Kubernetes 架构与原理

邓德源

目录

- Kubernetes 的前世今生
- Kubernetes 的架构和设计原则
- Kubernetes 的工作流程

本PPT未经允许请勿传播，所有解释权归CEUC大会主办方才云科技所有。

About me

- CMU Graduate, ex-Googler
- Fell in love with Robotics, once
- co-founder @ Caicloud
- CNCF TOC contributor
- Member of CNCF Training Committee
- Now actively working on ML system + Kubernetes

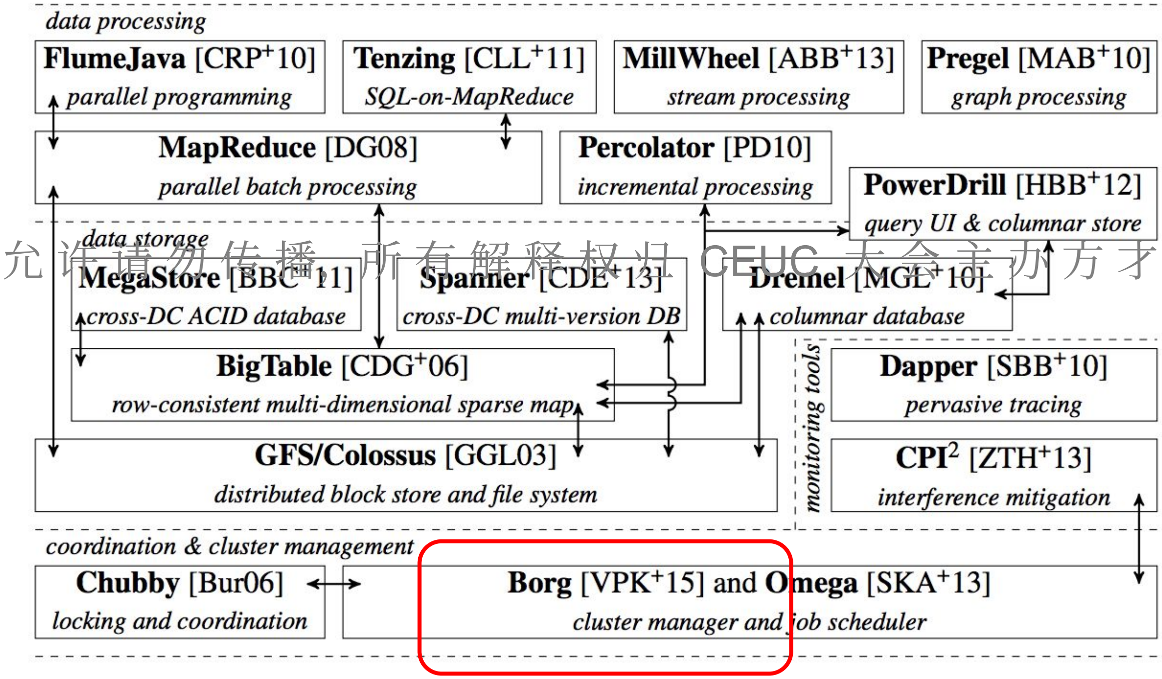
What is Kubernetes?

It all started in Google

- Use container since day 1
    - primary goal: save money - VM is heavy
    - high-density and performance
    - fast to start
- Start Container Journey
    - 2004 - ?
    - use container for decade
- Everything runs in container
    - even for storage system
    - >2B container a week

Google 内部容器主要是以 cgroup 为主的资源限制，与 docker 不同。

It all started in Google



data processing

FlumeJava [CRP$^+$10]
parallel programming

Tenzing [CLL$^+$11]
SQL-on-MapReduce

MillWheel [ABB$^+$13]
stream processing

Pregel [MAB$^+$10]
graph processing

MapReduce [DG08]
parallel batch processing

Percolator [PD10]
incremental processing

PowerDrill [HBB$^+$12]
query UI & columnar store

data storage

MegaStore [BBC$^+$11]
cross-DC ACID database

Spanner [CDE$^+$13]
cross-DC multi-version DB

Dremel [MGL$^+$10]
columnar database

BigTable [CDG$^+$06]
row-consistent multi-dimensional sparse map

GFS/Colossus [GGL03]
distributed block store and file system

monitoring tools

Dapper [SBB$^+$10]
pervasive tracing

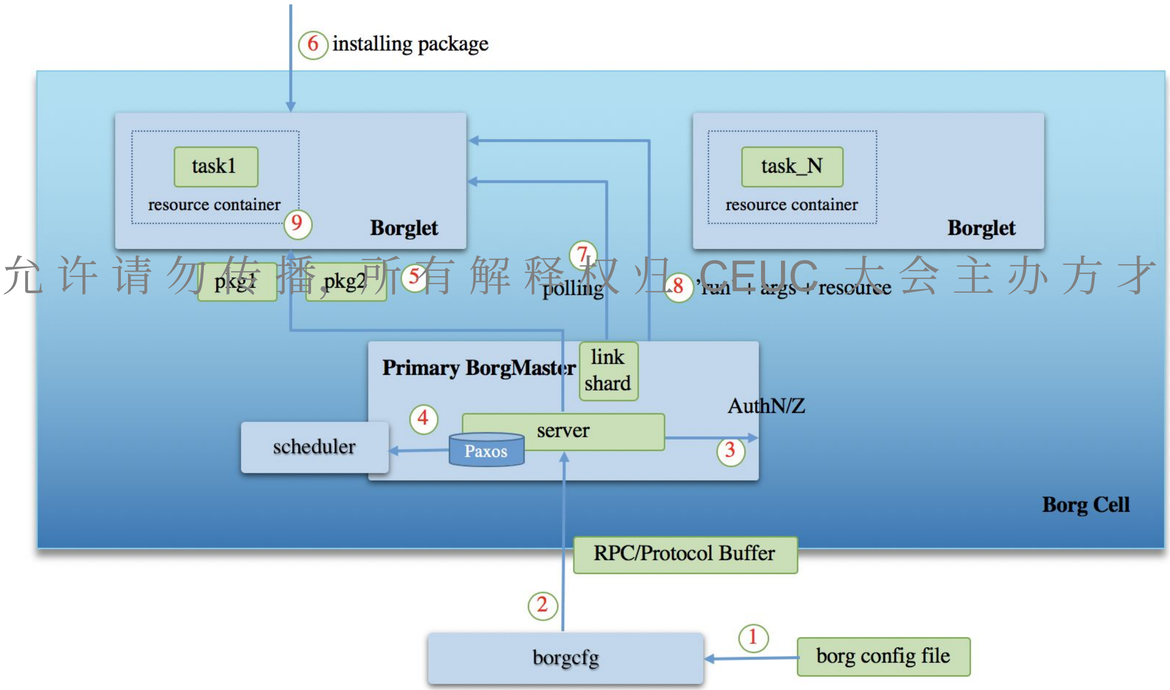
CPI^2 [ZTH$^+$13]
interference mitigation

coordination & cluster management

Chubby [Bur06]
locking and coordination

Borg [VPK$^+$15] and **Omega** [SKA$^+$13]
cluster manager and job scheduler

Borg at a glance



Borg at a glance

- 高稳定性、高自动化、高智能
- 极其复杂的配置文件
- 最流行语言排行榜？
  - Go, Python, Java, C++, Javascript, …
- 最令人畏惧语言排行榜?
  - Python, Borgcfg, Borgmon, Shell, …
- Borg master 越来越复杂

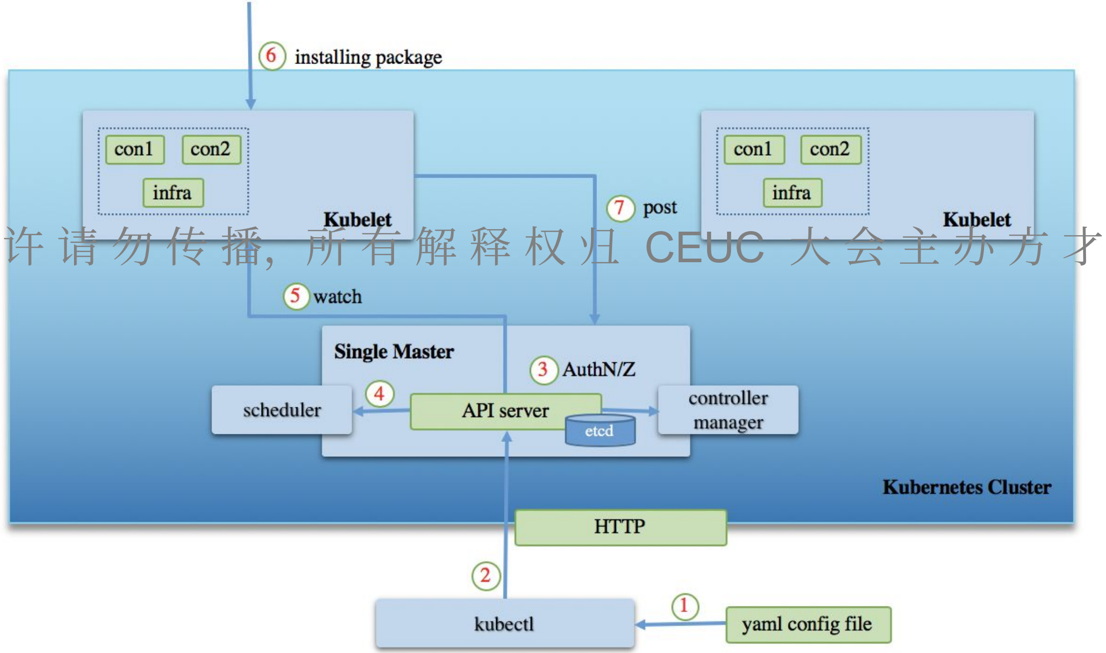
Omega at rescue

- Original Goal
  - parallelism at best
  - shared state
  - lock-free optimistic concurrency control
  - 近百人的研发团队
- 中期
  - 各种系统迁移成本太大
  - 开发缓慢
- 晚期
  - 戛然而止
  - Big shuffle

Cloud: The ‘Urs’qake

- 发展云战略
  - 经济上的巨大回报
  - 技术上的巨大领先
  - 产品化上的差异
- PaaS 与 IaaS 的矛盾
  - GAE vs GCE
  - Managed VM
  - Project 7 (被打回)

Kubernetes at a glance



What is Kubernetes?

What is Kubernetes?

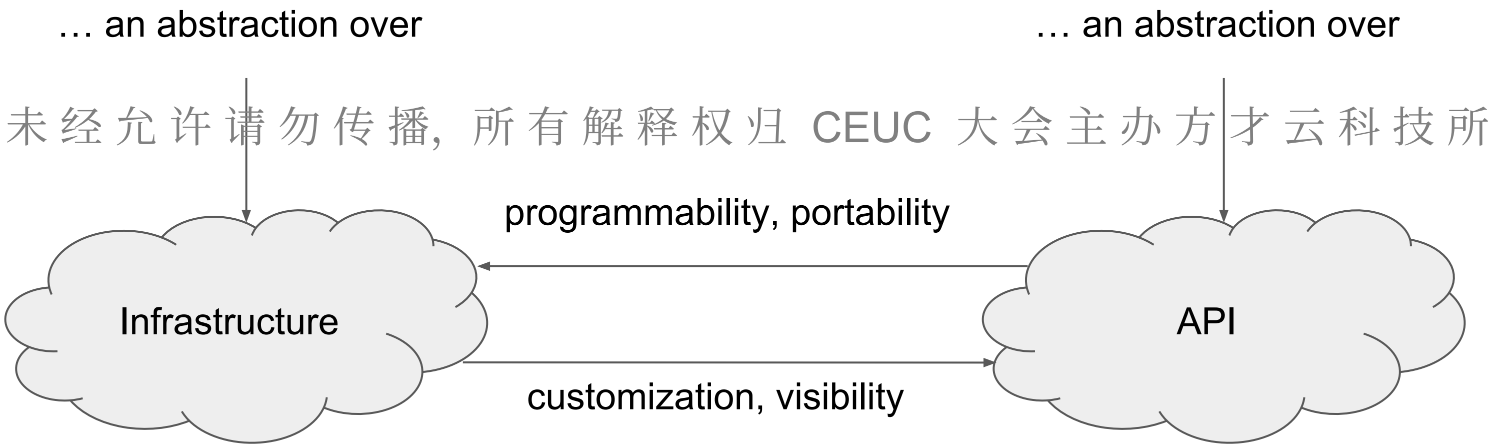
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications, and groups containers that make up an application into logical units for easy management and discovery.

What is Kubernetes?

~~Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications, and groups containers that make up an application into logical units for easy management and discovery.~~

Kubernetes is a container management ecosystem.

What is Kubernetes?



Infrastructure Abstraction & Extensibility

- Network Plugin
  - 每个公司，每个环境的网络都是一个感人的故事
  - Old Way
    - Send a PR to Kubernetes
  - Now
    - Container Network Interface (CNI)
    - Allows user to customize network as they need
  - Future
    - gRPC based API covering more than just interfaces and IPAM
    - Maybe more integration with Service (now CNI only covers Pod IP)
    - Multi-IP, Multi-Network

Infrastructure Abstraction & Extensibility

- Storage Plugin
  - 别造轮子，只求稳
  - 存在很多供应商
  - Old Way
    - Send a PR to Kubernetes
    - Flexvolume, exec based plugin
  - Now
    - Container Storage Interface (CSI)
    - gRPC based spec, now Industry Standard
  - Future
    - CSI to GA and move all plugins out of tree

Infrastructure Abstraction & Extensibility

- Device Plugin
  - 需要管理的设备越来越多
  - 存在很多供应商
  - Old Way
    - 按需提供，集中在 GPU
  - Now
    - gRPC based spec
    - 与 Kubernetes 资源管理模型紧密关联
  - Future
    - Graduate to GA

Infrastructure Abstraction & Extensibility

- Container Runtime ‘Plugin’
    - 为什么只能用 docker？想要隔离性更好的环境怎么办
    - Old way
        - Send PR to Kubernetes
    - Now

        - Container Runtime Interface (CRI)
        - gRPC based spec; adding more runtime support is much easier
    - Future
        - RuntimeClass
            - run multiple container runtime in your cluster
            - let container choose which runtime to use

Infrastructure Abstraction & Extensibility

- Scheduler ‘Plugin’
  - 调度场景太多，没法满足所有需求
  - Now
    - Scheduler extender: essentially callback
    - Multi-scheduler: run custom scheduler
  - Future
    - Scheduler v2: an extensible framework with a lot extension points

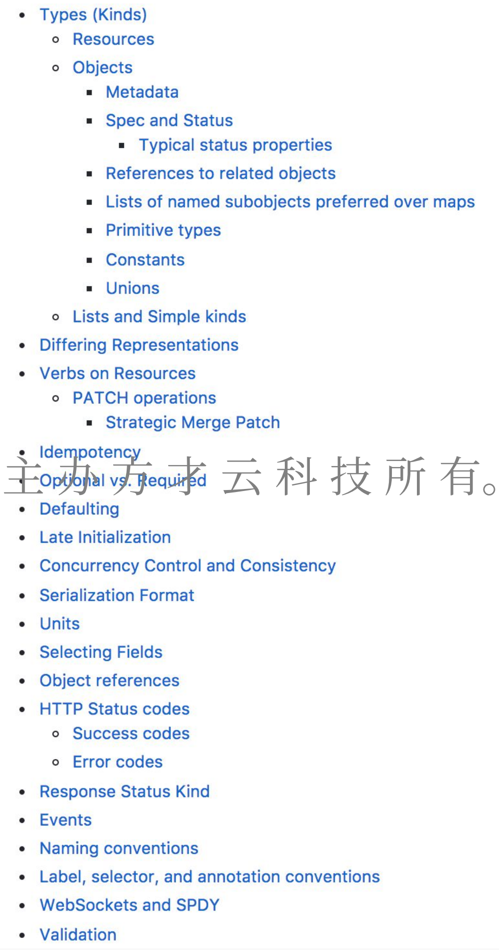
API Abstraction & Extensibility

- THE API

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
spec:
  containers:
  dnsPolicy: ClusterFirst
  nodeName: i-2zea47skez7ye2xr438v
  restartPolicy: Always
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  volumes: xxx
status:
  conditions: xxx
  hostIP: 10.44.164.150
  phase: Running
  podIP: 192.168.79.9
  startTime: 2016-11-22T14:54:57Z
```

Resource API version

```
apiVersion: v1
kind: Node
metadata:
  labels:
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/os: linux
    kubernetes.io/hostname: minikube
  name: minikube
  resourceVersion: "1027609"
  selfLink: /api/v1/nodesminikube
  uid: 21ffcb42-0f69-11e7-a9ff-080027e561ba
spec:
  externalID: minikube
status:
  addresses:
  - address: 192.168.99.101
    type: LegacyHostIP
  - address: 192.168.99.101
    type: InternalIP
  - address: minikube
    type: Hostname
```



- Types (Kinds)
  - Resources
  - Objects
    - Metadata
    - Spec and Status
      - Typical status properties
    - References to related objects
    - Lists of named subobjects preferred over maps
    - Primitive types
    - Constants
    - Unions
  - Lists and Simple kinds
- Differing Representations
- Verbs on Resources
  - PATCH operations
    - Strategic Merge Patch
- Idempotency
- Optional vs. Required
- Defaulting
- Late Initialization
- Concurrency Control and Consistency
- Serialization Format
- Units
- Selecting Fields
- Object references
- HTTP Status codes
  - Success codes
  - Error codes
- Response Status Kind
- Events
- Naming conventions
- Label, selector, and annotation conventions
- WebSockets and SPDY
- Validation

API Abstraction

- Controller
    - THE very core, fundamental, important, essential concept in Kubernetes
- Conceptually:
    - Kubernetes ~= API + Controller
- A lot components are built around the concept
    - Scheduler, Kubelet, Kube-proxy, Deployment, Autoscaling, Cloudproviders …..

API Abstraction

- Webhooks!
    - Mutating webhooks
    - Nonmutating webhooks
    - Authn webhooks
    - Authz webhooks
    - Policy webhooks
    - etc

What is Kubernetes?

What is Kubernetes?

What is Kubernetes?

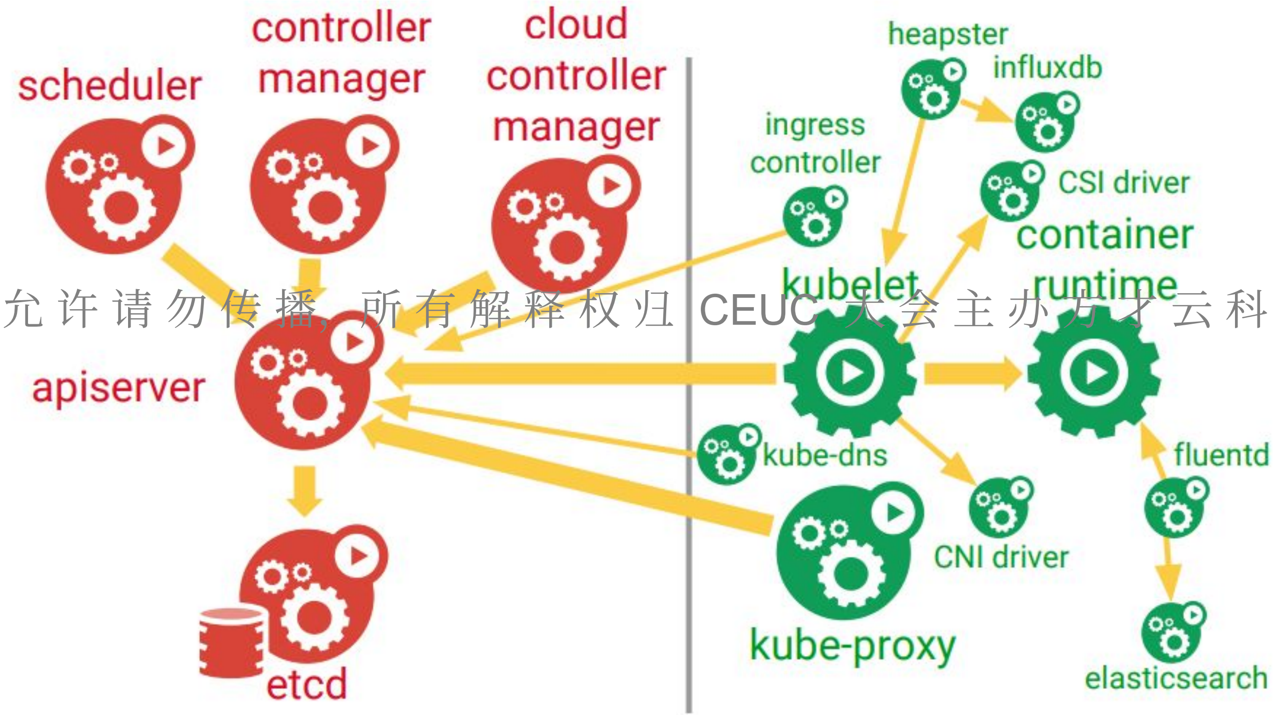
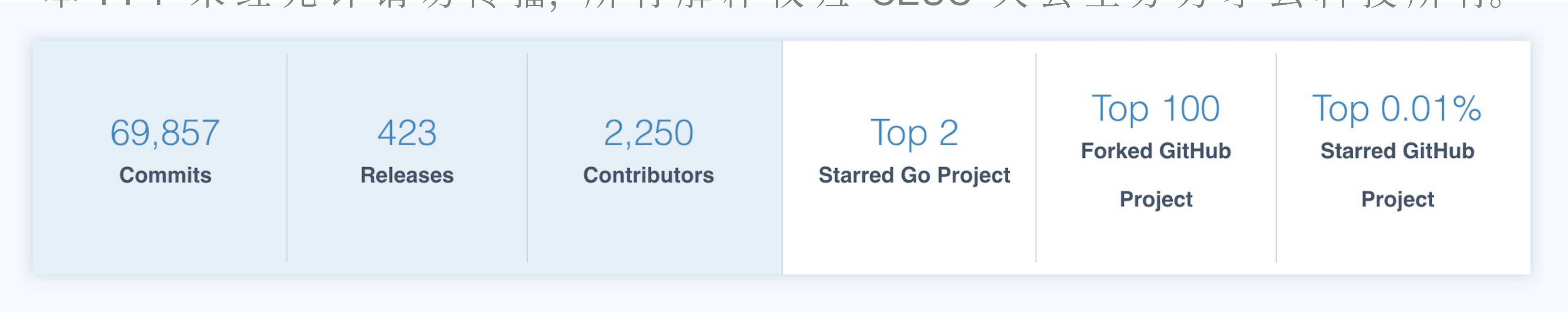


Image Source: Kubecon 2017, Austin

What is Kubernetes?

- Still a young project
- Becoming more and more complex
- But at its core
    - a set of cooperating microservices



69,857	423	2,250	Top 2	Top 100	Top 0.01%
Commits	Releases	Contributors	Starred Go Project	Forked GitHub Project	Starred GitHub Project

Image Source: Apprenda

What is Kubernetes?

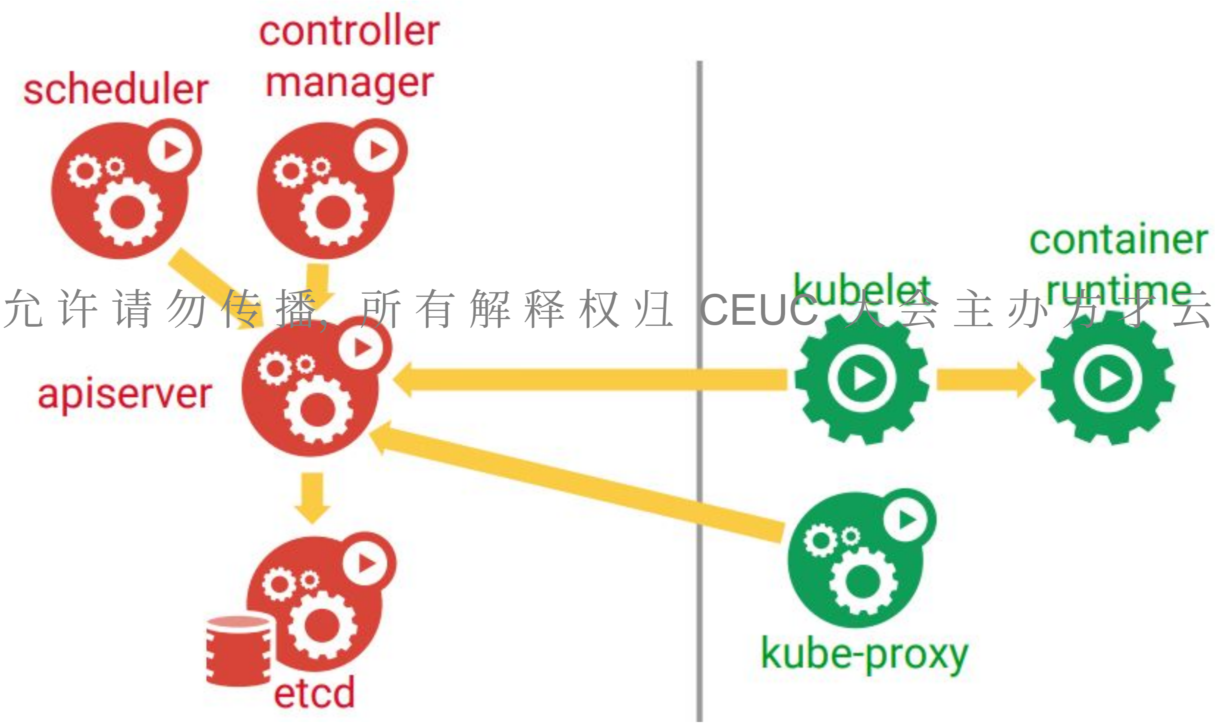
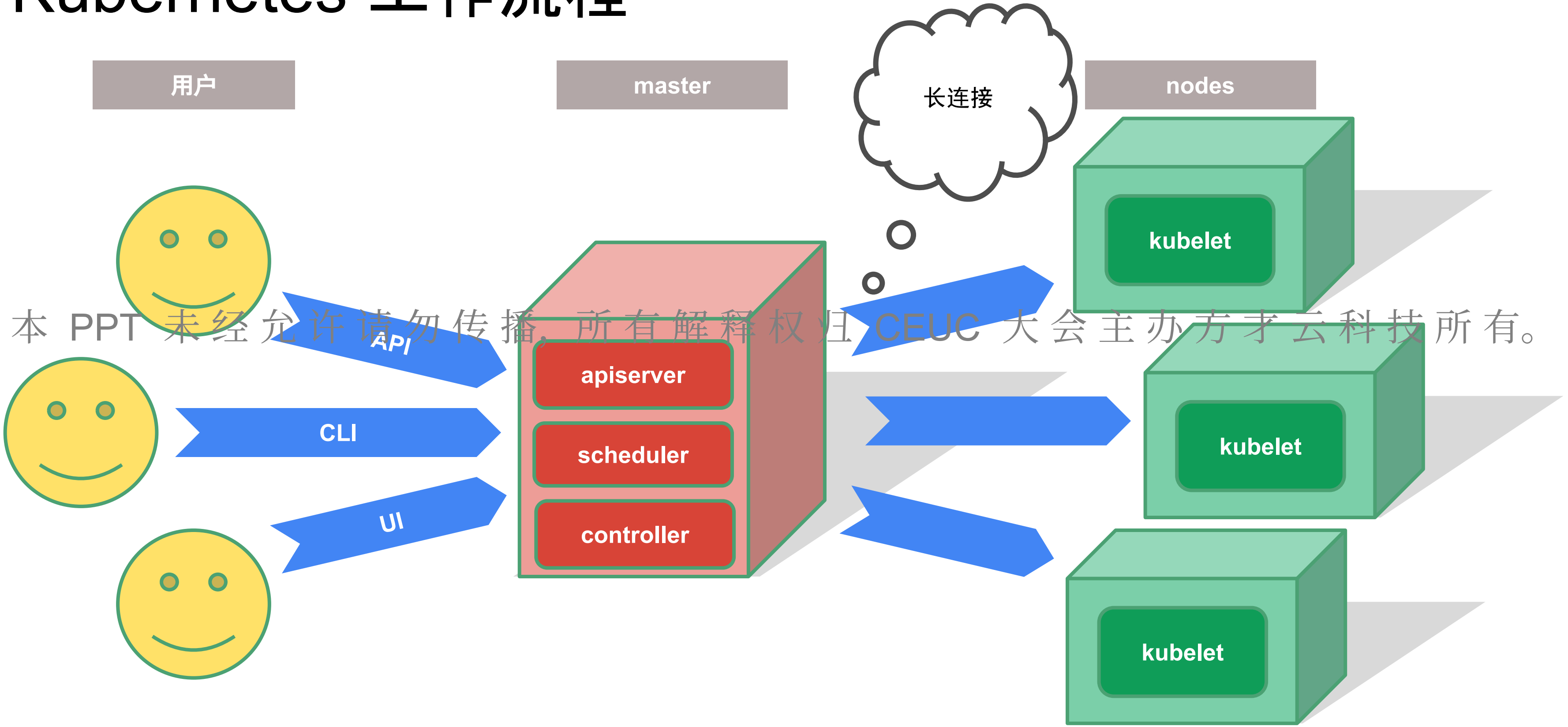


Image Source: Kubecon 2017, Austin

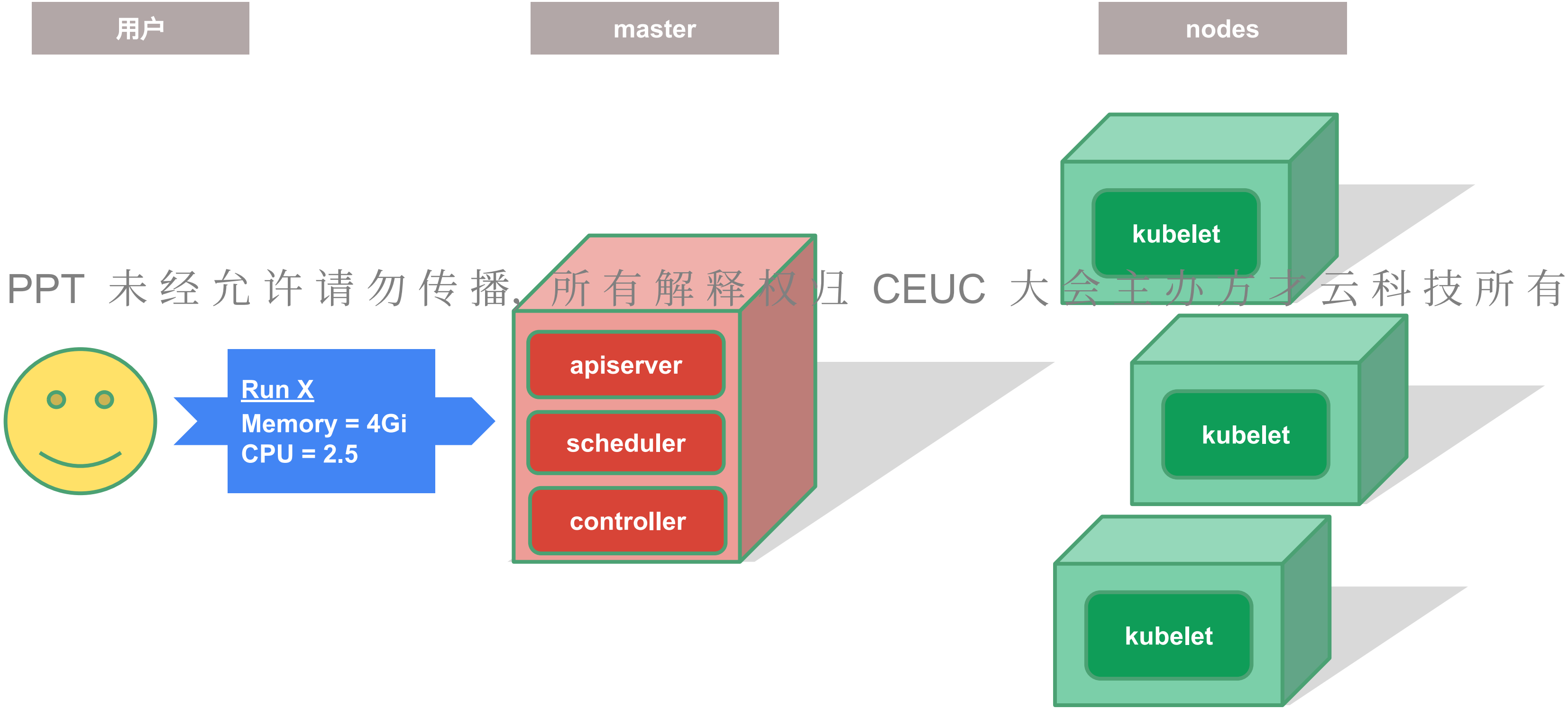
What is Kubernetes? - Design principle

- Declarative -> imperative: State your desired result, let the system actuate
  - e.g. I (user) want a volume with size 10G backed by Ceph
- Control loops: Observe, Rectify, Repeat
  - e.g. Ack! I will find a volume with at least 10G from Ceph, and if failed, will create one for you
- Simple > Complex: Try to do as little as possible
  - e.g. I will find a volume from Ceph, but I won't attach it; it's someone else's problem
- Modularity: Components, interfaces & plugin
  - e.g. Ceph plugin vs Glusterfs plugin
- Legacy Compatible: Meet users where they are
  - e.g. requiring apps to change is a non-starter
- Open > Closed: Open Source, Standards • e.g. JSON, REST, etc

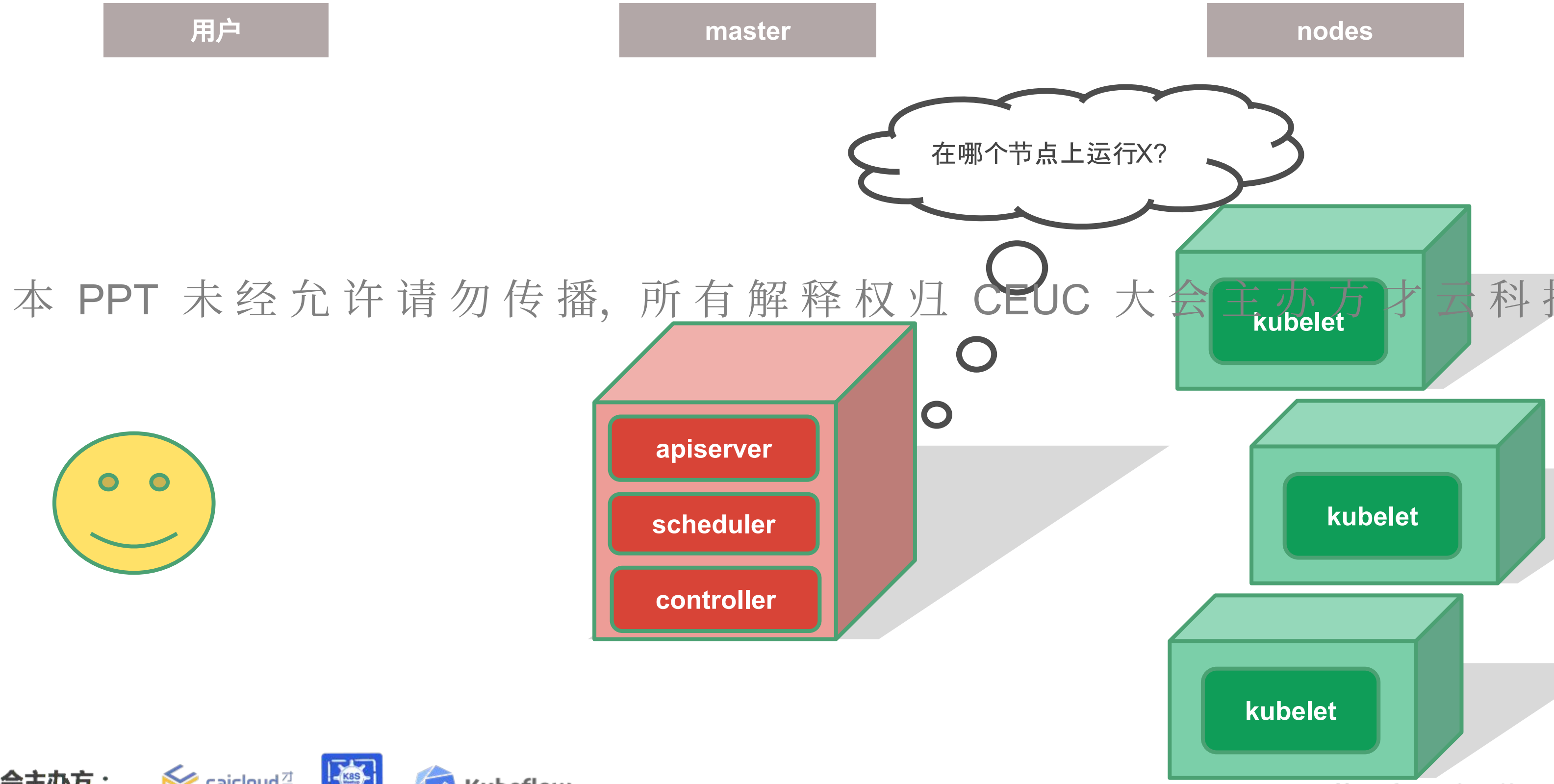
Kubernetes 工作流程



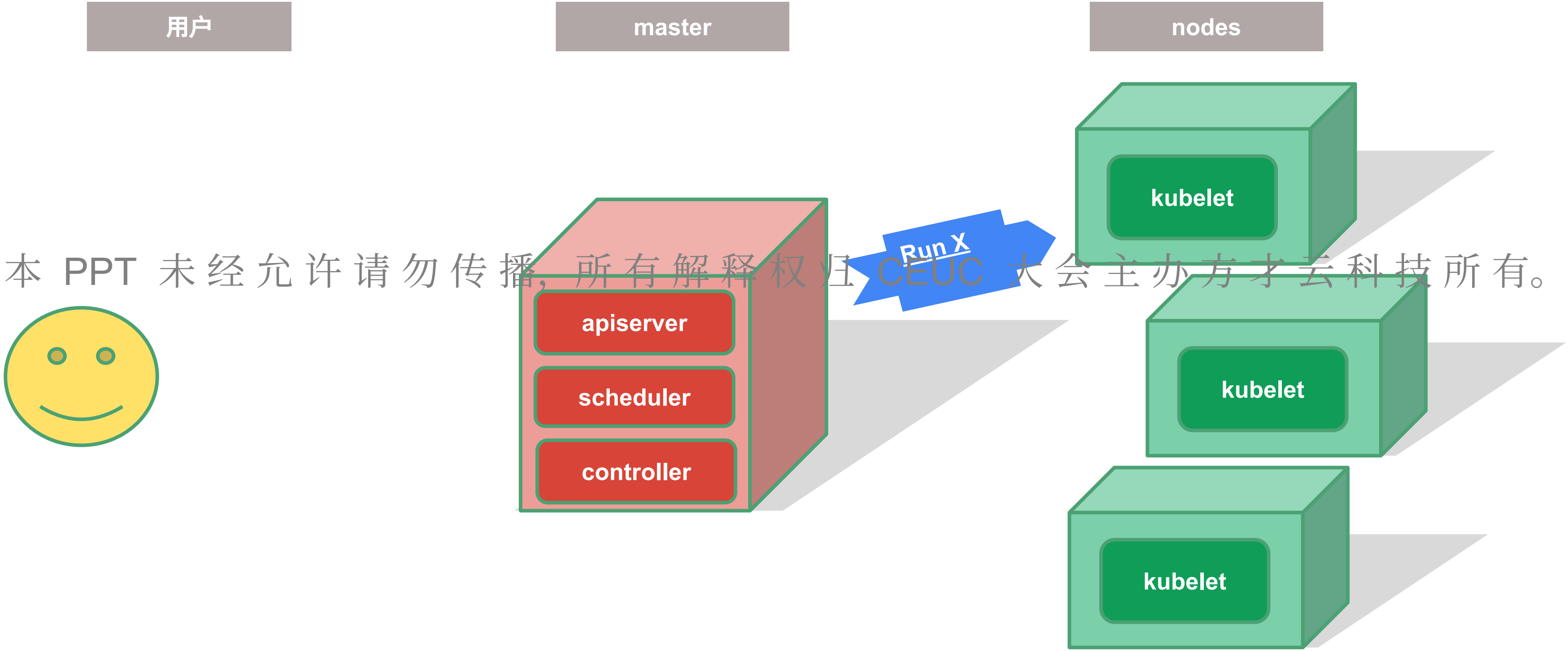
Kubernetes 工作流程



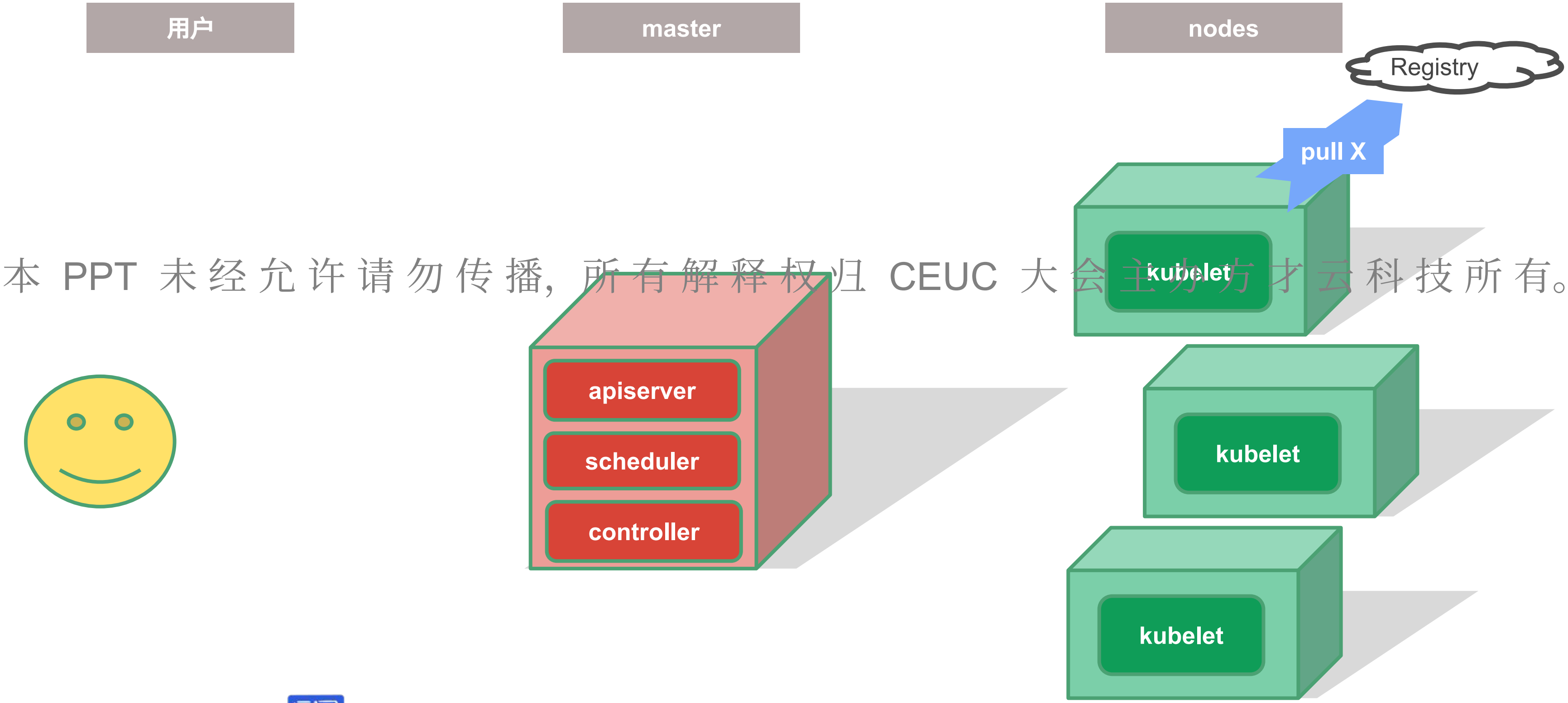
Kubernetes 工作流程



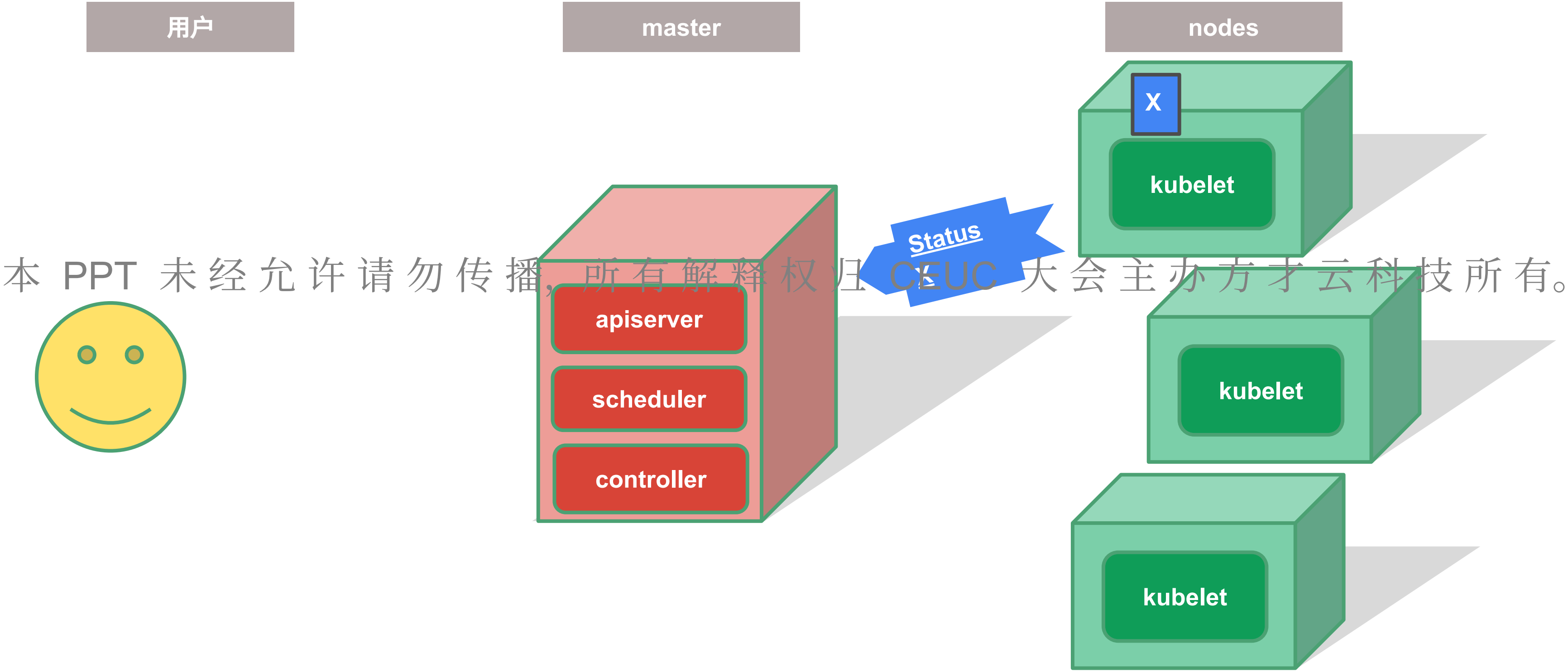
Kubernetes 工作流程



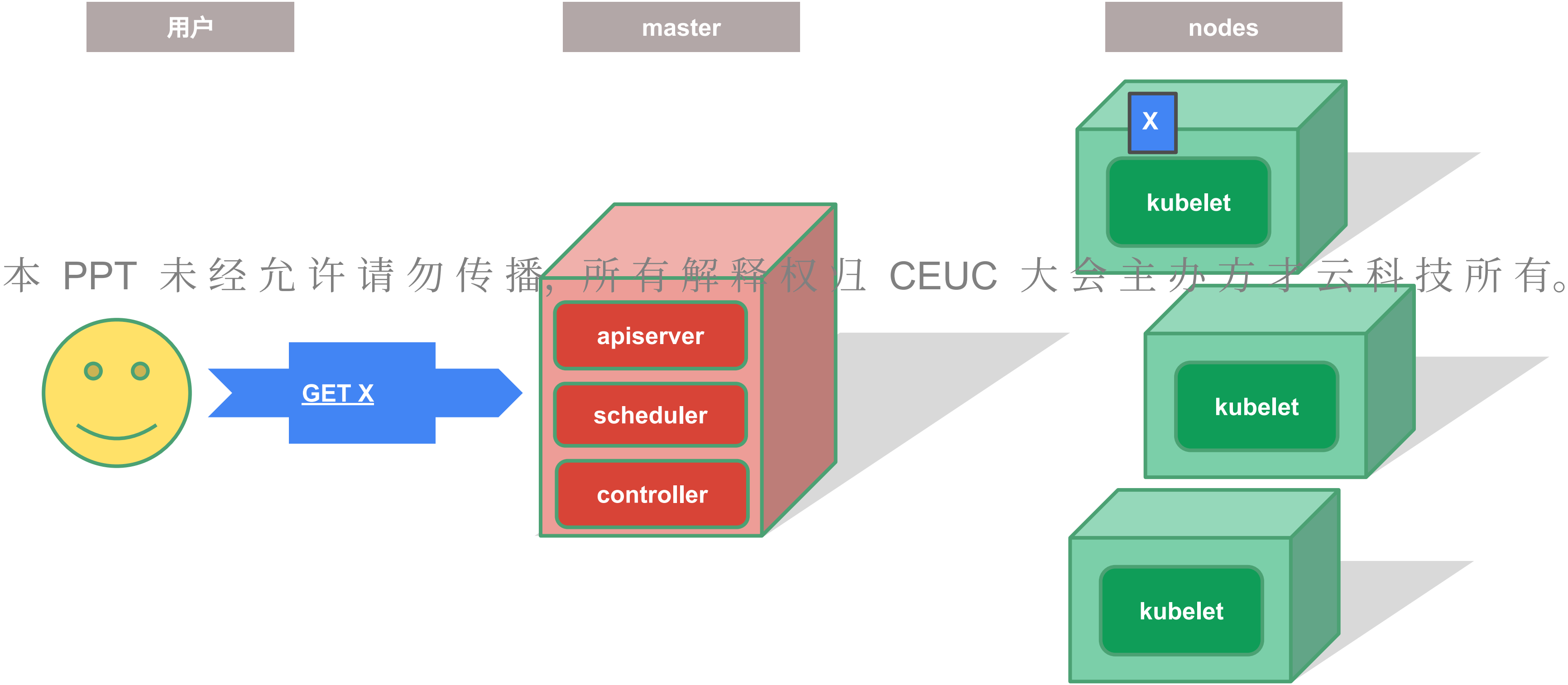
Kubernetes 工作流程



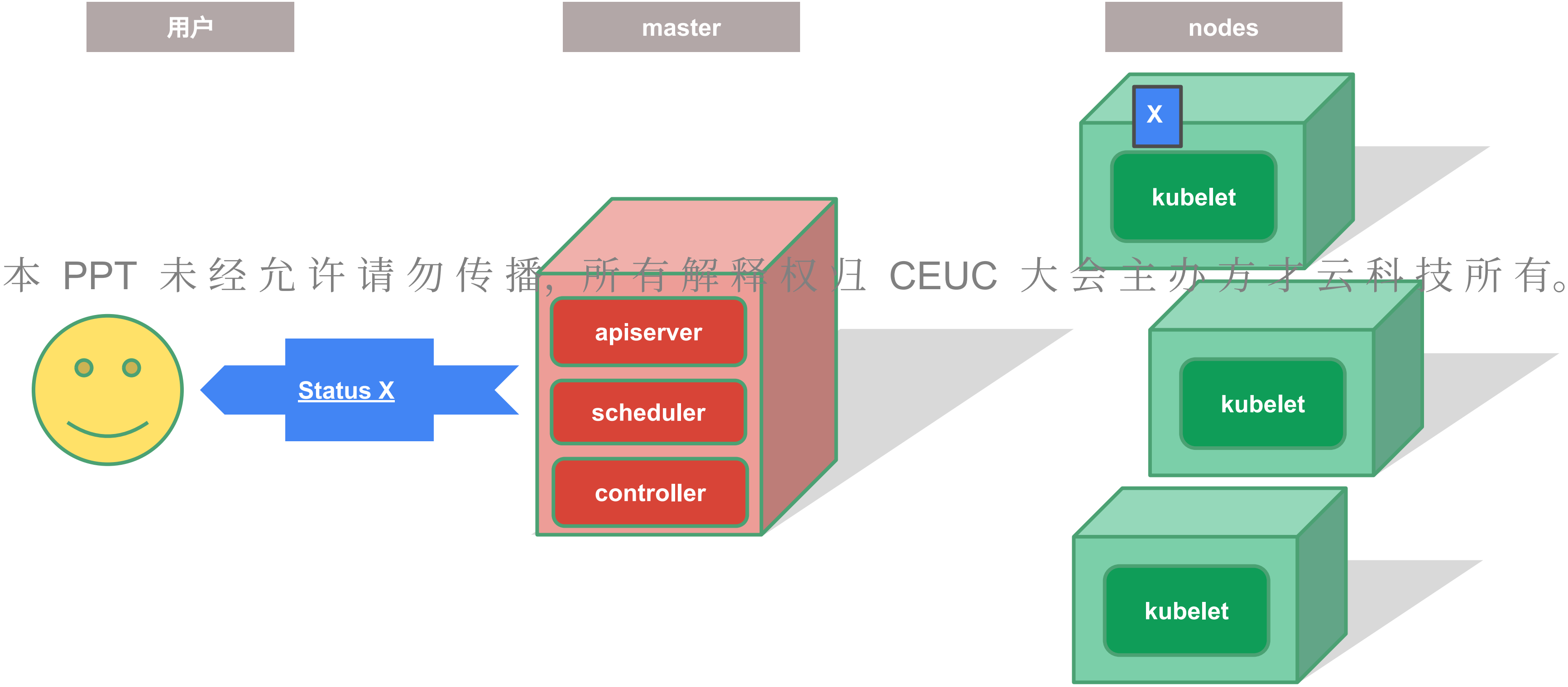
Kubernetes 工作流程



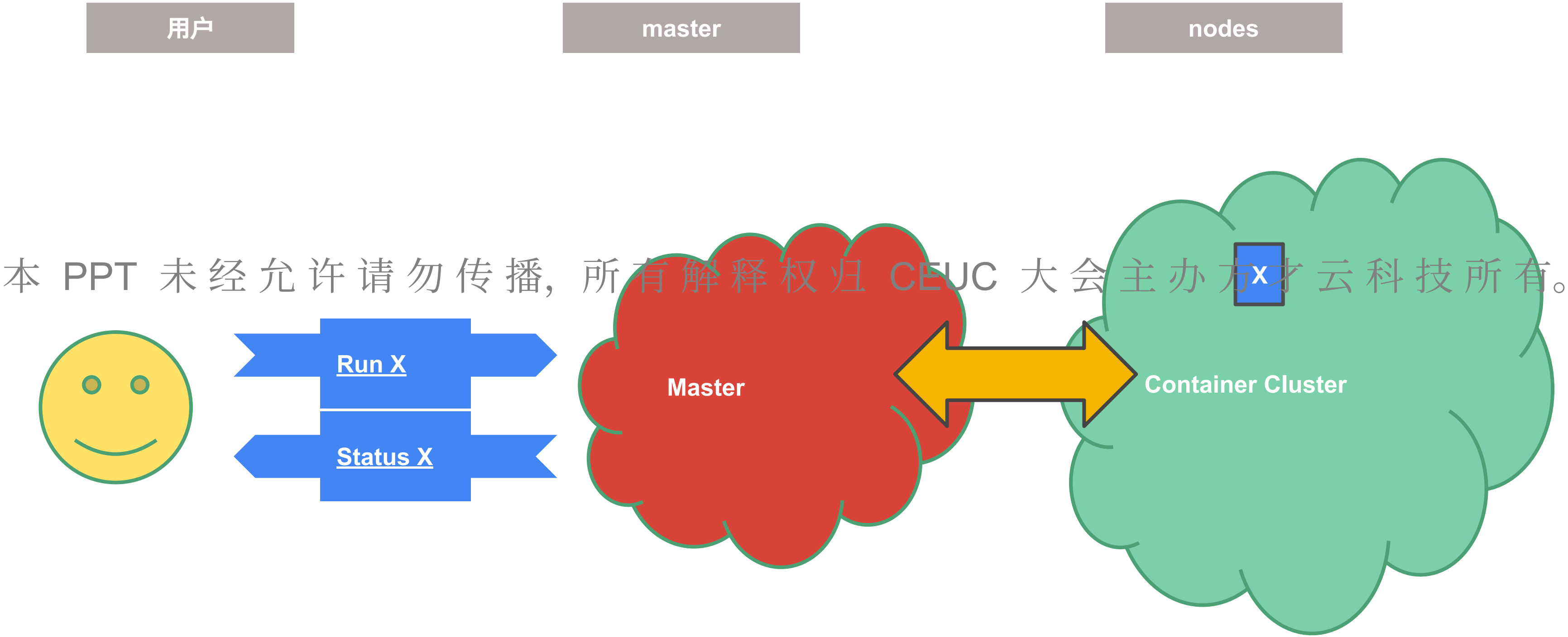
Kubernetes 工作流程



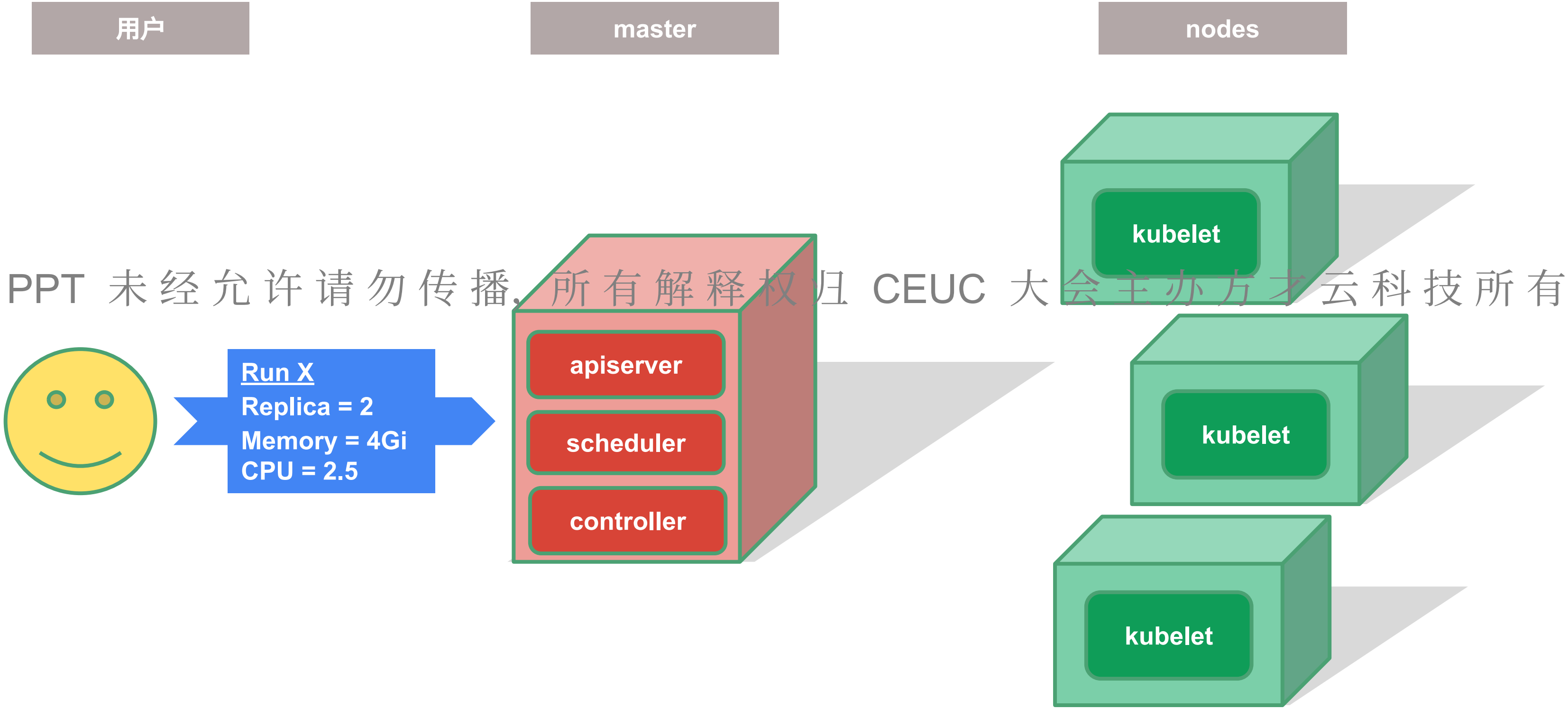
Kubernetes 工作流程



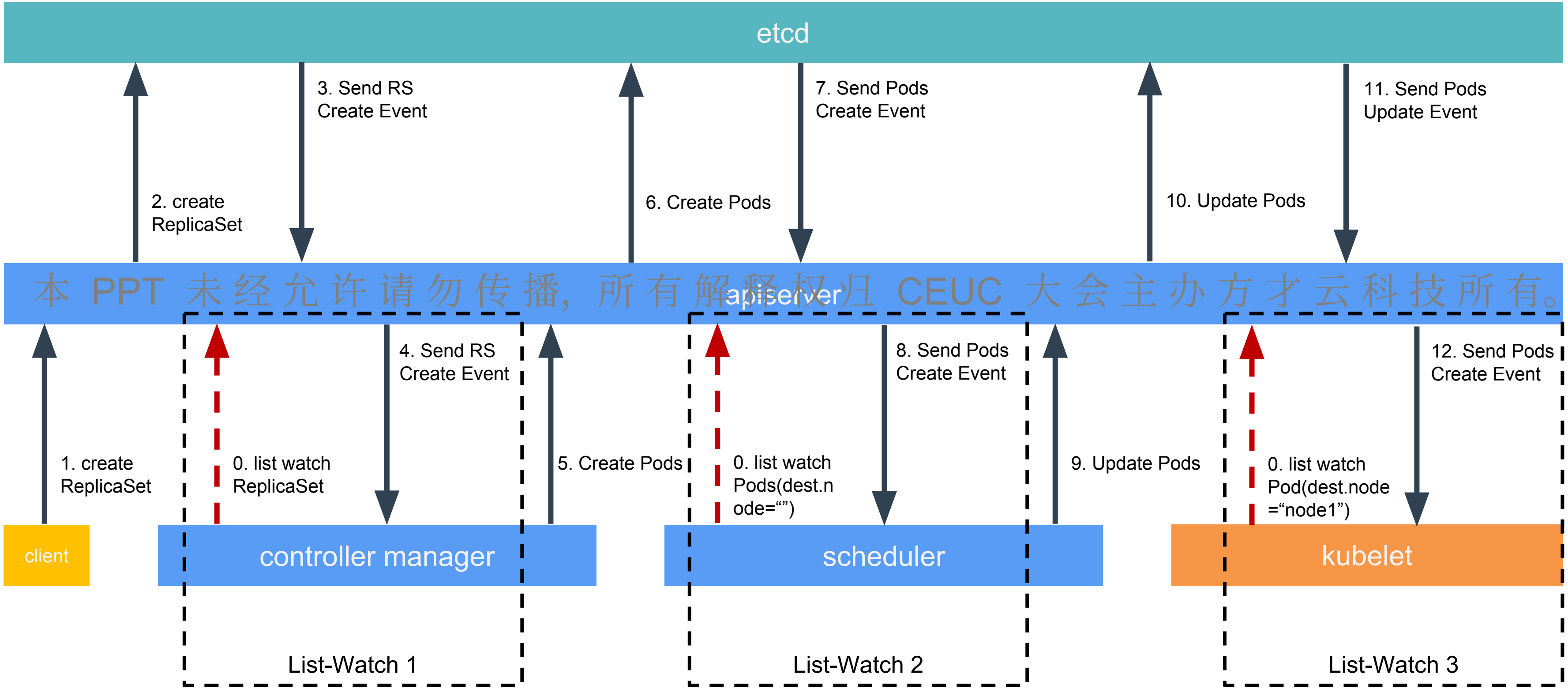
Kubernetes 工作流程



Kubernetes 工作流程?



Kubernetes 工作流程



感谢聆听