SIG Cluster Lifecycle: Deep Dive

Di Xu & Alexander Kanevskiy
2018-11-15
Who Are We?

Di Xu
Kubernetes Member
Top 50 Code Contributor to K/K
Ant Financial
@dixudx

Alexander Kanevskiy
Kubernetes Member
Open Source Technology Center
Intel
@kad
Our Mission

SIG Cluster Lifecycle’s objective is to simplify creation, configuration, upgrade, downgrade, and teardown of Kubernetes clusters and their components.
SIG Cluster Lifecycle Projects

- kubeadm
- cluster-api
- kops
- kubespray
- minikube
- bootkube
- kubeadm-dind-cluster
- ...

[Image of SIG Cluster Lifecycle Projects]
Agenda

- kubeadm
  - Best practices: install and fine-tuning
  - How it works
  - Current status and plans
- Cluster API
- Join and contribute!
kubeadm
Best Practices
Installing kubeadm

Ways of installing kubeadm:

- Supported distributions
  - Container Linux
  - DEBs and RPMs: Ubuntu, Debian, Hypriot, RHEL, CentOS, Fedora
- There are other distros
  - OpenSuSE, ArchLinux, ...

What is actually needed on the node:

- kubeadm
- kubelet
- kubectl
- CNI plugins
-cri-tools

For distros without official support you can use Container Linux section for manually install binaries
https://kubernetes.io/docs/setup/independent/install-kubeadm/
kubeadm: fine-tuning kubelet

- systemd
  - `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf`
  - Don’t edit it!
- Override a instance-specific kubelet flag via environment file
  - `/etc/default/kubelet` or `/etc/sysconfig/kubelet`
  - `KUBELET_EXTRA_ARGS=--flag=...`
- ConfigMap in “kube-system” namespace
  - “kubelet-config-1.X”
- kubeadm-managed files
  - `/var/lib/kubelet/kubeadm-flags.env`
  - `/var/lib/kubelet/config.yaml`
kubeadm: Offline installation

- Offline and air-gapped detection improved in v1.12
- “Calling home”
  - --kubernetes-version
    - stable, stable-1, stable-1.12, ...
    - latest, latest-1, latest-1.12, ...
    - ci/latest-1.13
  - upgrade plan

```bash
$ kubeadm config images list
```
- k8s.gcr.io/kube-apiserver
- k8s.gcr.io/kube-controller-manager
- k8s.gcr.io/kube-scheduler
- k8s.gcr.io/kube-proxy
- k8s.gcr.io/pause
- k8s.gcr.io/etcd
- k8s.gcr.io/coredns

```bash
$ kubeadm config images pull
```

https://dl.k8s.io/release/${RELEASE}/bin/linux/amd64

```bash
$ gsutil ls -l gs://kubernetes-release/release/v1.12.2/bin/linux/amd64/
```
kubeadm: Network and Proxies

- Proxy for CRI
  - HTTP_PROXY
  - HTTPS_PROXY
  - NO_PROXY
    - Attention: local registries

- Proxy for kubeadm
  - HTTP_PROXY
  - HTTPS_PROXY
  - NO_PROXY
    - Node IPs range
    - Service IPs range
    - POD IPs range
    - Cluster domains

NO_PROXY=tld.com, 192.168.0.0/16, 10.0.0.0/8, cluster.local
kubeadm: Using own certificates

- Directory with all your custom certificates (--cert-dir)
  - etcd serving: `etcd/ca.key` and `etcd/ca.crt`
  - Cluster CA: `ca.key` and `ca.crt`
  - API Server serving: `apiserver.key` and `apiserver.crt`
  - ServiceAccount signing key: `sa.pub` and `sa.key`
  - API Server Kubelet client: `apiserver-kubelet-client.key` and `apiserver-kubelet-client.crt`
  - Frontend Proxy -- for API Aggregation
    - CA: `front-proxy-ca.key` and `front-proxy-ca.crt`
    - Client: `front-proxy-client.key` and `front-proxy-client.crt`
kubeadm: Using ComponentConfig

● Why use a config file for configuring kubeadm?
  ○ Tired of long complicated flags
  ○ Wants more advanced features, including more customized parameters
  ○ The API spec in v1.12 is unstable, but there’s always an upgrade path
  ○ The API spec is graduated to v1beta1 in v1.13

● Supported Kinds in config file
  ○ InitConfiguration: kubeadm init master-local runtime config
  ○ ClusterConfiguration: cluster-wide settings, wide range of parameters
  ○ KubeletConfiguration: cluster-wide kubelet options
  ○ JoinConfiguration: kubeadm join node-local runtime config

● Saved to ConfigMaps for future upgrading
  ○ “kubeadm-config” and “kubelet-config-1.x” in “kube-system”
kubeadm: InitConfiguration

Usage
- “kubeadm init --config ...”

Why
- Custom API endpoint address
- Specify init bootstrap tokens
- Pass custom kubelet flags
- Set node name/taints

```
apiVersion: kubeadm.k8s.io/v1beta1
group: InitConfiguration
localAPIEndpoint:
  address: "10.100.0.1"
  port: 6443
nodeRegistration:
  criSocket: "/var/run/crio/crio.sock"
kubeletExtraArgs:
  cgroupDriver: "cgroupfs"
bootstrapTokens:
  ...
**kubeadm: ClusterConfiguration**

- **Usage**
  - "kubeadm init --config ..."

- **Why**
  - Fine tune cluster defaults
  - Custom args and volume mounts to control plane components

```yaml
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: "v1.12.2"
imageRepository: registry.example.com
networking:
  serviceSubnet: "10.96.0.0/12"
  dnsDomain: "cluster.local"
etcd:
  ...
apiServer:
  extraArgs:
    ...
  extraVolumes:
    ...
```
Kubeadm: KubeletConfiguration

Usage
- "kubeadm init --config ...

Why
- Fine tuning kubelet parameters

```yaml
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
cpuManagerPolicy: static
failSwapOn: false
maxPods: 110
resolvConf: /etc/my-cluster-resolv.conf
```
Kubeadm: JoinConfiguration

```yaml
apiVersion: kubeadm.k8s.io/v1beta1
kind: JoinConfiguration
nodeRegistration:
  criSocket: /var/run/crio/crio.sock
  name: k8s-node2.example.com
kubeletExtraArgs:
  cgroupDriver: "cgroupfs"

discovery:
  bootstrapToken:
    apiServerEndpoint: kube-apiserver:6443
    token: abcdef.0123456789abcdef
```

- **Usage**
  - "kubeadm join --config …"

- **Why**
  - Customize discovery options
  - Pass custom kubelet flags
  - Set node name/taints

```
kubeadm config print-default --api-objects JoinConfiguration
```
Kubeadm: KubeProxyConfiguration

Usage
- “kubeadm init --config …”

Why
- Fine tuning kubeproxy parameters

```yaml
apiVersion: kubeadm.k8s.io/v1beta1
kind: KubeProxyConfiguration
iptables:
  masqueradeAll: false
  masqueradeBit: 14
  syncPeriod: 30s
ipvs:
  syncPeriod: 30s
```
kubeadm: Managing “addons”

• Installed and configured by kubeadm
  • DNS
    • CoreDNS has already been marked as default starting from 1.12
    • kube-dns can be used via feature-gate
  • kube-proxy - deployed as a DaemonSet
• Anything that isn’t required for meeting the Conformance criteria is outside of kubeadm scope
  • Dashboard
  • CNI plugins
• Future: Cluster Bundles…
kubeadm
Under the hood
# kubeadm: Atomic work “phases”

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preflight</td>
<td>Run pre-flight checks</td>
</tr>
<tr>
<td>kubelet</td>
<td>Commands related to handling the kubelet.</td>
</tr>
<tr>
<td>certs</td>
<td>Generates certificates for a Kubernetes cluster</td>
</tr>
<tr>
<td>kubeconfig</td>
<td>Generates all kubeconfig files for the control plane and the admin kubeconfig file</td>
</tr>
<tr>
<td>controlplane</td>
<td>Generates all static Pod manifest files necessary to establish the control plane</td>
</tr>
<tr>
<td>kubeconfig</td>
<td>Generates all kubeconfig files for the control plane and the admin kubeconfig file</td>
</tr>
<tr>
<td>upload-config</td>
<td>Uploads the currently used configuration for kubeadm to a ConfigMap</td>
</tr>
<tr>
<td>mark-master</td>
<td>Mark a node as master</td>
</tr>
<tr>
<td>bootstrap-token</td>
<td>Manage kubeadm-specific bootstrap token functions</td>
</tr>
<tr>
<td>addon</td>
<td>Installs required addons for passing Conformance tests</td>
</tr>
</tbody>
</table>
kubeadm init

- Run Preflight Checks
- Configure Kubelet
- Generate Certificates
- Generate KubeConfig Files
- Generate static Pod Manifests for the Control Plane
- Wait for the Control Plane to be healthy
- Install DNS and Proxy Addons
- Setup the RBAC Authorization System
- Generate a (by default random) Bootstrap Token
- Taint and label the master
- Upload kubeadm & kubelet config to a ConfigMap
- Configure Kubelet
- Generate Certificates
- Generate KubeConfig Files
- Generate static Pod Manifests for the Control Plane
- Wait for the Control Plane to be healthy
- Install DNS and Proxy Addons
- Setup the RBAC Authorization System
- Generate a (by default random) Bootstrap Token
- Taint and label the master
- Upload kubeadm & kubelet config to a ConfigMap
kubeadm join --experimental-control-plane

1. Preflight Checks
   - Checks if the Cluster Configuration Supports

2. Fetches Init Configuration
   - Run Specific Checks based on the Init Configuration

3. Runs Kubeadm Init Preflight Check

4. Generates static Pod Manifests for new Control Plane

5. Bootstraps Kubelet

6. Uploading Currently Used InitConfiguration
   - Applies Master Label and Taints
   - --experimental-control-plane
kubeadm upgrade: Control Plane

Preflight Checks

Checks if the cluster is healthy

Gets the configuration from the "kubeadm-config" ConfigMap

Enforces all version skew policies

Upgrades the control plane Static Pods

Upgrades RBAC rules and addons
kubeadm upgrade: Node

- Currently kubeadm only supports upgrading the kubelet *configuration*, **NOT** the kubelet itself.
  - Manually upgrade the kubelet package using your package manager (e.g. apt, yum)

- Kubeadm will download ConfigMap “`kubelet-config-1.x`” in the “`kube-system`” namespace for global cluster kubelet settings
  - Will be stored as “`/var/lib/kubelet/config.yaml`”
  - Require kubelet version v1.11+
kubeadm
Status and Roadmap
kubeadm 1.12 Changelog

- Config / ComponentConfig
- Experimental: join master
- Improved CRI support
- Improved offline / air-gapped support
- Deprecating of SelfHosting, HA
kubeadm 1.13 Roadmap

- kubeadm is graduated to GA/stable!!!
- Promote the config file from v1alpha3 to v1beta1
- Phases graduated to beta
  - addon, bootstrap-token, certs, control-plane, etcd, kubeconfig, kubelet-start, mark-control-plane, preflight, upload-config
- Bug fixes
Cluster API
What is the Cluster API?

- **Declarative API**
  - Cluster
  - Machine
  - Machine Set + Machine Deployment

- **Common Logic**
  - Machine Lifecycle (including Provisioning)
  - Machine Upgrade

- **Pluggable Architecture**
  - Infrastructure platform (vSphere, GCP, AWS, etc.)
  - Support for various Operating Systems

- **Tooling, Services**
  - Cluster Bootstrapping, Upgrade
  - Auto-scaling, Repair, Node Auto-provisioning
Bootstrapping - 10 km view

Cluster
- Machine Controller
- Cluster Controller

Control Plane
- Machine A
- Cluster A

CLI
- User

Local Temporary
External Cluster
- Machine Controller
- Machine A
- Cluster Controller
- Cluster A

Control Plane
Cluster API: Status

• Actively developed providers
  • AWS: https://github.com/kubernetes-sigs/cluster-api-provider-aws
    • v1.0.0-alpha.3
  • DigitalOcean: https://github.com/kubernetes-sigs/cluster-api-provider-digitalocean
    • v0.2.0
  • AWS/Openshift: https://github.com/openshift/cluster-operator
  • Azure: https://github.com/platform9/azure-provider
  • GCE, https://github.com/kubernetes-sigs/cluster-api-provider-gcp
  • OpenStack: https://github.com/kubernetes-sigs/cluster-api-provider-openstack
  • vSphere: https://github.com/kubernetes-sigs/cluster-api-provider-vsphere
SIG Cluster Lifecycle
Join us and get involved!
How can you contribute to our SIG

- Contributing to SIG Cluster Lifecycle documentation
- We’re working on growing the contributor/reviewers pool; scaling the SIG
- We have “Office Hours” for our projects: weekly for kubeadm, bi-weekly for kops and kubespray...
- Cluster API office hours weekly for both US West Coast and EMEA
- Full list of SIG meetings and links to minutes and recordings can be found on SIG page
- Attend our Zoom meetings / be around on Slack
- Look for “good first issue”, “help wanted” and “sig/cluster-lifecycle” labeled issues in our repositories
What now?

- Follow the [SIG Cluster Lifecycle YouTube playlist](#)
- Check out the [meeting notes](#) for our bi-weekly SIG meetings
- Join [#sig-cluster-lifecycle, #kubeadm, #cluster-api, #kops-dev, #kops-users, #kubespray, #minikube, ...](#)
- Prep for and take the [Certified Kubernetes Administrator](#) exam
- Check out the [kubeadm setup guide, reference doc and design doc](#)
- Read how you can [get involved](#) and improve kubeadm!
Thank You!