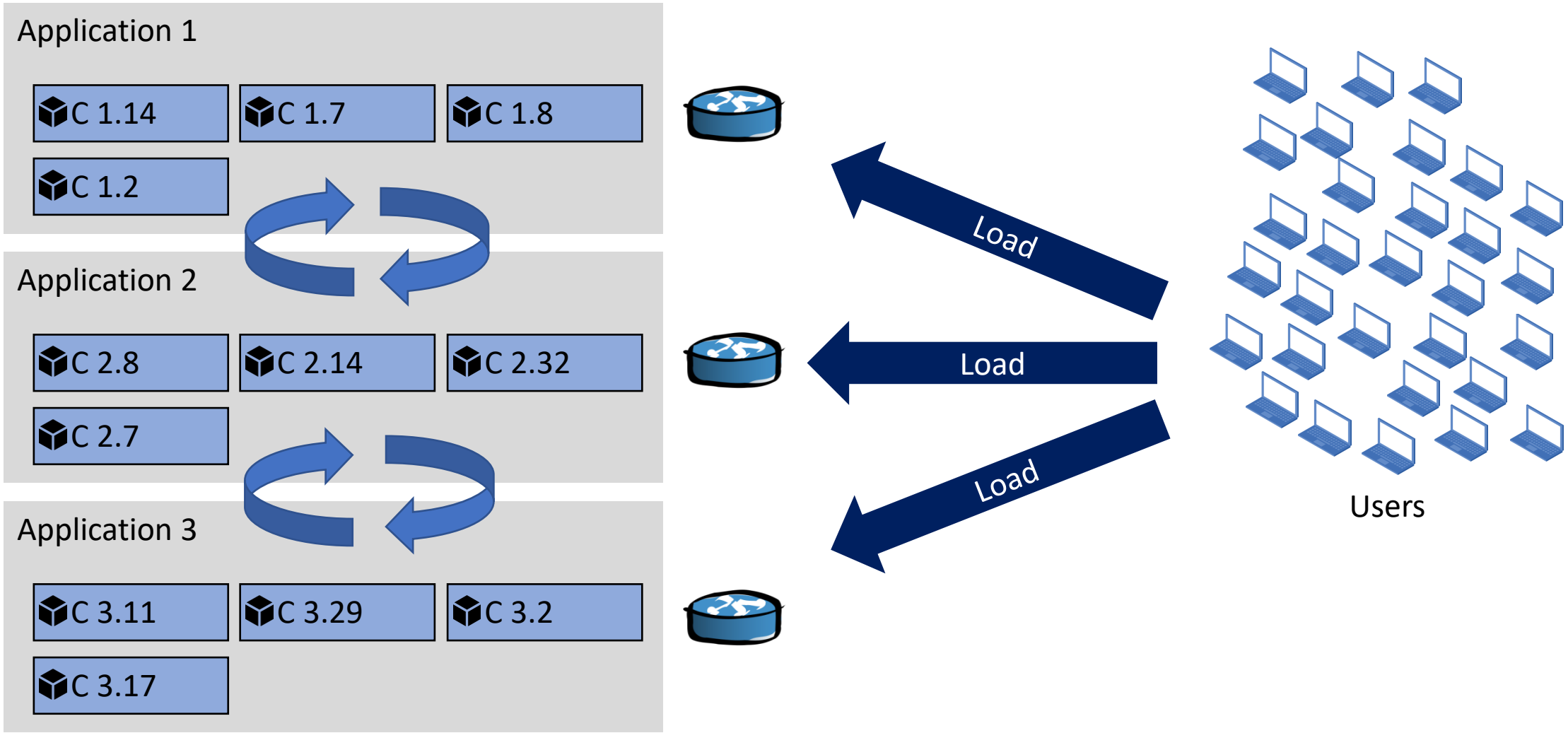
Early error detection during rollouts





Stop rollout if C5 misbehaves

Naming and discovery



Functionalities

- Optimized placement
- Resource monitoring
- Autoscaling
- Load balancing
- Failure recovery
- Update rollout and rollback
- Naming and discovery

CompSci 401: Cloud Computing

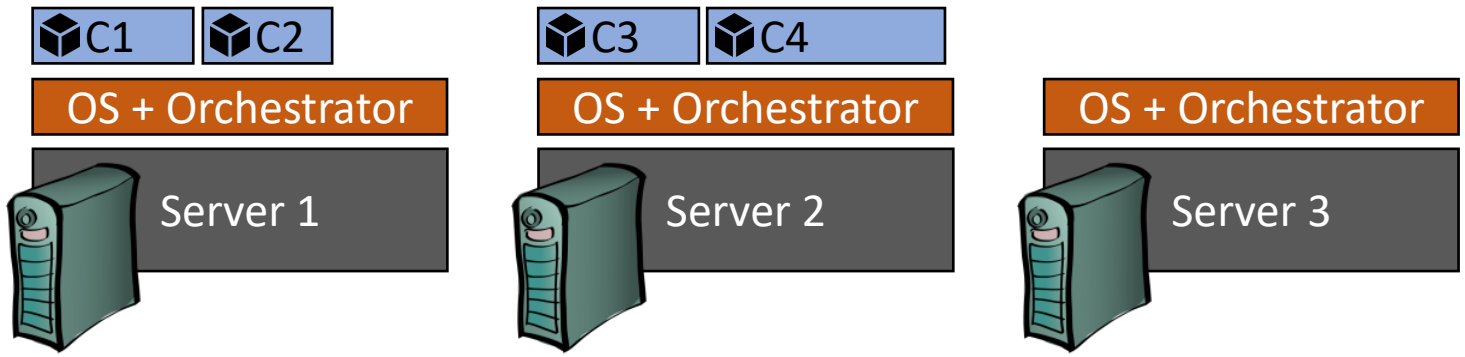
Kubernetes

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

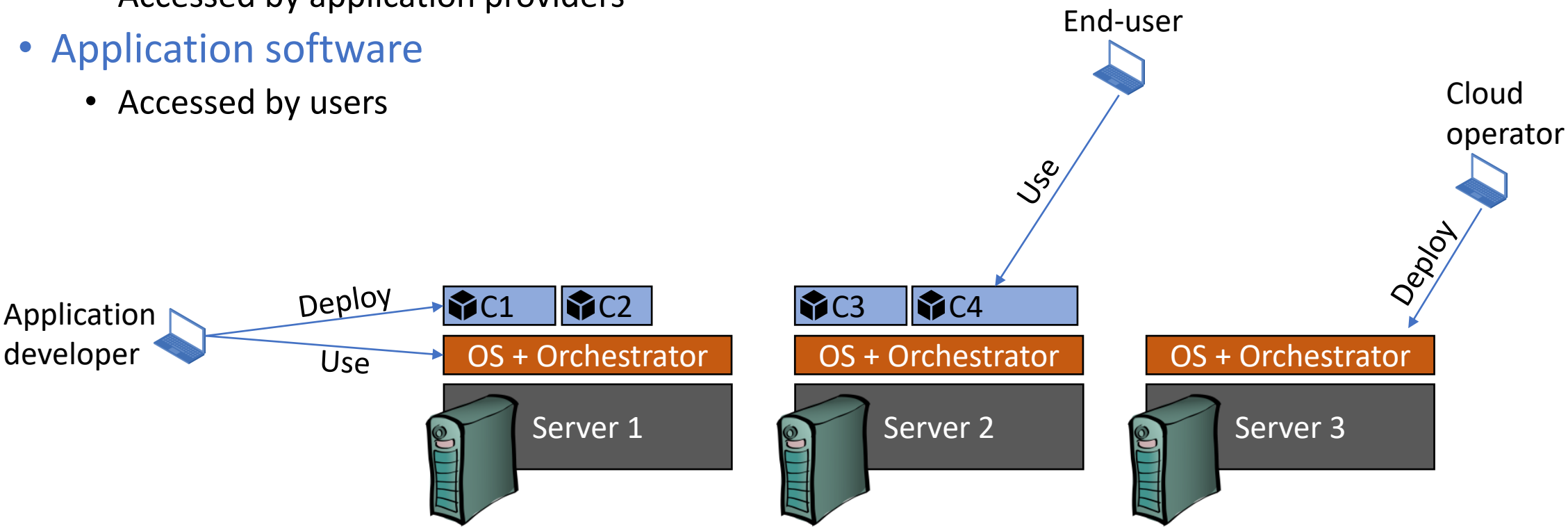
Kubernetes cluster model

- Container orchestration
- Kubernetes is complex and has many functionalities
- It helps to understand the terminology is uses
  - Cluster: Set of containers and orchestration software, number of containers limited by demand and hardware



Kubernetes cluster model

- Two layers of software in Kubernetes
  - Orchestration software
    - Accessed by application providers
  - Application software
    - Accessed by users

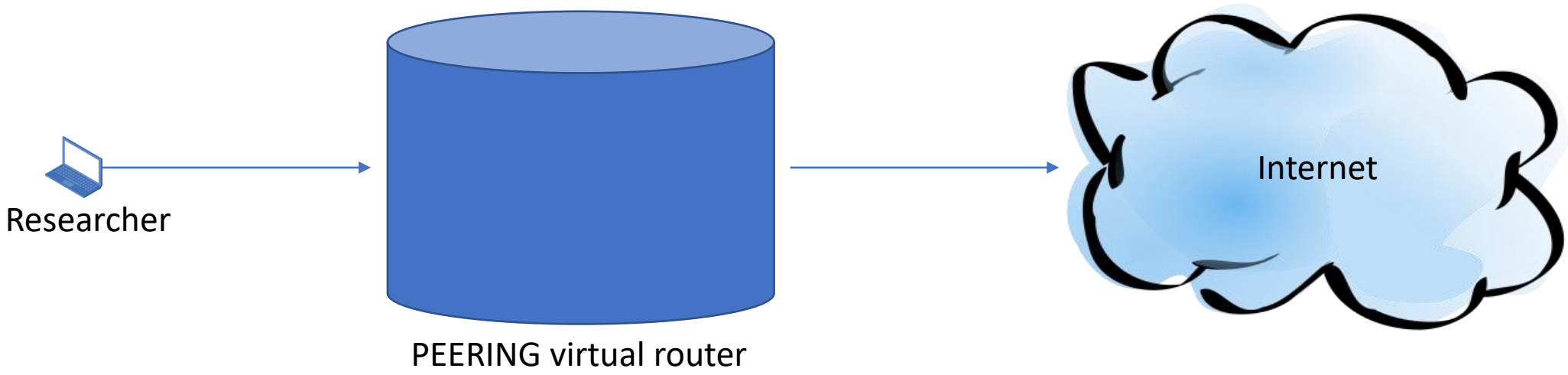


Kubernetes pods

- Kubernetes manages pods, not individual containers
  - The pod is the smallest unit that can be managed
- A pod provides a well-defined functionality
  - A set of tightly coupled containers, but single-container pods are typical
  - Containers in a pod deployed on the same server
  - Containers in a pod communicate over the host's local "network"
    - Naming and discovery provided by the orchestrator

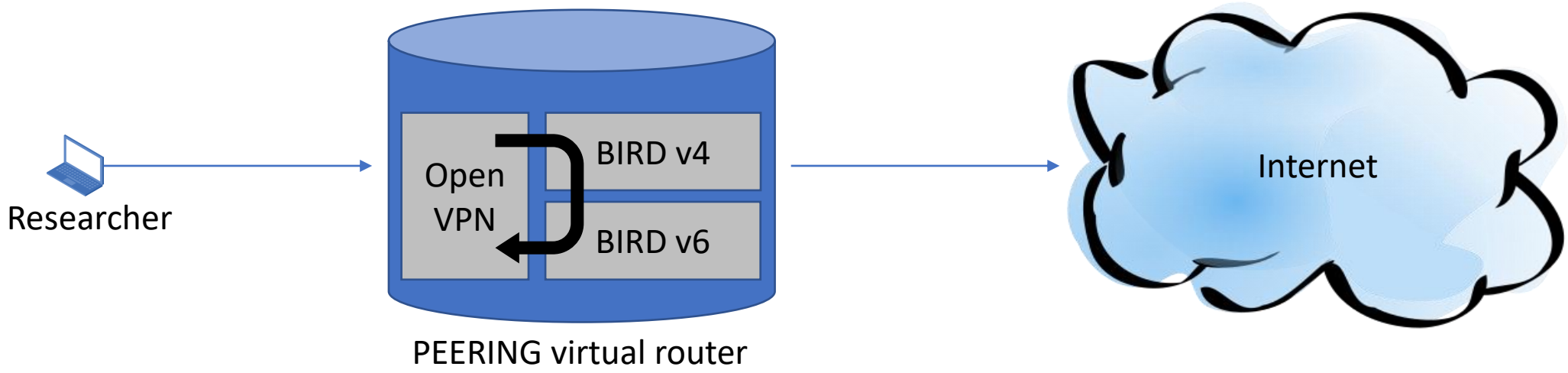
PEERING pod

- PEERING provides a “virtual router” that researchers can use to interact with the Internet
  - Components are packaged in containers
  - Containers cooperate to implement the virtual router



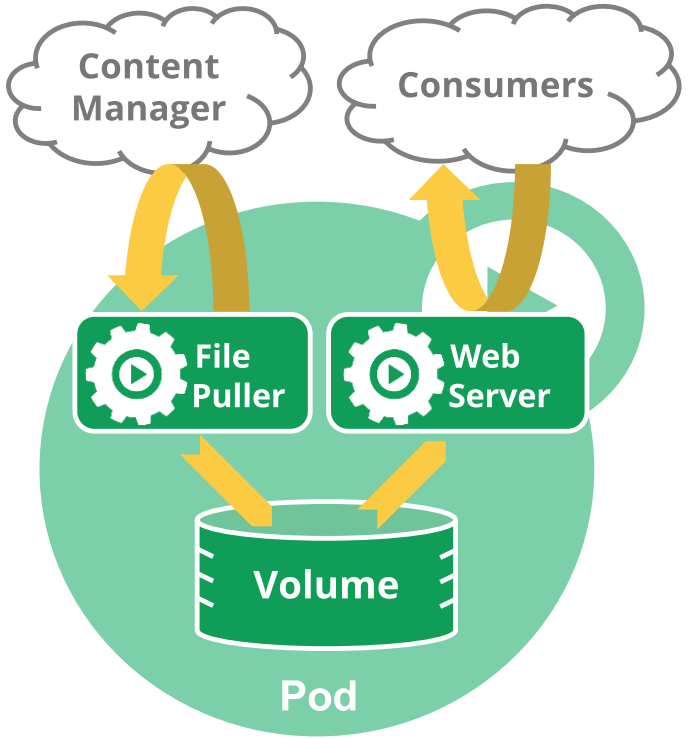
PEERING pod

- PEERING provides a “virtual router” that researchers can use to interact with the Internet
  - Components are packaged in containers
  - Containers cooperate to implement the virtual router



Web pod

- Web server for user requests
- Auxiliary container updates content on the website
- Both containers share a disk volume



Pod specification and creation

- Pods are specified in a Kubernetes-defined API
  - JSON or YAML
- A pod has a specification
  - Name and image of containers
  - Environment variables
  - Network ports
- Pods are instantiated following specification
- A pod can have metadata attached to it
  - Used to search and as management targets

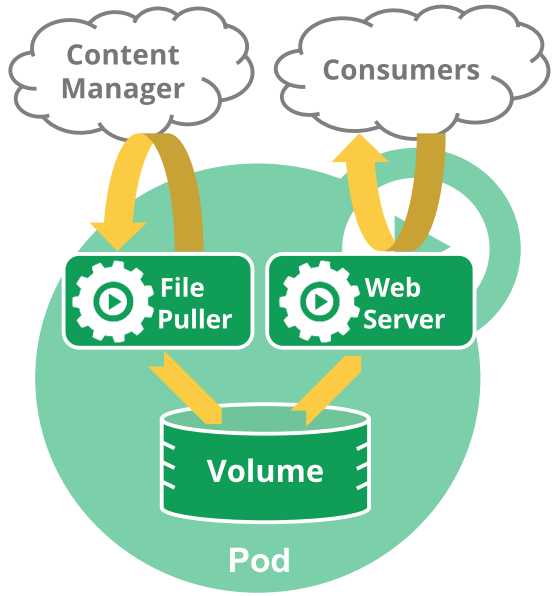
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Init containers

- A pod can specify an init container
- Init container runs before the other containers in the pod
- Performs initial setup or verifications
  - Common use: check the environment before starting other containers

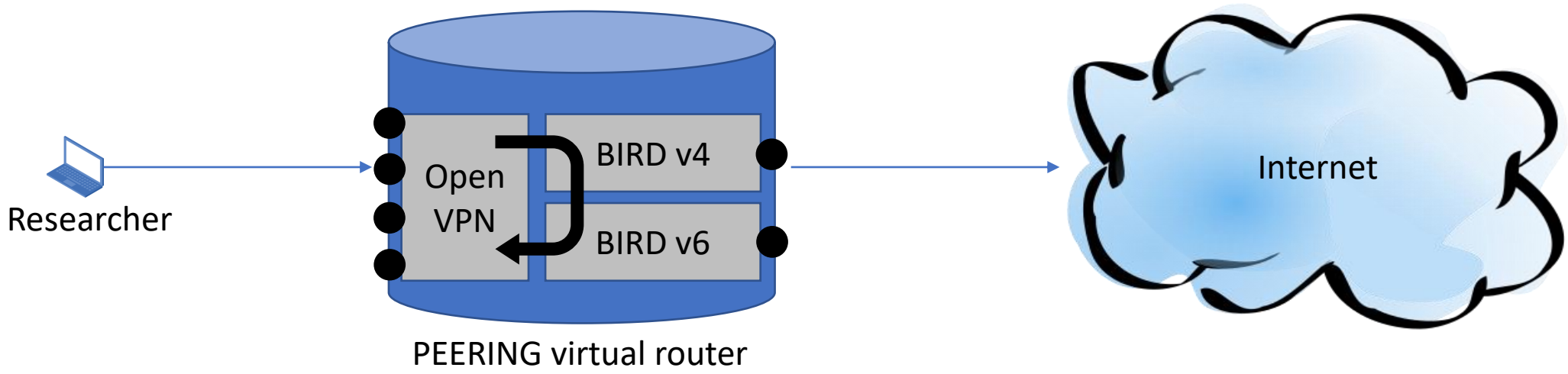
Init containers

- A pod can specify an init container
- Init container runs before the other containers in the pod
- Performs initial setup or verifications
  - Common use: check the environment before starting other containers



Init containers

- A pod can specify an init container
- Init container runs before the other containers in the pod
- Performs initial setup or verifications
  - Common use: check the environment before starting other containers
  - PEERING's init container sets up the networking for the app containers



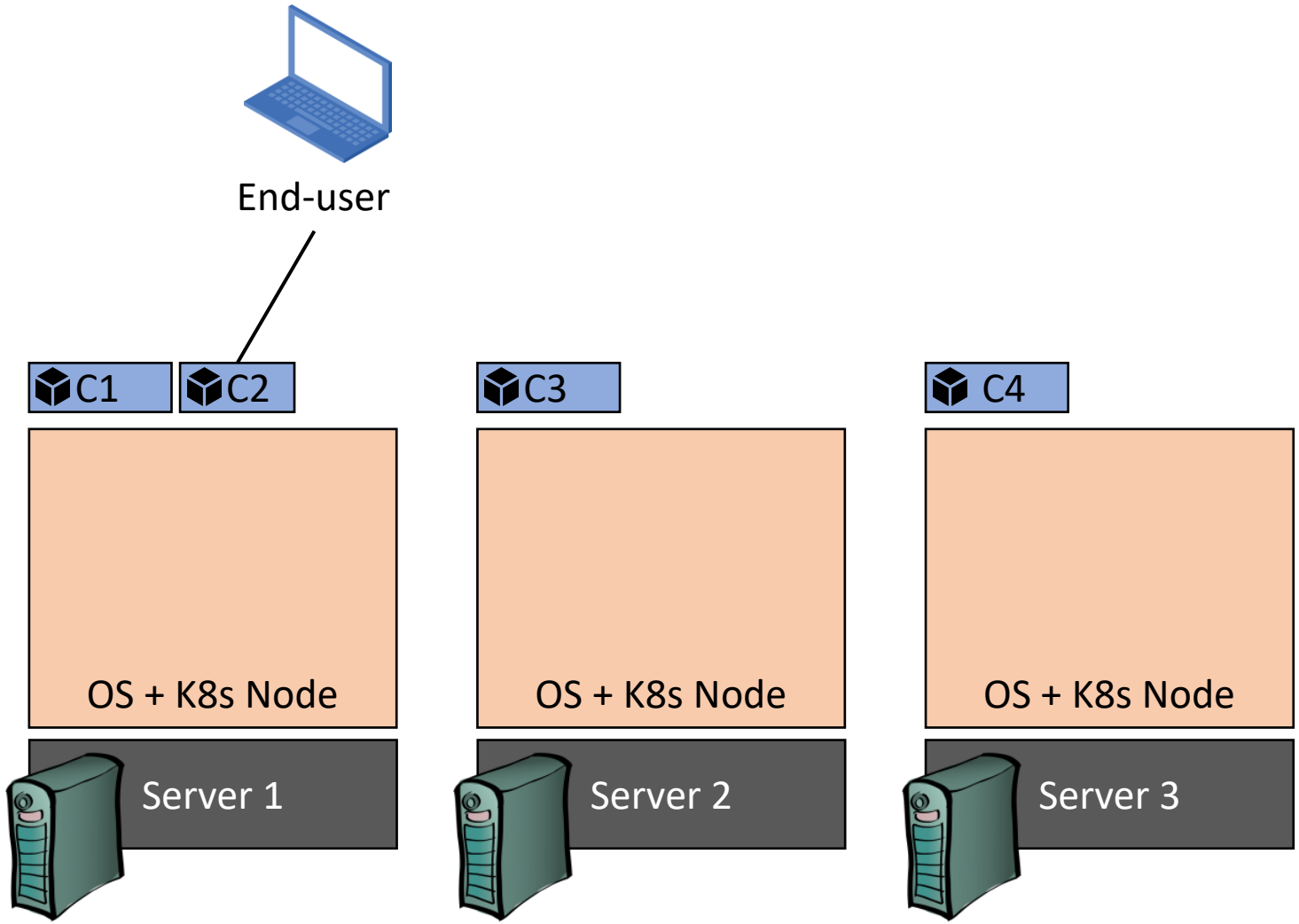
CompSci 401: Cloud Computing

K8s Nodes and Control Plane

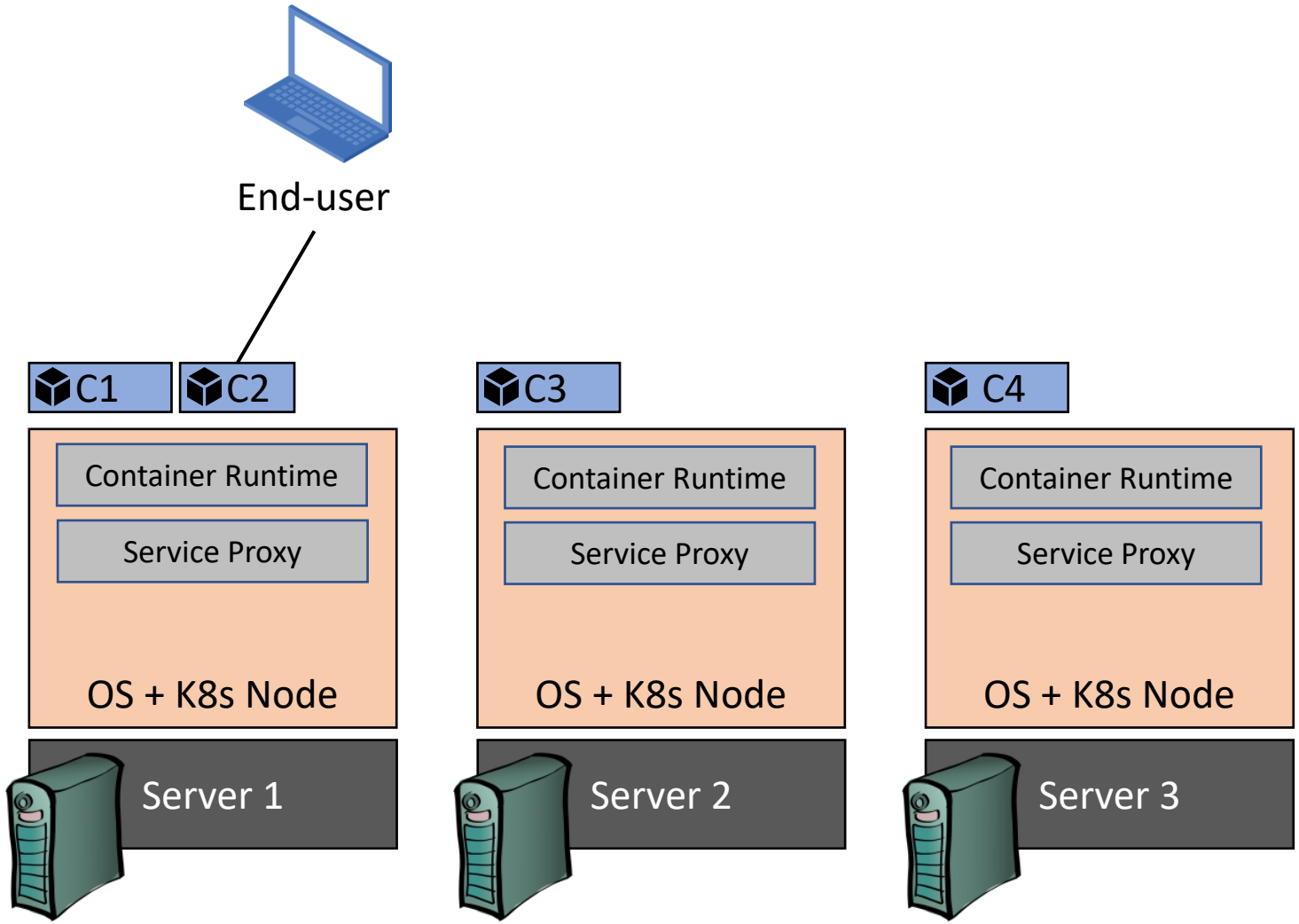
Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

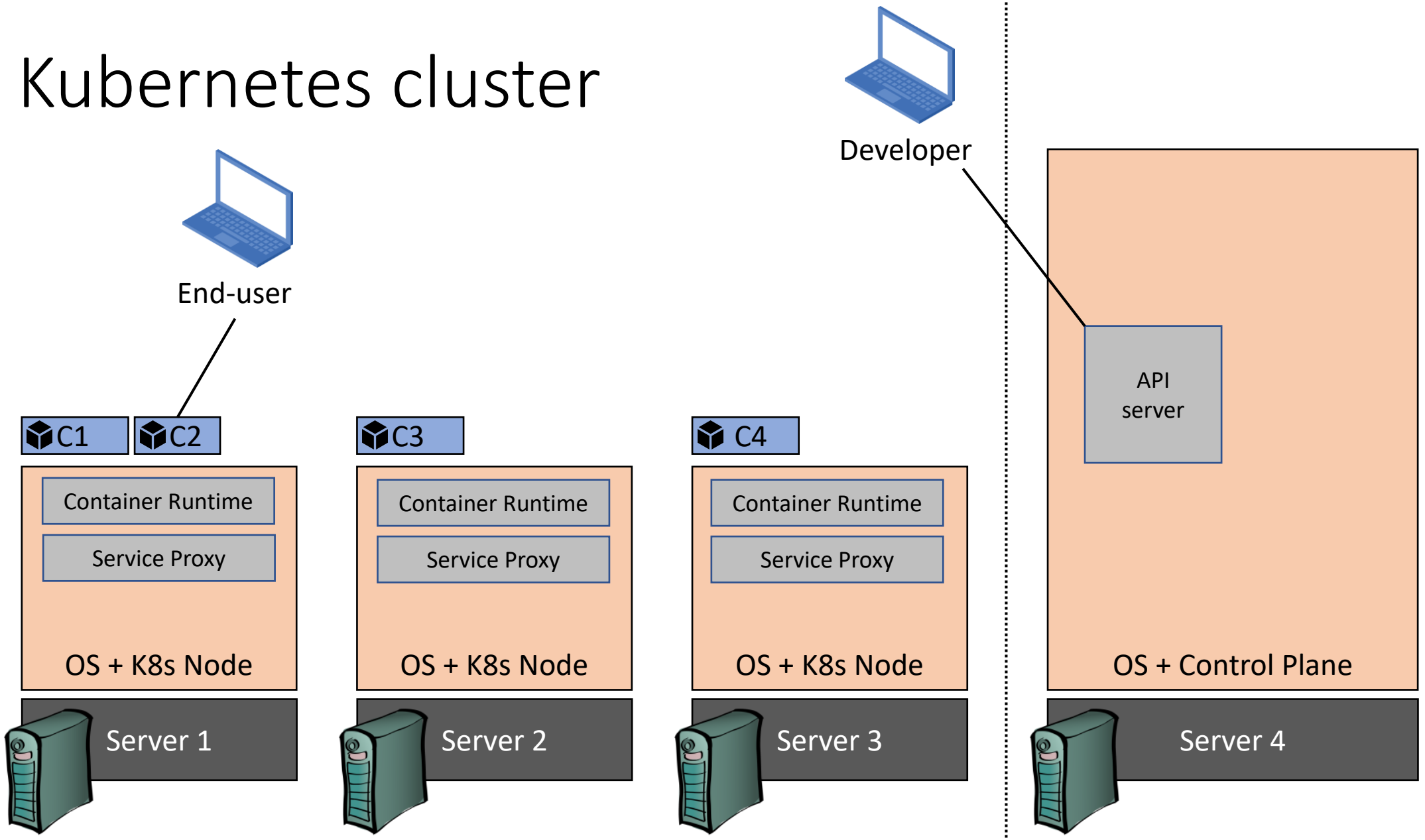
Kubernetes cluster



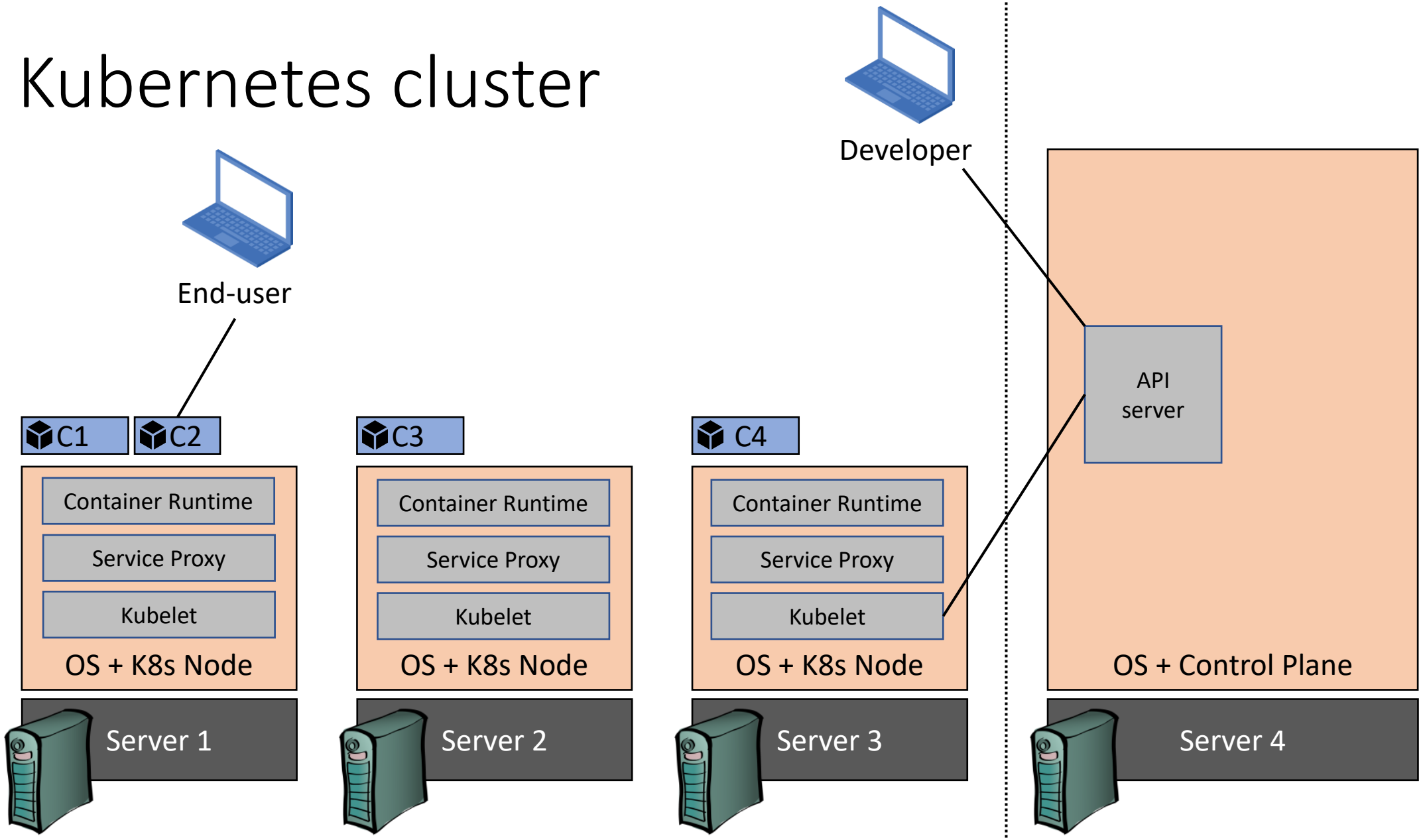
Kubernetes cluster



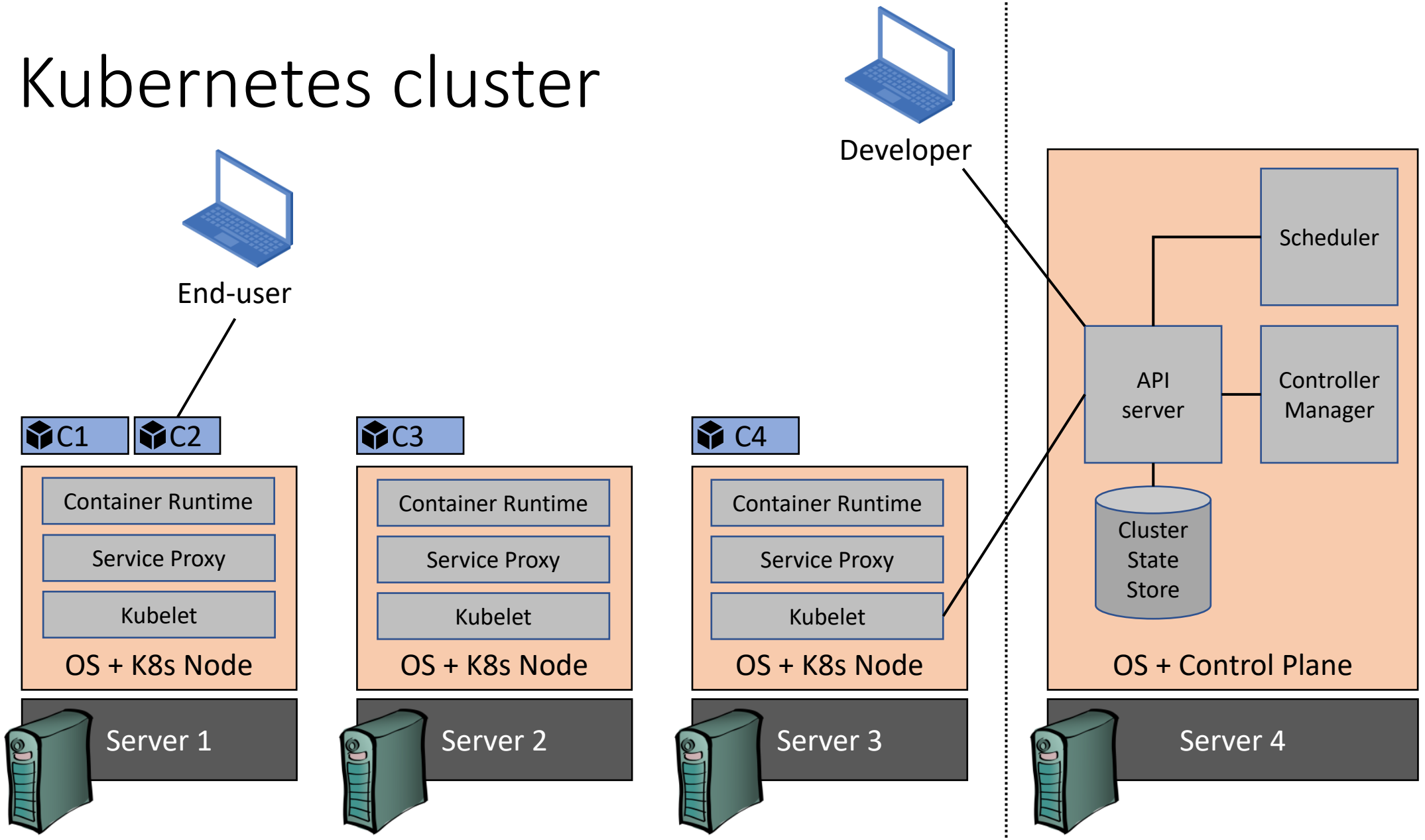
Kubernetes cluster



Kubernetes cluster



Kubernetes cluster



Kubernetes cluster