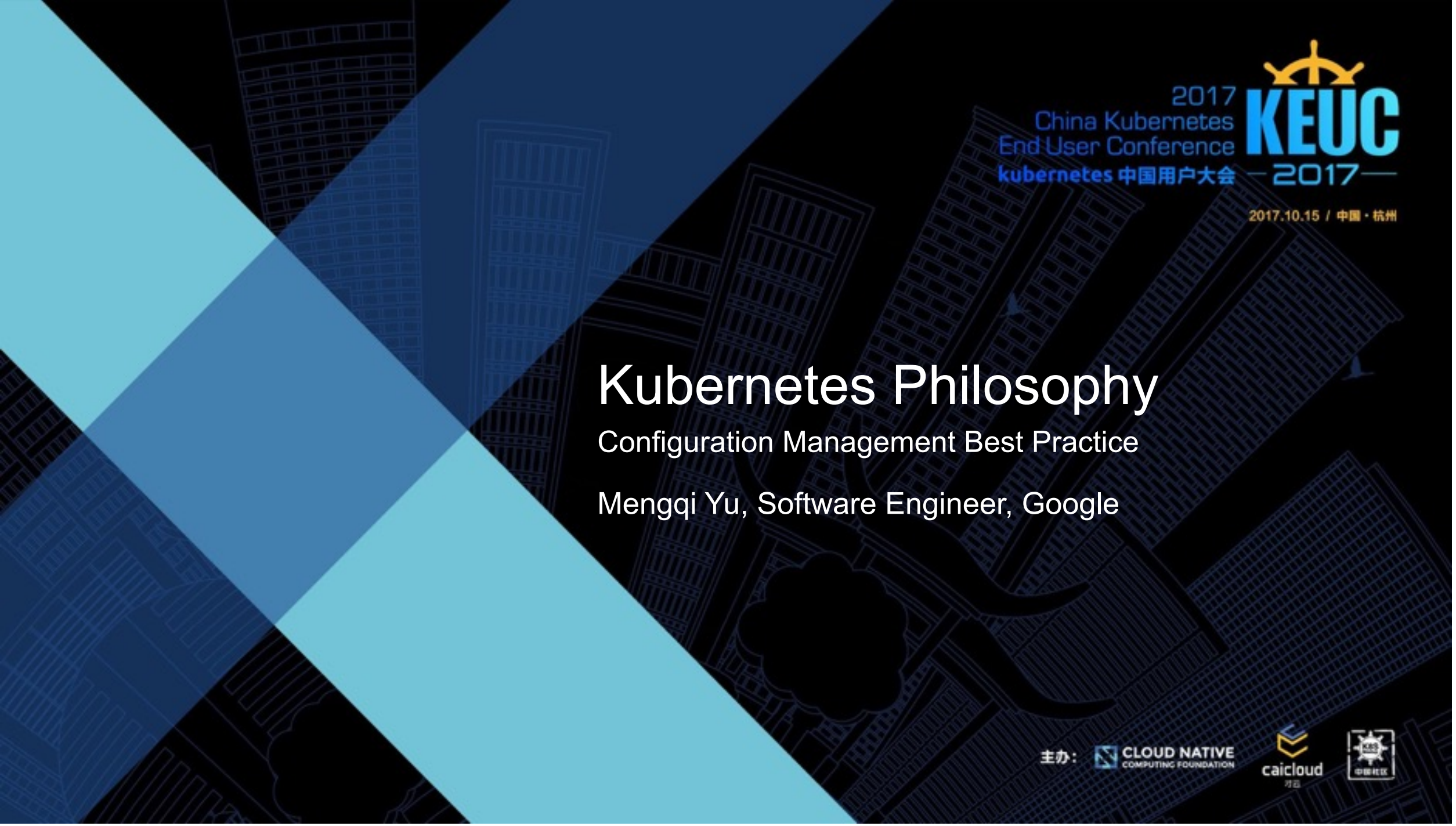

2017
China Kubernetes
End User Conference
kubernetes 中国用户大会
KEUC
2017
2017.10.15 / 中国・杭州
Kubernetes Philosophy
Configuration Management Best Practice
Mengqi Yu, Software Engineer, Google
主办:
CLOUD NATIVE
COMPUTING FOUNDATION
caicloud

Kubernetes Concepts

Kubectl Commands

Configuration Management

Demo

Declarative App Management

Kubernetes Concepts

- Declarative
  - But kubernetes support imperative
  - Support self-healing
- Level-triggered
- Asynchronous

Declarative vs Imperative

Declarative:

Definition: user specifies what the desired state is. The system will know what to do to move current state to desired state.

user: set desired state to 3

observed: 2

system: +1

observed: 3

Imperative:

Definition: user tells the system what to do.

user: +1

system: +1

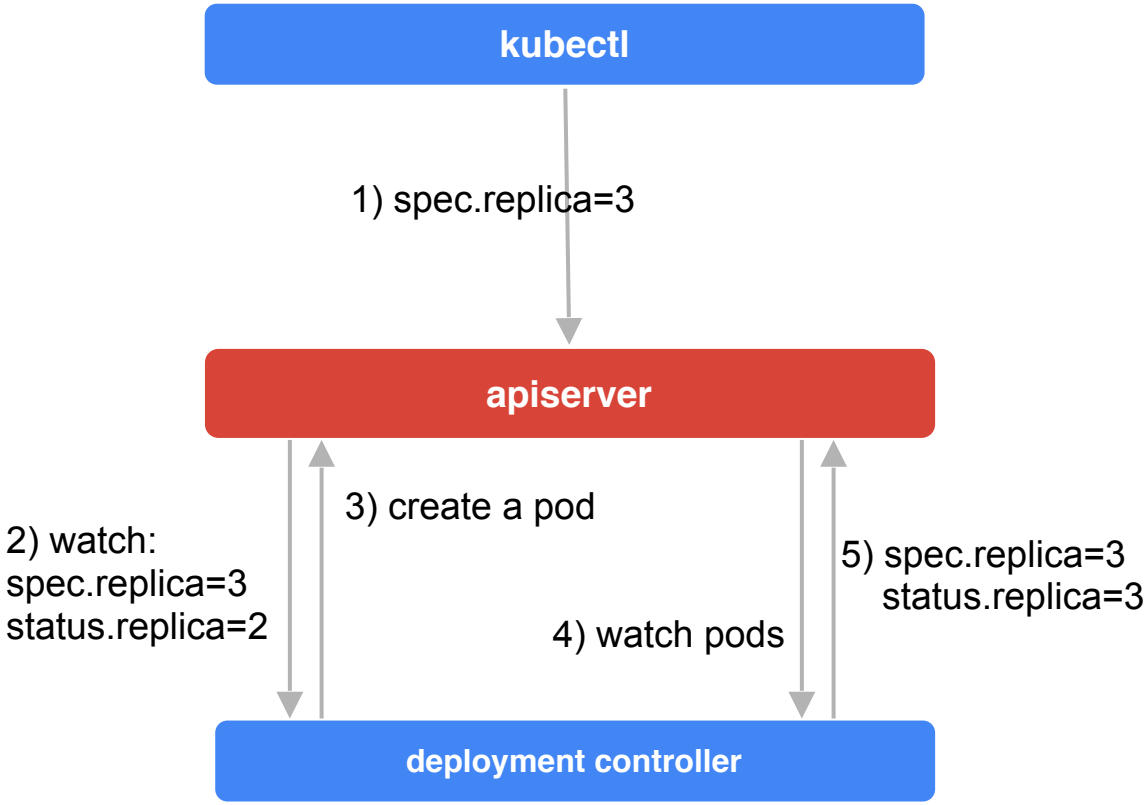
Spec vs Status

Most kubernetes API objects have Spec and Status fields.

- Spec: user desired state
  - Used by reconciliation loops to update other objects in the cluster or external objects.
- Status: system observed state
  - Used by clients and reconciliation loops to view the state of the object.

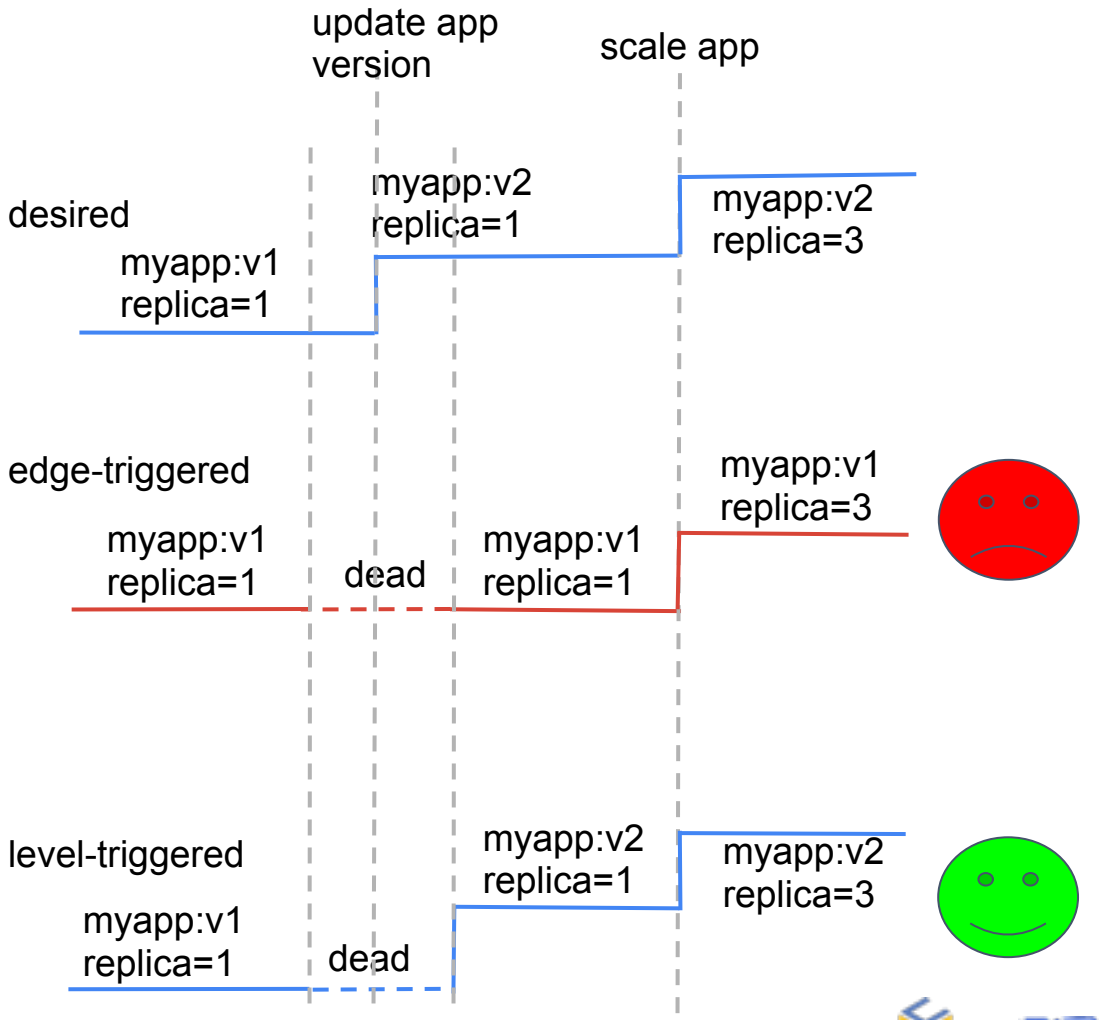
Reconcile

- Controller watches the resources.
- Desired state changes.
- Controller is notified of an update and compares current desired vs current observed states.
- Controller responds to actual-desired mismatch by executing operations (e.g. create a Pod, allocate clusterIp).
- Observed state matches desired state.



Level-triggered vs Edge-triggered

- Kubernetes is level-triggered
- Edge-triggered
  - Pros: straightforward and easy to implement.
  - Cons:
    - missing edges may result in wrong states.
    - does not take most direct route to current desired state
- Level-triggered
  - Pros: don't need to worry about missing the edge.
  - Cons: harder to implement.



Kubernetes Concepts

Kubectl Commands

Configuration Management

Demo

Declarative App Management

Imperative Commands

- **Imperative commands**: kubectl supports imperative create / update / delete commands that parse command line flags into yaml / json to send to the server.
  - This hides the underlying object configuration for the users.
  - e.g. kubectl run, expose
  - e.g. kubectl create deployment
  - e.g. kubectl delete <resource_kind>/<resource_name>

Imperative Object Configuration

- **Object configuration**: objects are defined in yaml / json with syntax defined in reference docs.
- **Imperative object configuration**: kubectl supports imperative create / update / delete commands that work directly on **object configuration** stored locally or in scm.
  - e.g. kubectl create -f
  - e.g. kubectl replace -f
  - e.g. kubectl delete -f

Imperative Commands vs Imperative Object Configuration

- Imperative commands
  - Pros: straightforward, easy to learn and easy to remember.
  - Cons:
    - no change review processes.
    - no audit trails.
    - not provide a template.
- Imperative object configuration
  - Pros:
    - integrate with a source control system, e.g. git.
    - integrate with reviewing changes before push processes.
    - integrate with audit trails.
  - Cons:
    - requires basic understanding of the object schema.
    - need to write the yaml files.

Declarative Object Configuration

- **Declarative object configuration**: kubectl supports declarative workflow that works directly on **object configuration**.
  - User does not specify create / update / delete.
  - Command figures out the operation.
  - e.g. kubectl apply --prune -f (a.k.a configuration reconciliation)

- Imperative object configuration
  - Pros: simpler and easier to understand.
  - Cons:
    - best on files, not directories.
    - updates by other writers (e.g. controller) MUST be reflected in configuration files.

- Declarative object configuration
  - Pros:
    - updates to live objects by other writers are retained. - e.g. autoscalers change replicas
    - works with directories
    - automatically choose operations (create, patch, delete)
  - Cons:
    - behavior is harder to understand

Kubernetes Concepts

Kubectl Commands

Configuration Management

Demo

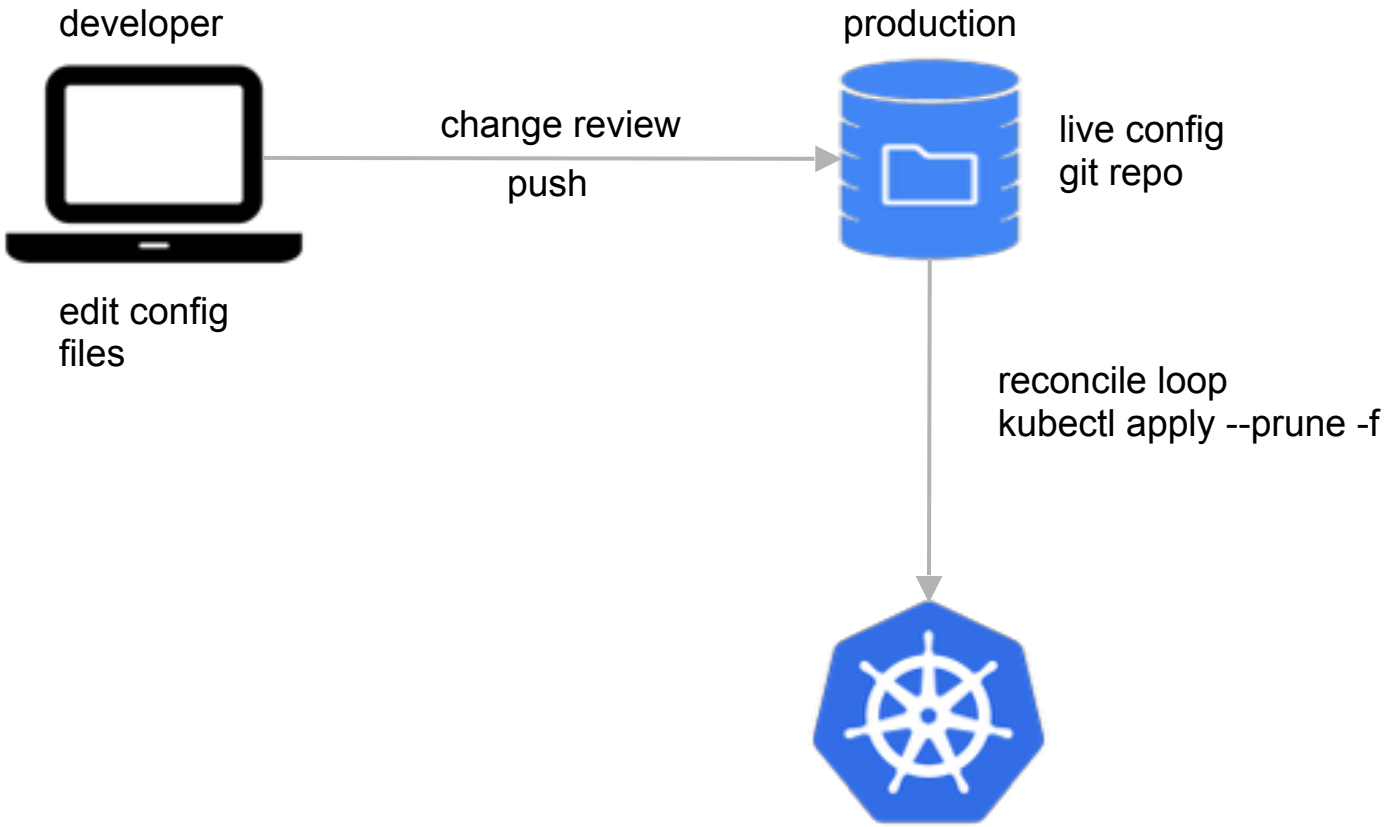
Declarative App Management

Configuration Management

- Beginning user
  - Start with imperative commands
  - Work with imperative object configuration
- Experienced User
  - Use declarative object configuration
  - Use version control
  - Use change review process

- Beginning user
  - Kubernetes API is the only store for desired state.
  - Support some number of revision history
  - But no long history, no cross-object history
- Experienced User
  - Revision control contains user defined desired state.
  - Support cross-object history.
    - Multiple deployments that depend on each other: tag the whole repository of cluster state at known good points, can roll back entire cluster by applying state at that tag.
  - Auto reconciliation e.g. [kube-applier]

Workflow


developer
production
change review
push
live config
git repo
edit config
files
reconcile loop
kubectl apply --prune -f


2017
China Kubernetes
End User Conference
kubernetes 中国用户大会
KEUC
2017
2017.10.15 / 中国 · 杭州
Demo
主办:
CLOUD NATIVE
COMPUTING FOUNDATION
caicloud

Kubernetes Concepts

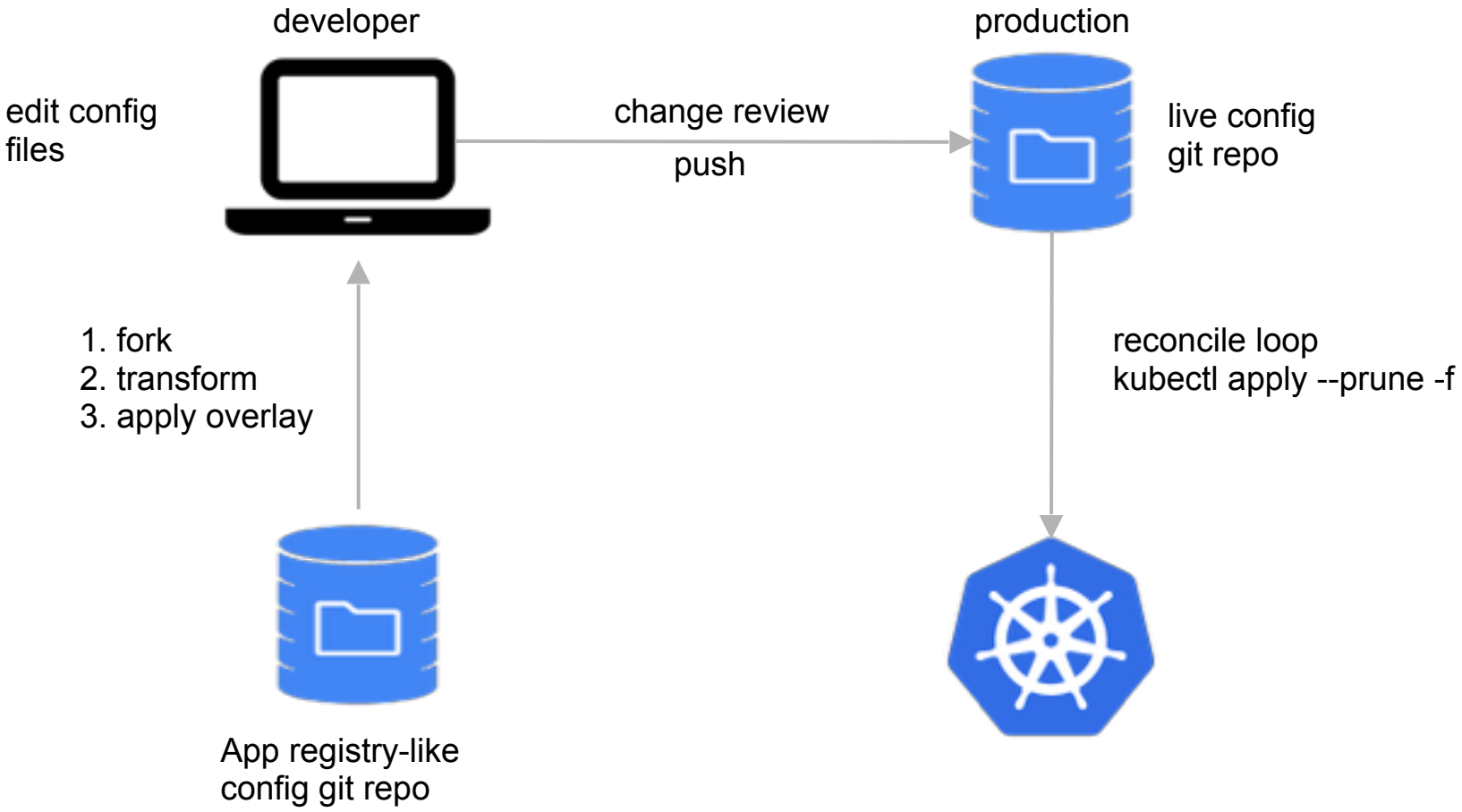
Kubectl Commands

Configuration Management

Demo

Declarative App Management

Declarative Application Management Workflow



Declarative Application Management (DAM)

- Lookup the off-the-shelf application in an app registry
- Fork
- (Optional) Make change of it: labels, name prefix
- (Optional) Create multiple instances of overlays (e.g. dev, staging, prod)
- Check them in your git repo
- CD system auto reconciliation

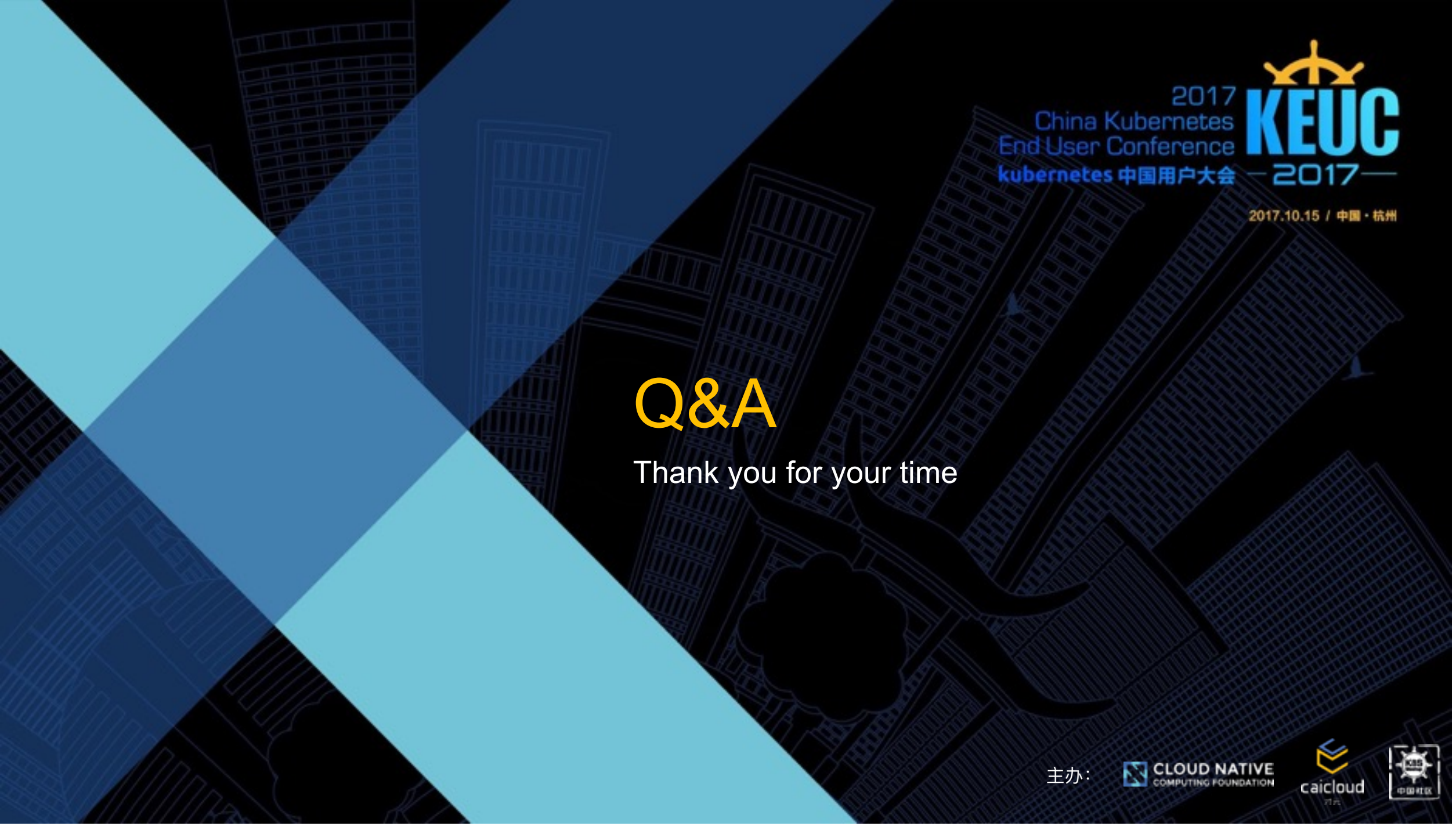
DAM is actively being explored but in early development phase.

Package Management

- Kubernetes Charts
  - Curated applications for Kubernetes.
  - http://github.com/kubernetes/charts
- Kubernetes Helm
  - A tool for manage kubernetes packages.
  - https://github.com/kubernetes/helm
- Challenge: it's hard to find the balance between enough parameters to satisfy all users, without the number of parameters becoming hard to maintain and understand
  - e.g. Jenkins chart

Recap

- Declarative vs Imperative
- Spec vs Status
- Kubectl commands
- Configuration management
  - Version control system


2017
China Kubernetes
End User Conference
kubernetes 中国用户大会
KEUC
2017
2017.10.15 / 中国·杭州
Q&A
Thank you for your time
主办:
CLOUD NATIVE
COMPUTING FOUNDATION
caicloud