IBM

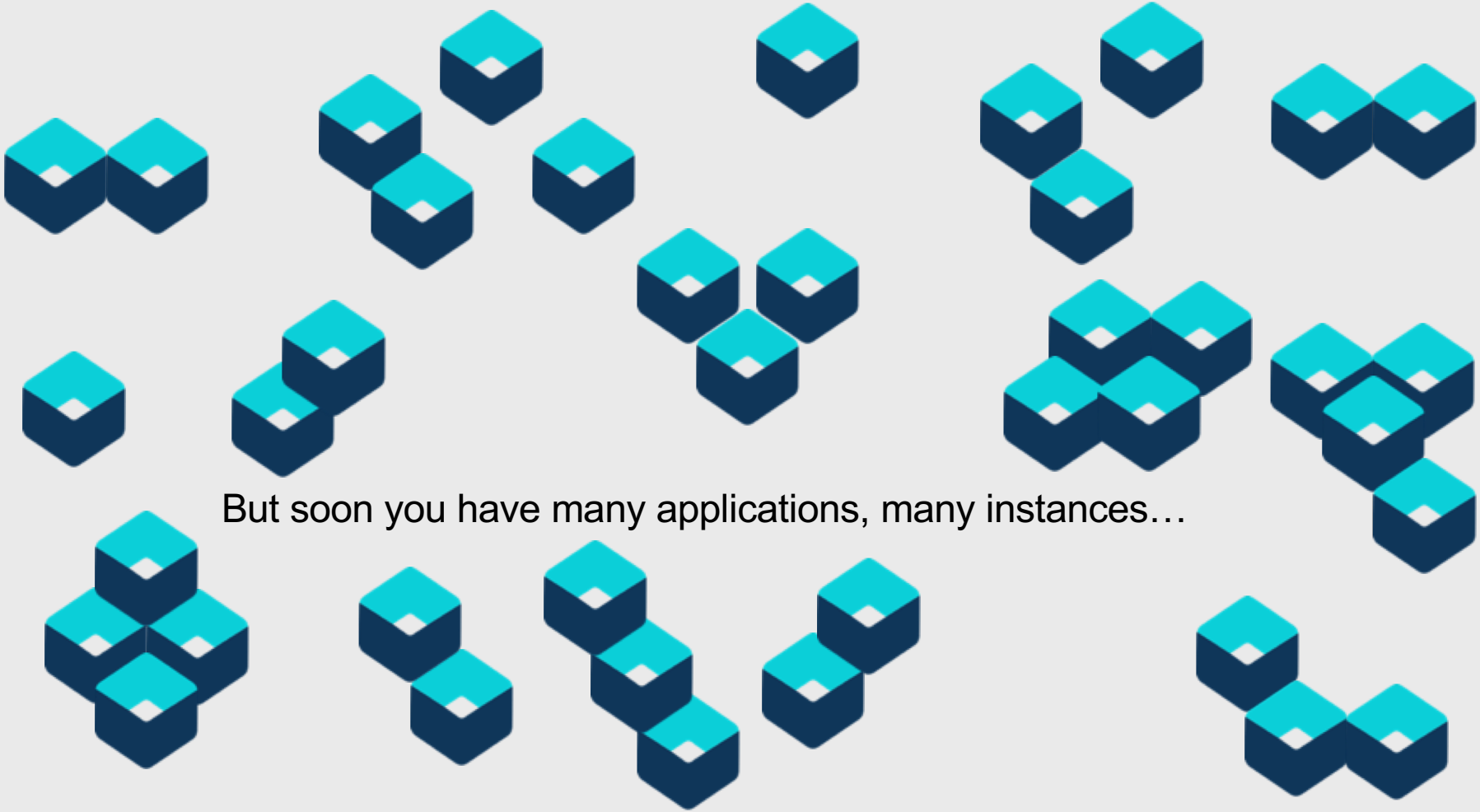
Kubernetes

Basics

IBM **Cloud**

Everyone’s container journey starts with one container….

At first the growth is easy to handle….



But soon you have many applications, many instances…

And that is why there is container orchestration

What is container orchestration?

Management of the deployment, placement, and lifecycle of workload containers

Cluster management

- Federates multiple hosts into one target

Scheduling

- Distributes containers across nodes

Service discovery

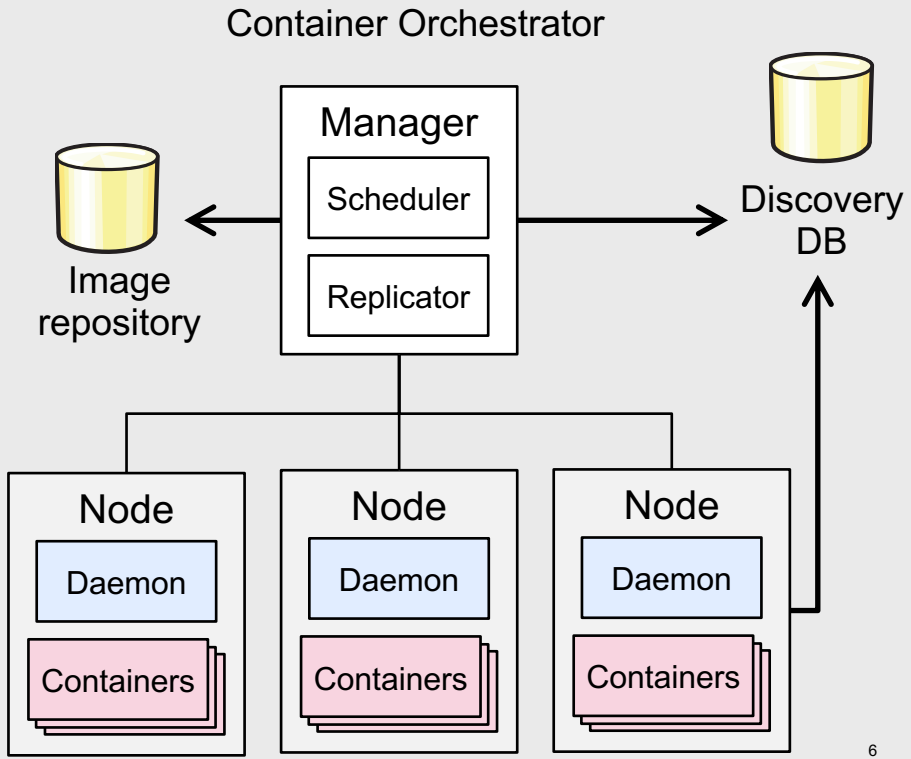
- Knows where the containers are located
- Distributes client requests across the containers

Replication

- Ensures the right number of nodes and containers

Health management

- Replaces unhealthy containers and nodes



Container orchestration responsibilities

Container orchestration

- Scheduling
- Cluster management
- Service discovery

Related functionality

- Provisioning
- Monitoring
- Configuration management

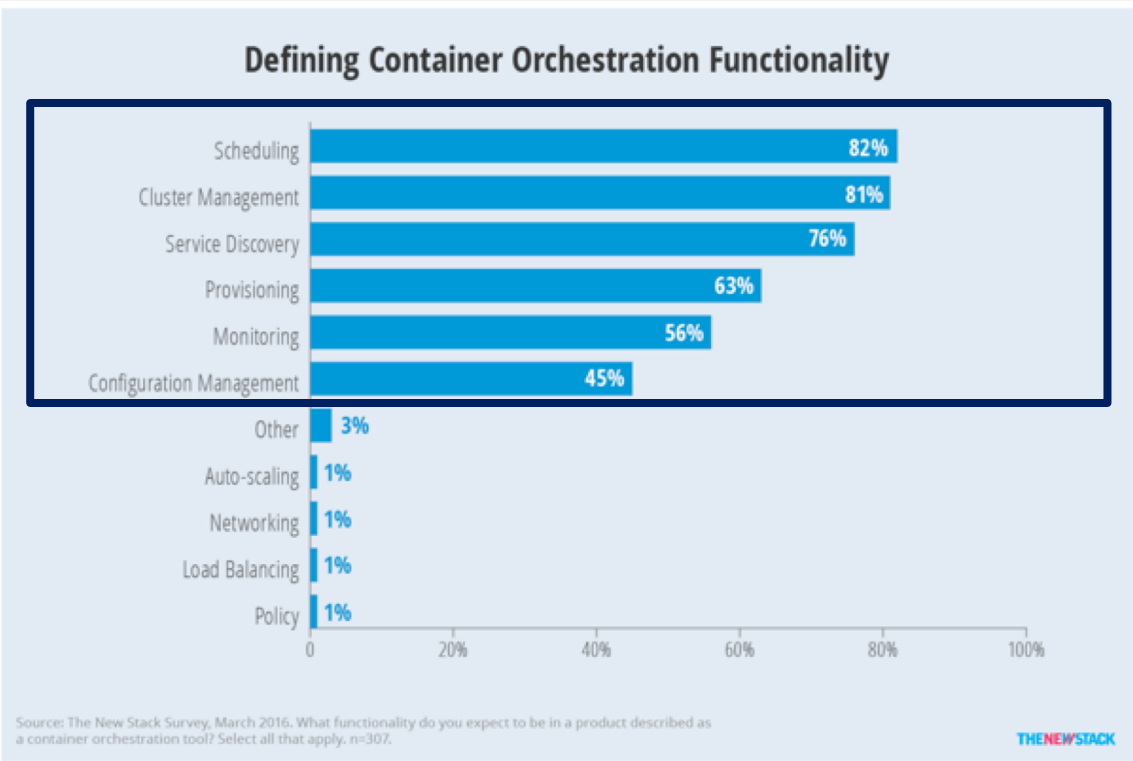
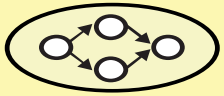


Figure 3: Only 45 percent of respondents consider configuration management to be part of a container orchestration product.

Container ecosystem

7		Application workflow	
6		Container orchestration	Kubernetes, Docker Swarm, Apache Mesos
5		Container scheduling	
4		Container engine	Docker
3		Operating system	Ubuntu, RedHat, CoreOS
2		Virtual infrastructure	VMWare, AWS
1		Physical infrastructure	

© Copyright IBM Corporation 2017

What is Kubernetes?

Container orchestrator

- Runs and manages containers
- Unified API for deploying web applications, batch jobs, and databases
- Maintains and tracks the global view of the cluster
- Supports multiple cloud and bare-metal environments

Manage applications, not machines

- Rolling updates, canary deploys, and blue-green deployments

Designed for extensibility

- Rich ecosystem of plug-ins for scheduling, storage, and networking

Open source project managed by the Linux Foundation

- Inspired and informed by Google's experiences and internal systems
- 100% open source, written in Go

Kubernetes strengths

Clear governance model

- Managed by the Linux Foundation.
- Google is driving the product features and roadmap, while allowing the rest of the ecosystem to participate.

Growing and vibrant ecosystem

- IBM, Huawei, Intel, and Red Hat are among the companies making prominent contributions to the project.

Avoid dependency and vendor lock-in

- Active community participation and ecosystem support.

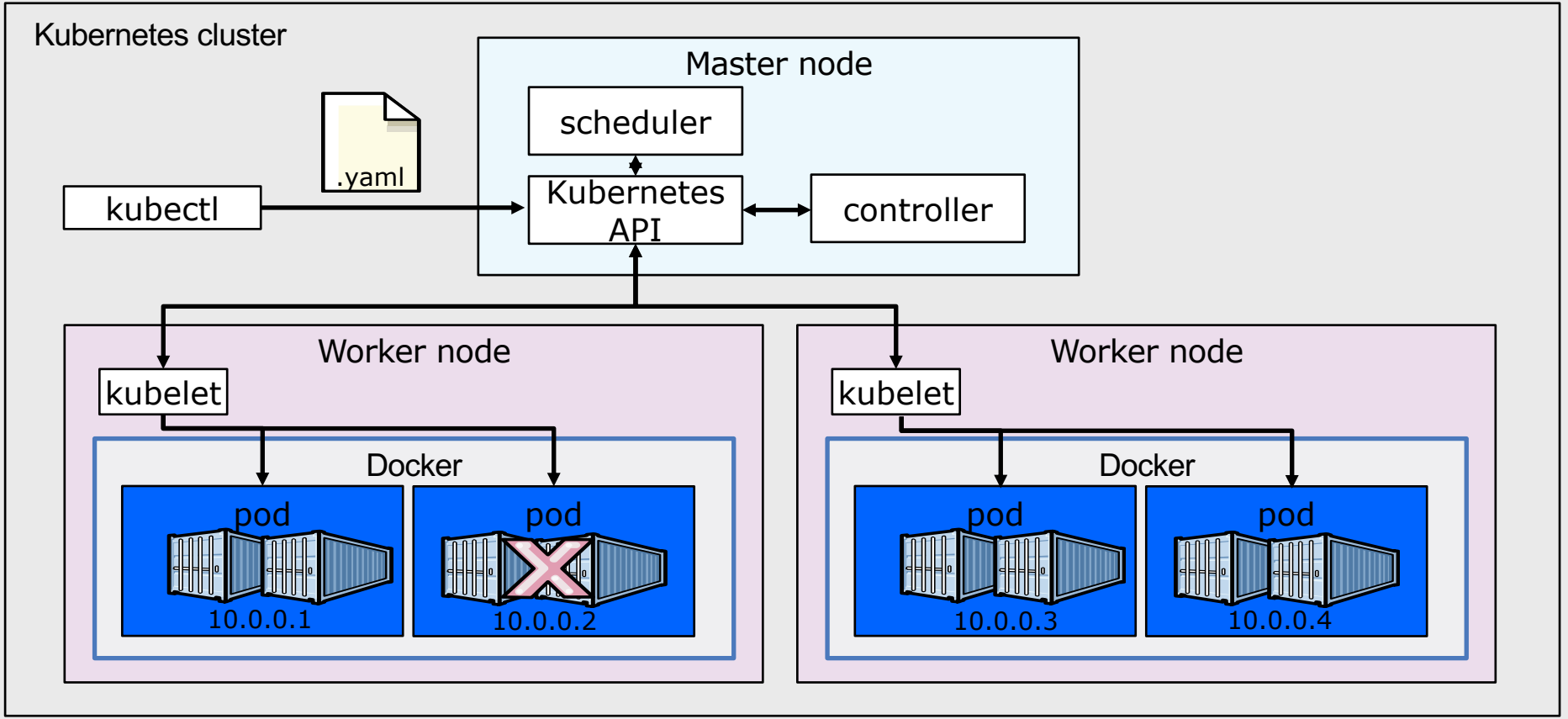
Support for a wide range of deployment options

- Customers can choose between bare metal, virtualization, private, public, and hybrid cloud deployments
- Wide range of delivery models across on-premises and cloud-based services.

Design is more operations-centric

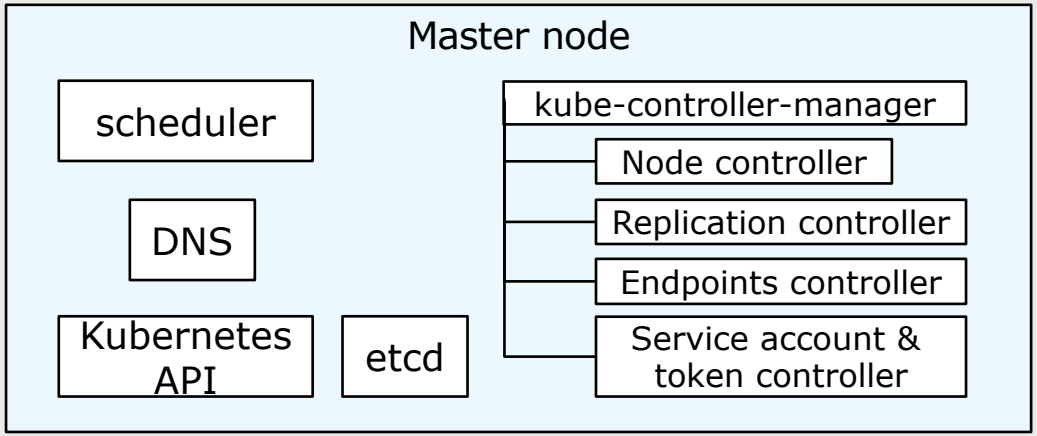
- First choice of DevOps teams.

Kubernetes cluster architecture



Master node components

- Make scheduling decisions for the cluster, and respond to cluster events, like a node failure
- Can run on any node in the cluster, but typically all master components run on the same virtual machine (vm), and do not run any container apps on that vm



Kubernetes terminology: Master node components

Etcd

- A highly-available key value store
- Stores all cluster data

API Server

- Exposes API for managing Kubernetes
- Used by kubectrl CLI

Scheduler

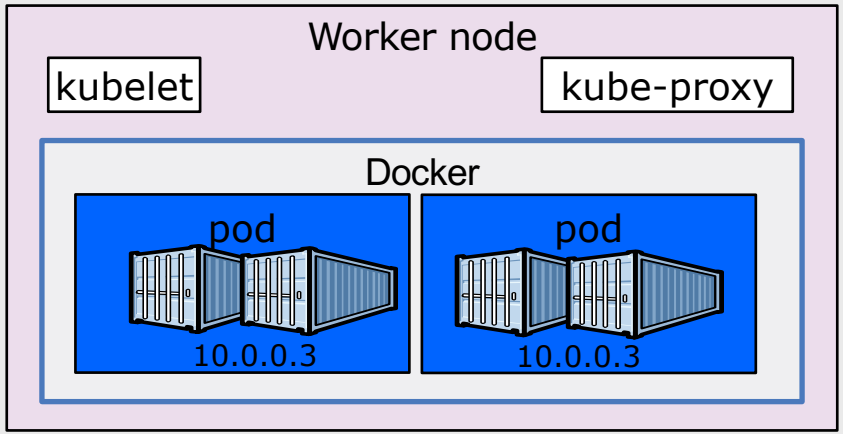
- Selects the worker node for each pods runs

Controller manager

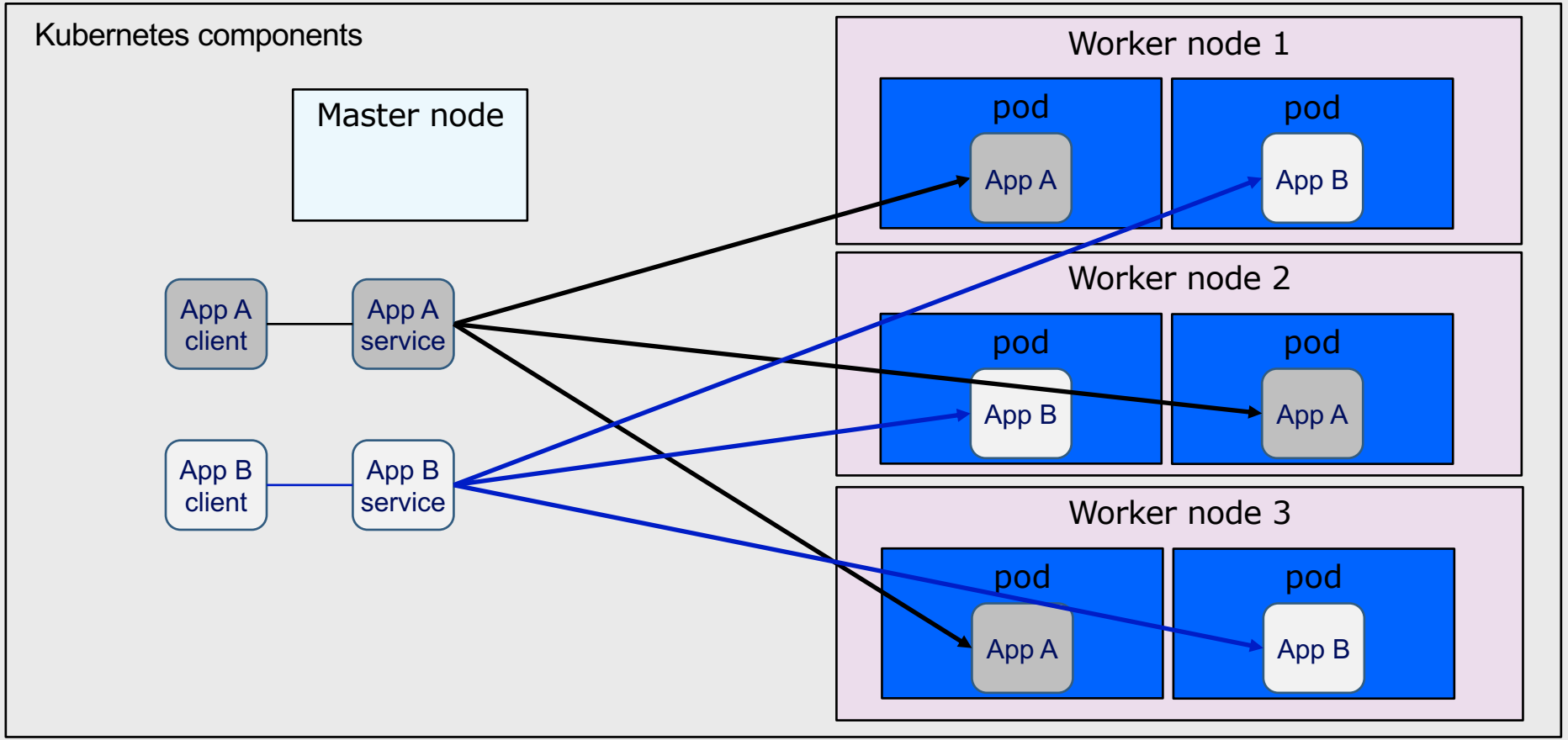
- Daemon that runs controllers (background threads that handle routine tasks in the cluster)
- Node Controller – Responsible for noticing and responding when nodes go down
- Endpoints Controller – Populates the Endpoints object (joins services and pods)
- Service Account and Token Controllers – Create default accounts and API access tokens for new namespaces

Worker node: Components

- Provide the Kubernetes runtime environment; run on every node
- Maintain running pods



Kubernetes terminology: Workloads



© 2017 IBM Corporation

Kubernetes terminology: Workloads

Container

- Unit of packaging

Pod

- Smallest deployment unit in Kubernetes
- Collection of containers that run on a worker node
- Has its own IP address
- Shares a PID namespace, network, and hostname

Service

- Collection of pods exposed as an endpoint
- Types:
  - ClusterIP
  - NodePort
  - LoadBalancer
  - ExternalName

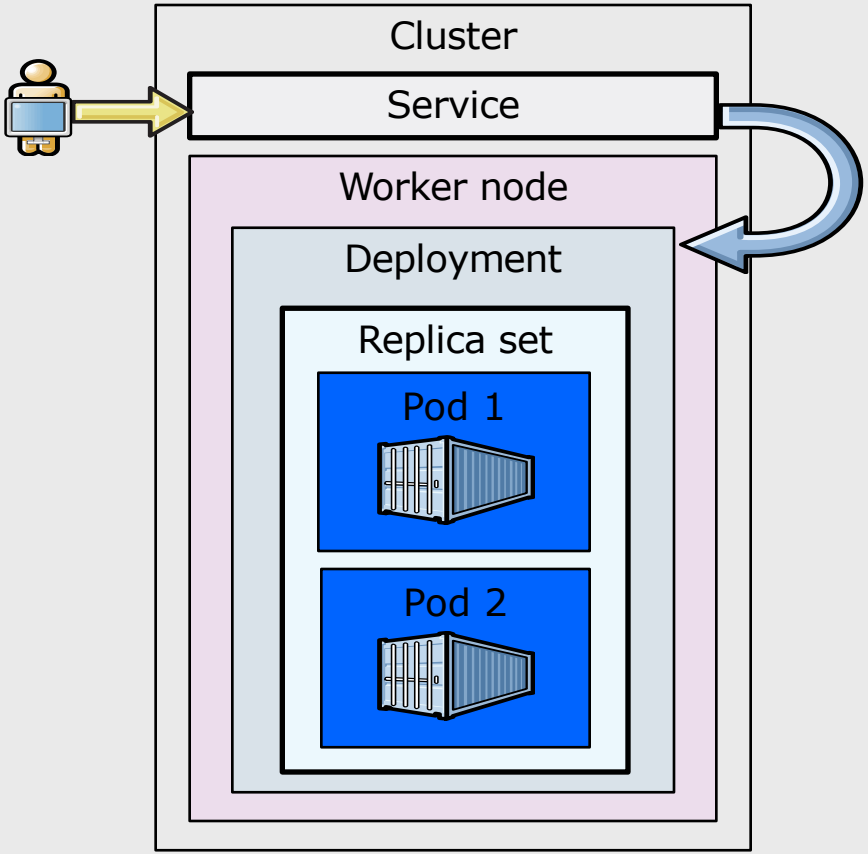
Kubernetes terminology: Deployments and replica sets

Deployments

- Describe the “desired state”
- Provide declarative updates for pods and replica sets

Replica sets

- Use pod templates to define a set of pod replicas
- Ensure that the specified number of pod replicas are running
- Can be used directly, but are typically used with deployments to orchestrate the pod lifecycle



© Copyright IBM Corporation 2018

Kubernetes terminology: Deployments

Set of pods deployed together such as an application.

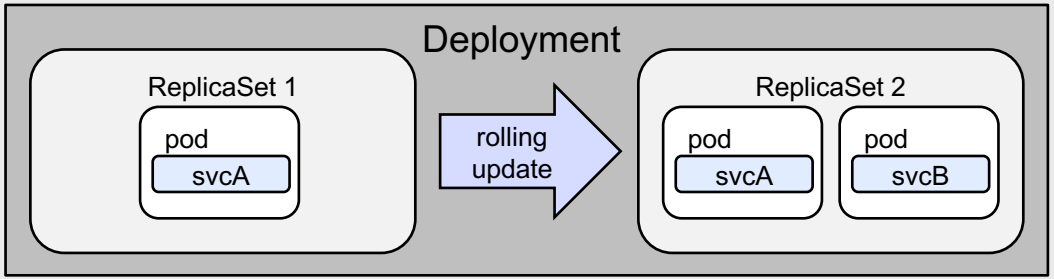
Declarative: Revising a deployment creates a ReplicaSet describing the desired state

Rollout: Deployment controller changes the actual state to the desired state at a controlled rate

Rollback: Each deployment revision can be rolled back

Scale and autoscale: A deployment can be scaled

.



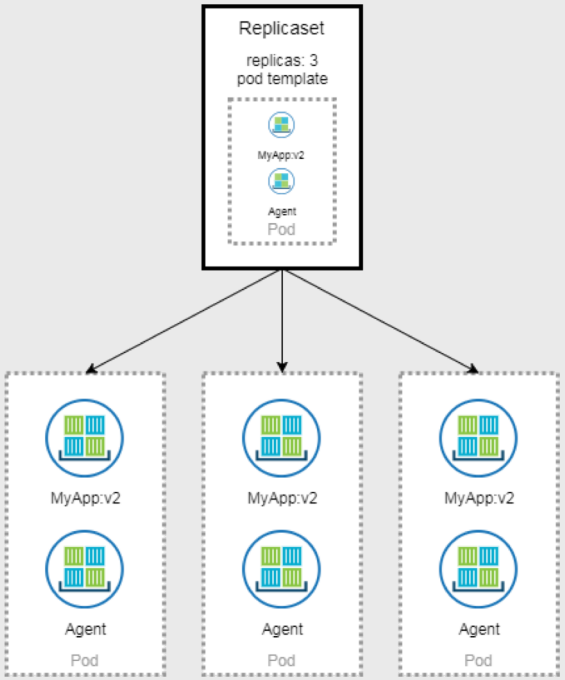
Kubernetes terminology: Replica sets

Scale and provide resiliency.

Replicasets run one-to-many instances of the desired pod.

When possible the replica pod should be stateless or near-stateless.

Ensures that a specific number of pod replicas are running.

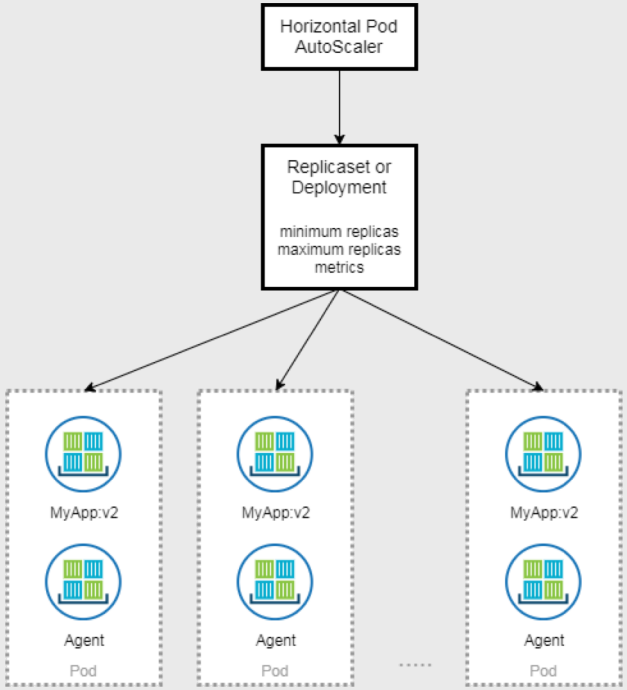



```yaml
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # this replicas value is default
  # modify it according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        ports:
        - containerPort: 80
```

Kubernetes terminology: Autoscaling

Horizontal Pod Auto-scaling (HPA)

Able to scale the number of running pods in a replicaset based upon resource (or application custom) metrics.




```yaml
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
status:
  observedGeneration: 1
  lastScaleTime: <some-time>
  currentReplicas: 1
  desiredReplicas: 1
  currentMetrics:
  - type: Resource
    resource:
      name: cpu
      currentAverageUtilization: 0
      currentAverageValue: 0
```

Kubernetes terminology: Naming

Name

- Each resource object by type has a unique name

Namespace

- Resource isolation: Each namespace is a virtual cluster within the physical cluster
  - Resource objects are scoped within namespaces
  - Low-level resources are not in namespaces: nodes, persistent volumes, and namespaces themselves
  - Names of resources need to be unique within a namespace, but not across namespaces
- Resource quotas: Namespaces can divide cluster resources
- Initial namespaces
  - `default` – The default namespace for objects with no other namespace
  - `kube-system` – The namespace for objects created by the Kubernetes system

Kubernetes configuring Containers and Resources

Label

- Metadata assigned to Kubernetes resources (pods, services, etc.)
- Key-value pairs for identification
- Critical to Kubernetes

Selector

- An expression that matches labels to identify related resources

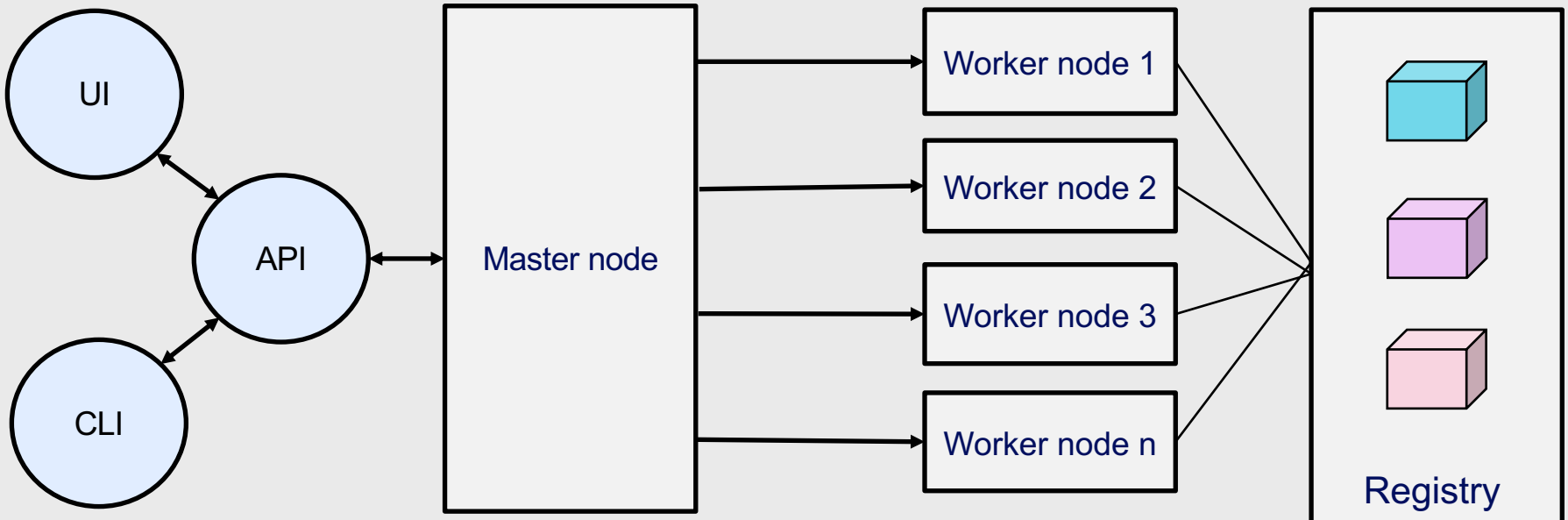
ConfigMap

- Configuration values used by containers in a pod
- Configuration is stored outside of the container image

Secrets

- Sensitive information that containers read or consume
- Encrypted in special volumes mounted automatically

Kubernetes management architecture



- Etcd
- API Server
- Controller Manager Server
- Scheduler

Kubectl commands

Support different approaches to working with Kubernetes objects:

- Imperative commands on live objects.
- Individual configuration files or directories of files.

Important: maintain a consistent approach when working with the same object; do not mix approaches.

Basic syntax:

```
<verb> <objecttype> [<subtype>] <instancename>
```

- Where the `<verb>` is an action such as: **`create`**, **`run`**, **`expose`**, **`autoscale`**.
- `<objecttype>` is the object type, such as a **`service`**.
- Some objects have subtypes. For example, a service has **`ClusterIP`**, **`LoadBalancer`**, **`NodePort`**.
- Use the **-h** flag to find the arguments and flags supported by a subtype
- `<instancename>` specifies the name of the object

© Copyright IBM Corporation 2018

Kubectl command useful examples

Get the state of a cluster

```
$ kubectl cluster-info
```

Get all the nodes of a cluster

```
$ kubectl get nodes -o wide
```

Get info about the pods of a cluster

```
$ kubectl get pods -o wide
```

Get info about the replication controllers of a cluster

```
$ kubectl get rc -o wide
```

Get info about the services of a cluster

```
$ kubectl get services
```

Get full config info about a Service

```
$ kubectl get service NAME_OF_SERVICE -o json
```

Get the IP of a Pod

```
$ kubectl get pod NAME_OF_POD -template={{.status.podIP}}
```

Delete a Pod

```
$ kubectl delete pod NAME
```

Delete a Service

```
$ kubectl delete service NAME_OF_SERVICE
```

Resources

Kubernetes tutorial

- https://kubernetes.io/docs/tutorials/kubernetes-basics/

Introduction to container orchestration

- https://www.exoscale.ch/syslog/2016/07/26/container-orch/

TNS Research: The Present State of Container Orchestration

- https://thenewstack.io/tns-research-present-state-container-orchestration/

Large-scale cluster management at Google with Borg

- https://research.google.com/pubs/pub43438.html