
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

Navigating Kubernetes Cloud Provider

Pengfei Ni
Microsoft Azure

About Me

- Principal Software Eng Mgr @Microsoft
- Networking and Cloud Provider of Azure Kubernetes Service
- Kubernetes ecosystem for 8+ years
- Opensource Kubernetes handbook at kubernetes.feisky.xyz
- @feiskyer on Github

What and Why

What is Cloud Provider in Kubernetes

- ”Cloud Provider” have multiple meanings in different contexts
- Most commonly it refers to the infrastructure provider that Kubernetes is deployed on, for example: AWS, Azure, GCP, OpenStack, etc.
- As part of Kubernetes components, it means the kube-controller-manager or cloud-controller-manager which implements Cloud Provider Interface (CPI)

Why Kubernetes Cloud Provider

- Implement CPI to enable Kubernetes' interaction with cloud infrastructures through a set of cloud controllers
- Four cloud controllers
  - Node: initialize Node's status (IP addresses, zone, region) and remove taint node.cloudprovider.kubernetes.io/uninitialized (which is added by kubelet)
  - Node Lifecyle: update and delete Nodes that have been removed or shutdown on the cloud platform
  - Route: setup routes for nodes and enable connections between containers running on different nodes
  - Service: provision cloud LoadBalancer for LoadBalancer typed services

Why out-of-tree

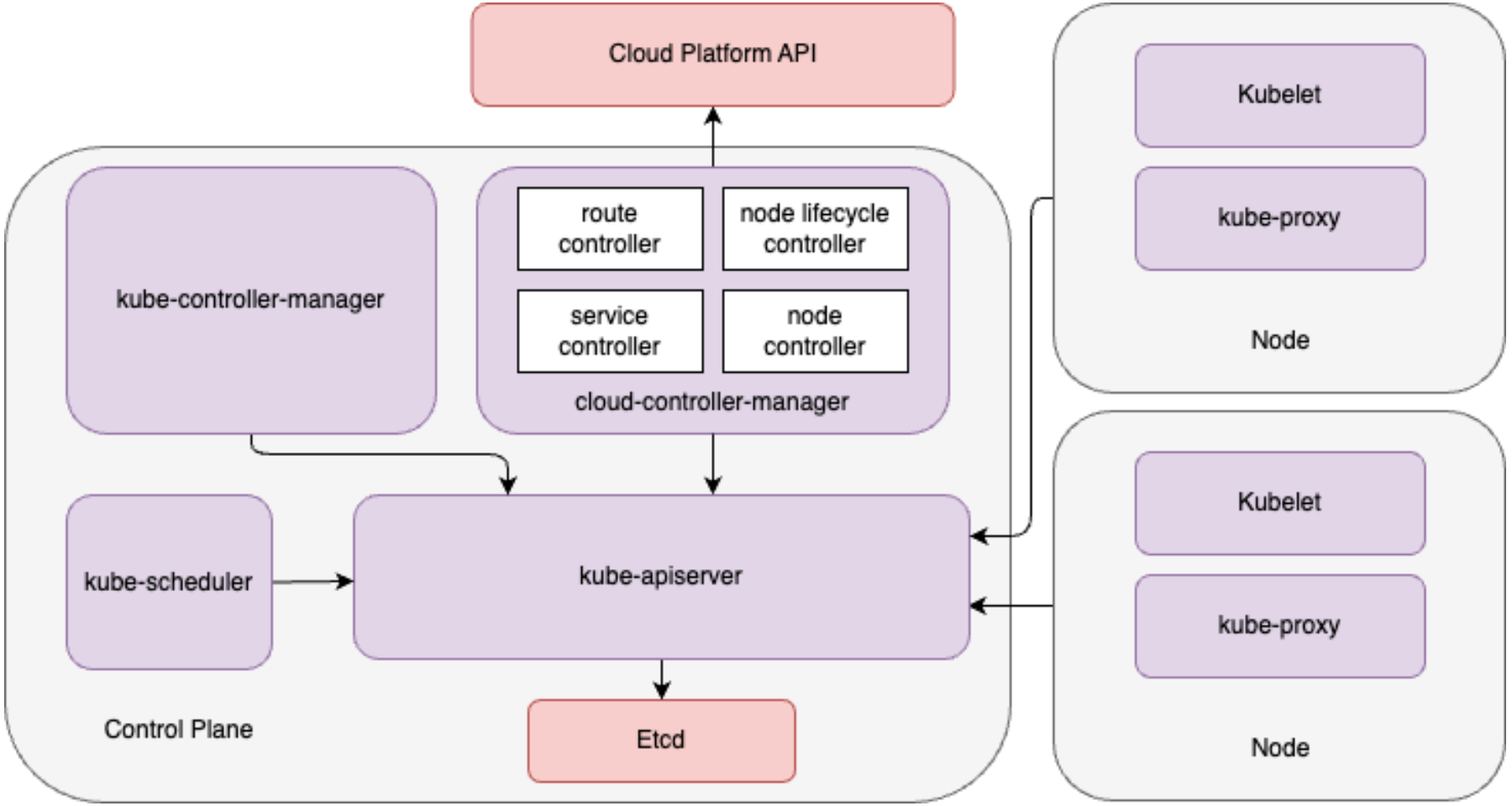
- Acronyms
  - CPI: Cloud Provider Interface
  - KCM: kube-controller-manager (in-tree)
  - CCM: cloud-controller-manager (out-of-tree)
- As introduced in KEP-1179 and KEP-2395, the Kubernetes community is working to migrate the cloud provider code from core k/k repo into separate cloud specific repos, which is called “moving out-of-tree”.
  - Make the core Kubernetes code easier to maintain
  - Make the infrastructure providers newfound freedom to innovate

Moving out-of-tree status

- Feature gates
  - DisableCloudProviders: v1.22 alpha, v1.29 beta
  - DisableKubeletCloudCredentialProviders: v1.23 alpha, v1.29 beta
  - “Out-of-tree” would be default in Kubernetes once they are beta
- Removing in-tree cloud providers
  - OpenStack in-tree provider removed on v1.26
  - AWS in-tree provider removed on v1.27
  - Others (Azure, GCE, vSphere) soon to be removed #2395
    - CCMs for those providers have already been GA
    - Existing clusters should migrate to CCM as they are planned for removal soon

CCM Deep Dive

CCM within the cluster

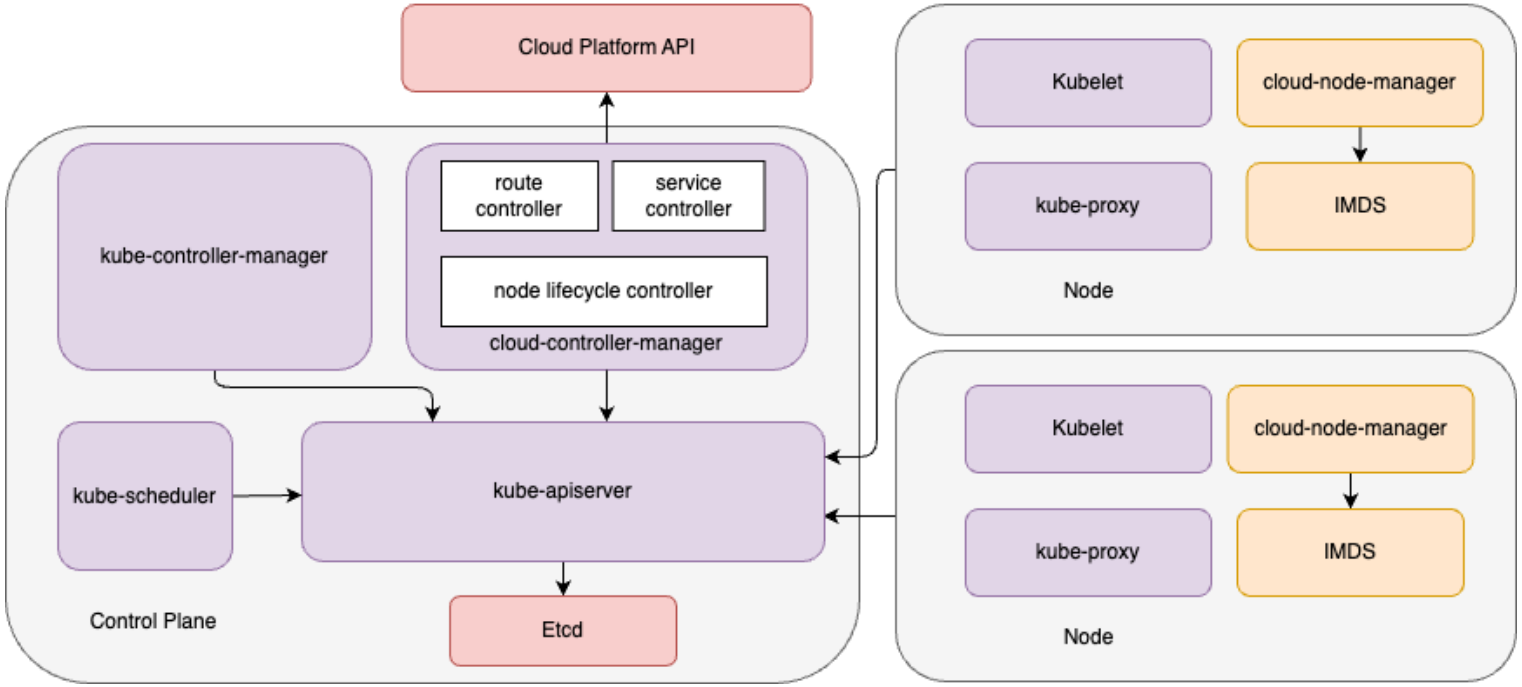


Node Controller

- Responsible for updating status and labels on Nodes.
- Sets providerID and removes the *node.cloudprovider.kubernetes.io/uninitialized* taint for new Nodes.
- Adds network addresses and hostnames to the Node status.
- Synchronizes zonal and regional information between instances and Node by applying the well-known Kubernetes labels:
  - topology.kubernetes.io/region
  - topology.kubernetes.io/zone
- The cloud-provider library also has support for converting and synchronizing the deprecated labels for zones and regions.

Node Controller on Azure

- As introduced in KEP-2328, Azure cloud APIs would be throttled when initializing large scale of nodes, hence node controller is moved out of CCM and added in cloud-node-manager daemonset, which could leverage IMDS for node initialization.



Node Lifecycle Controller

- Responsible for updating and deleting Nodes that have been removed or shutdown in the cloud provider.
- Deletes Nodes where the instances have been removed from the cloud provider.
- Adds the *node.cloudprovider.kubernetes.io/shutdown* taint when an instance is in the process of being shutdown by the cloud provider.
- Removes the *node.cloudprovider.kubernetes.io/shutdown* taint if a previously shutdown instance becomes active again by advertising the Ready condition as true.

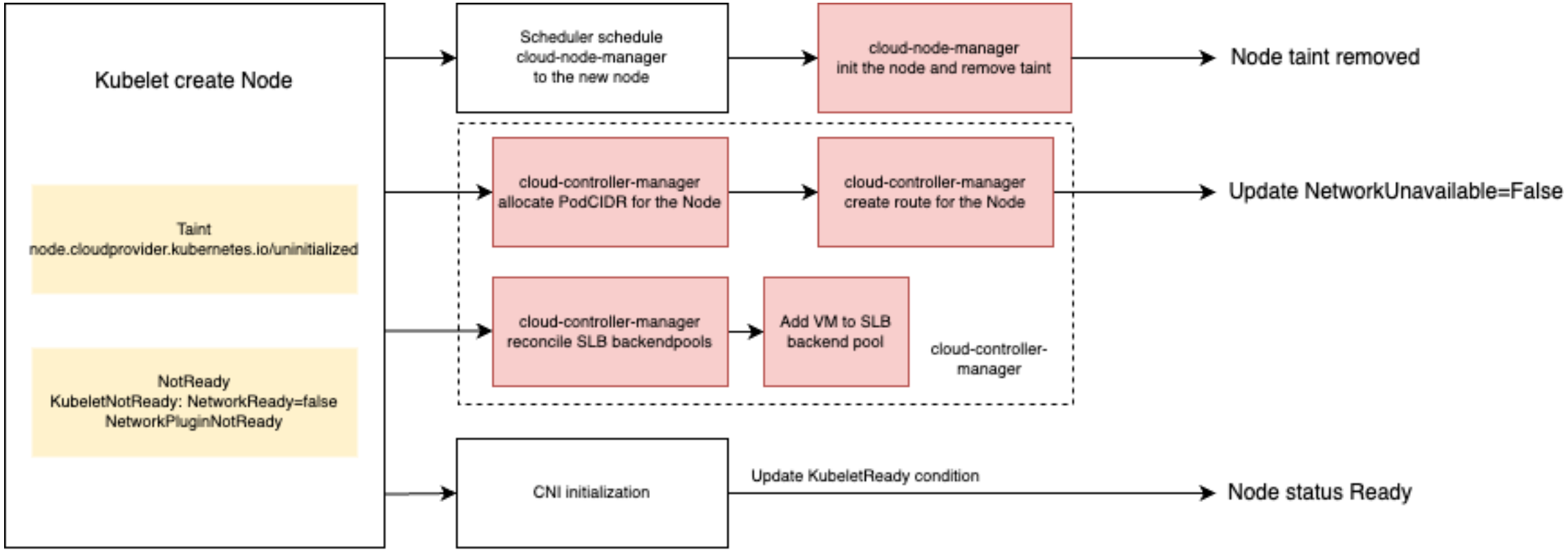
Route Controller

- Responsible for configuring routes appropriately in the cloud provider so that containers on different Nodes can communicate with each other.
- Reacts to Node addition, deletion, and update.
- Tracks Pod CIDRs on a Node to maintain network routes.
  - This behavior is highly provider specific; not all providers implement this interface at all, or in the same manner.

Service Controller

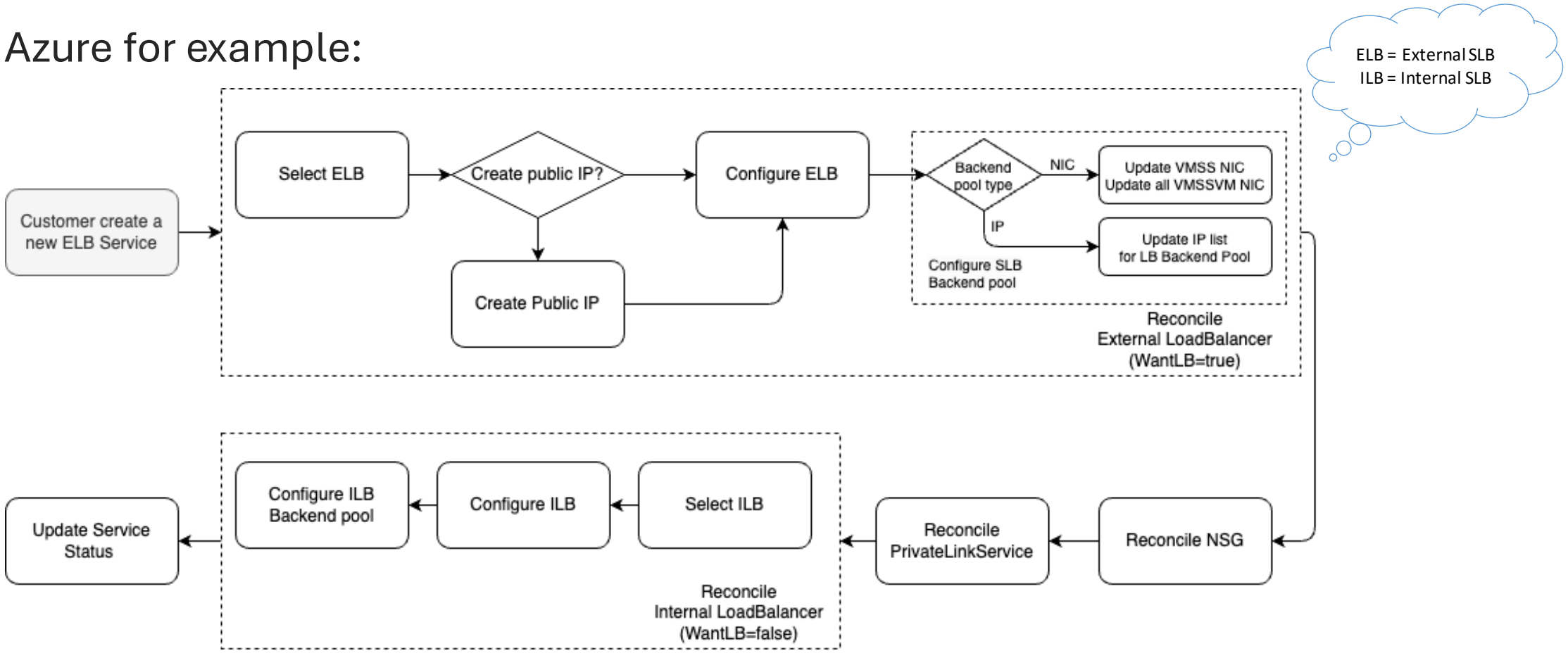
- Responsible for creating, deleting, and updating load balancers in the cloud platform to match Service objects.
- Reacts to changes in the LoadBalancer field of Service objects, creating load balancers when set to true, and deleting them when changing from true to false.
- Update load balancer configurations reacting to Service annotation changes.
  - This behavior is highly provider specific, ensure to refer the provider specific documents for the supported annotations
- Updates load balancer targets when Node objects change IP address.

Connect them together (Create Node)



Create/Update a LoadBalancer Service

- Take Azure for example:



CCM on Azure

- Implemented CPI in both KCM and CCM (KCM to be removed).
- CNM introduced as daemonset for throttling limits.
- CCM GA since v1.21 and enabled on all AKS clusters.
- Tons of annotations supported to customize Azure LoadBalancer.
- Thanks to the benefits of out-of-tree, many features and optimizations are available without changing Kubernetes core repo.
- A set of upcoming new features and enhancements, including IP-based SLB, multiple SLB, improved health probe, connection drain, VM deletion via node deletion, and more.

Operating CCM

How to run CCM

- CCM can be run anywhere, but suggested to be run on the control plane (similarly to KCM).
- To ensure high availability, CCM should be run with multiple replicas with leader election enabled.
- CNI plugins may depend on the CCM to initialize the Node addresses, hence CCM must:
  - Tolerate the *node.kubernetes.io/not-ready* taint
  - Use host networking, with a direct connection to the Kube API server

How to migrate from KCM

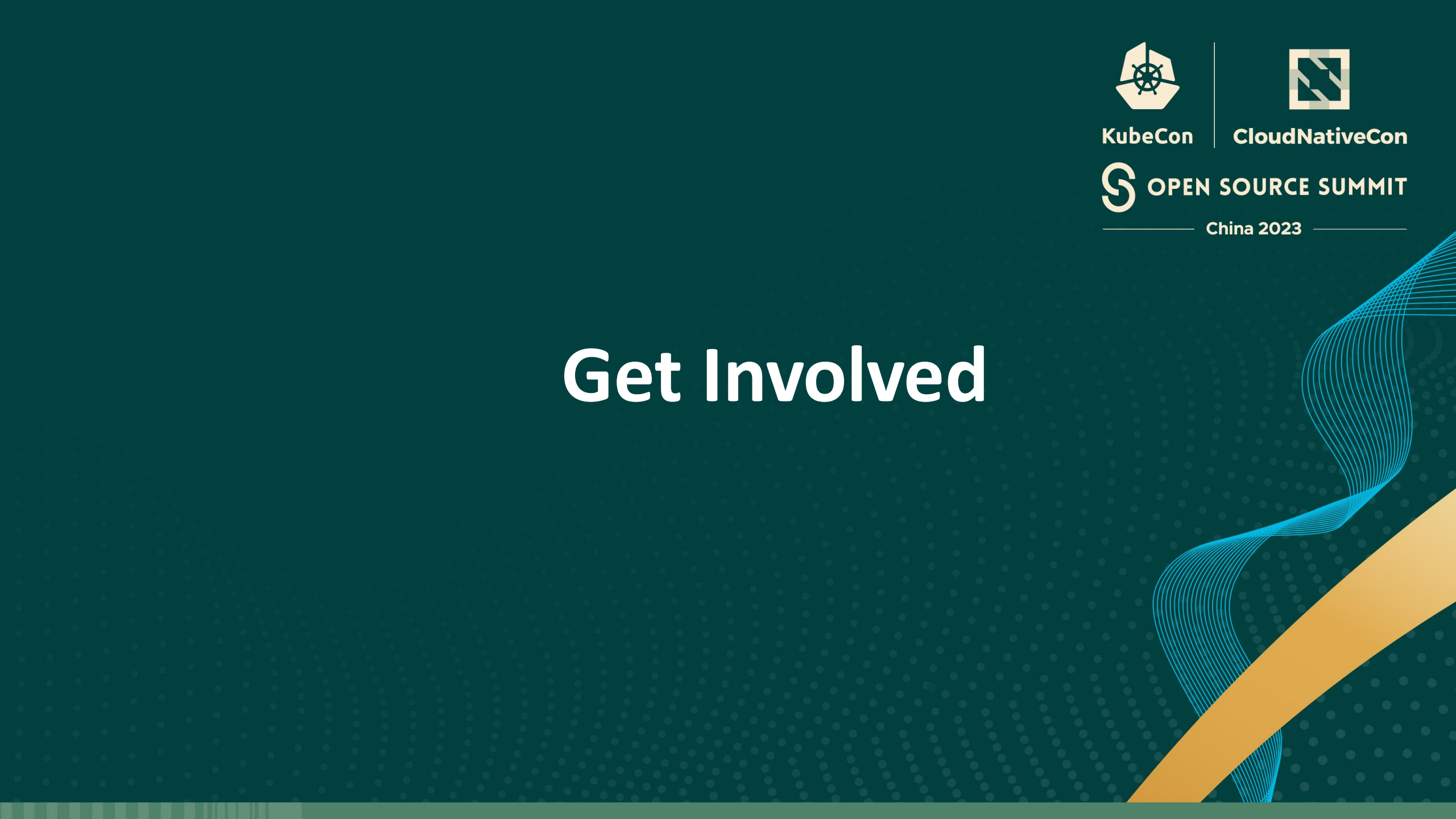
- Migration steps
  - Ensure kubelet running with *--cloud-provider=external*
  - Ensure KCM running without cloud provider
  - Enable CCM with required role bindings
  - Use the HA leader migration mechanism to aid the migration process
- Refer Migrate Replicated Control Plane To Use Cloud Controller Manager for the detailed migration steps

Build a new CCM

Build a new CCM

- SIG Cloud Provider maintains a shared library of k8s.io/cloud-provider, which could be the starting point for creating a new CCM.
  - Sample is provided for reference.
- Both CPI interfaces and CCM controllers are part of the library
  - It's not necessarily to implement all of those interfaces (return false if some are not implemented).
  - Instance/Zones are for in-tree, InstanceV2 is replacement in CCM.

```
type Interface interface {
    // Initialize provides the cloud with a kubernetes client builder and may spawn goroutines
    // to perform housekeeping or run custom controllers specific to the cloud provider.
    // Any tasks started here should be cleaned up when the stop channel closes.
    Initialize(clientBuilder ControllerClientBuilder, stop <-chan struct{})
    // LoadBalancer returns a balancer interface. Also returns true if the interface is supported, false otherwise.
    LoadBalancer() (LoadBalancer, bool)
    // Instances returns an instances interface. Also returns true if the interface is supported, false otherwise.
    Instances() (Instances, bool)
    // InstancesV2 is an implementation for instances and should only be implemented by external cloud providers.
    // Implementing InstancesV2 is behaviorally identical to Instances but is optimized to significantly reduce
    // API calls to the cloud provider when registering and syncing nodes. Implementation of this interface will
    // disable calls to the Zones interface. Also returns true if the interface is supported, false otherwise.
    InstancesV2() (InstancesV2, bool)
    // Zones returns a zones interface. Also returns true if the interface is supported, false otherwise.
    // DEPRECATED: Zones is deprecated in favor of retrieving zone/region information from InstancesV2.
    // This interface will not be called if InstancesV2 is enabled.
    Zones() (Zones, bool)
    // Clusters returns a clusters interface.  Also returns true if the interface is supported, false otherwise.
    Clusters() (Clusters, bool)
    // Routes returns a routes interface along with whether the interface is supported.
    Routes() (Routes, bool)
    // ProviderName returns the cloud provider ID.
    ProviderName() string
    // HasClusterID returns true if a ClusterID is required and set
    HasClusterID() bool
}
```


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Get Involved

Get involved

- Join Cloud Provider SIG
  - Join regular SIG meeting: Wednesdays at 9:00 PT (Pacific Time) (biweekly)
  - Join the cloud specific meeting (usually monthly)
- Get involved with active development (filter by label:sig/cloud-provider)
  - Discuss the active KEPs
  - Review or open the cloud provider PRs
  - Triage or report cloud provider issues
  - Enhance the end-to-end testing and make it less cloud specific
  - Enhancement the documents
- Refer Kubernetes community document for more details


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Thank you!