Phases in kubeadm - implementation

This document will be maintained during the implementation of different phases, documenting common implementation guidelines and principles.

Atomicity

- The final goal to this activity is to break down kubeadm into atomic phases, thus providing a reusable and composable API.

Atomicity vs Usability/Simplicity

- While phases implementation should seek for atomicity, phase UX should also support usability/simplicity.
  This will be address by providing UX command for:
  - atomic phases, e.g. **kubeadm phase kubeconfig etcd** (the highest level of atomicity, the lowest of usability)
  - “bulk phases”, e.g. **kubeadm phase kubeconfig all**, (thus supporting an intermediate level of atomicity and usability in between kubeadm init and atomic phases)

Consistency:

- (Behavioral Consistency) Each phase, when invoked, by default should provide the very same behaviour of the same phase executed in the context of kubeadm init
- (UX Consistency) Each phase UX, should be consistent with kubeadm init UX, and thus providing the very same flags, being sensitive to the same env variables, being feed by the same config file etc. etc.

Flexibility

- When significative, additional flags (a.k.a flags not supported by kubeadm init) will be added to phases, providing support for additional behaviours.
  e.g. --stdout flag for writing output io.stdout instead of /etc/kubernetes .
- Whenever an additional flag would apply to more than one phases, naming and behaviour should be consistent;