# kubify Documentation

***Release rtfd***

**Oct 29, 2018**

# Contents

**1 Kubify Troubleshooting Guide** **3**
- 1.1 SSH into Machines . . . 3
- 1.2 Fix Control plane by using the Bootstrap Controlplane . . . 3
- 1.3 Defrag ETCD and free up Disk Space . . . 4
- 1.4 Restore Cluster from Backup . . . 4

# Kubify

Kubify is a [Teraform](https://www.terraform.io/) based provisioning project for setting up production ready [Kubernetes](https://kubernetes.io/) clusters on public and private Cloud infrastructures. Kubify currently supports:

- OpenStack
- AWS
- Azure

Key features of Kubify are:

- Multi master node setup
- Etcd backup and restore
- Supports rolling updates

CHAPTER 1

# Kubify Troubleshooting Guide

Before you start make sure that your Terraform version is *v0.11.3*.

## 1.1 SSH into Machines

If *k8s/bin/{master X/worker X}* isn't working, make sure the following:

- Bastion hosts might be offline, reboot/start the machine
- The SSH daemon on the target machine stopped working. If it is a worker machine, just __restart it__. If it is a Master node, first make sure that all 3 *kube-etcd-{000X}* pods are working. If that is the case, restart the affected Master node. If ETCD has not all the members, make sure, that the one missing is on the master you are about to restart.

## 1.2 Fix Control plane by using the Bootstrap Controlplane

In case the *kube-{apiserver,controller-manager,scheduler}* isn't working, you can use the bootstrap control plane on *master-0* to recover the Kubernetes control plane. To do that you need to copy the *bootstrap-sercrets* into the */etc* folder

```
` k8s/bin/master sudo cp -r /opt/bootkube/assets/tls/ /etc/kubernetes/
bootstrap-secrets `
```

And then copy the corresponding boostrap manifest into the Kubernetres manifest folder

```
` # Kube API Server sudo cp /opt/bootkube/assets/bootstrap-manifests/
bootstrap-apiserver.yaml /etc/kubernetes/manifests/ # Kube Controller Manager
sudo cp /opt/bootkube/assets/bootstrap-manifests/bootstrap-controller-manager.
yaml /etc/kubernetes/manifests/ # Kube Scheduler sudo cp /opt/bootkube/assets/
bootstrap-manifests/bootstrap-scheduler.yaml /etc/kubernetes/manifests/ `
```

Once the Bootstrap components have reconciled the broken Kubernetes control plane, their corresponding manifests can be safely removed again.

` sudo rm /etc/kubernetes/manifests/bootstrap-apiserver.yaml sudo rm /etc/kubernetes/manifests/bootstrap-controller-manager.yaml sudo rm /etc/kubernetes/manifests/bootstrap-scheduler.yaml ` The Boostrap secrets are also not needed anymore and can be removed. ` sudo rm -rf /etc/kubernetes/bootstrap-secrets `

## 1.3 Defrag ETCD and free up Disk Space

Startup ETCD client pod

Spin up an Pod which has the ETCD certificates mounted ` k8s/bin/ks apply -f https://raw.githubusercontent.com/gardener/kubify/master/docs/manifests/etcdctl.yaml k8s/bin/ks exec -it etcdctl sh ` Show current memory consumption

Run a *k8s/bin/ks get pods | grep kube-etcd* to figure out which active ETCD pods are currently running and adapt the *kube-etcdXXXX* number in the command below. The etcd-operator might have reacted pods with different numbers (due to an outage/recreation) ` export ETCDCTL_API=3 && /usr/local/bin/etcdctl --cert /tls/etcd-client.crt --key /tls/etcd-client.key --cacert /tls/etcd-client-ca.crt --endpoints https://kube-etcd-0000.kube-etcd.kube-system.svc:2379,https://kube-etcd-0001.kube-etcd.kube-system.svc:2379,https://kube-etcd-0002.kube-etcd.kube-system.svc:2379 endpoint status -w table ` This should output something like that

| ENDPOINT | ID | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
|---|---|---|---|---|---|---|
| https://kube-etcd-0000.kube-etcd.kube-system.svc:2379<br>https://kube-etcd-0001.kube-etcd.kube-system.svc:2379<br>https://kube-etcd-0002.kube-etcd.kube-system.svc:2379 | 4fa3b433fd377be7<br>97432066a63c6f3a<br>f419c478000bdea4 | 3.1.8<br>3.1.8<br>3.1.8 | 201 MB<br>201 MB<br>201 MB | **false**<br>true<br>false | 45<br>45<br>45 | 156474994<br>156474994<br>156474995 |

Defrag ETCD storage

` export ETCDCTL_API=3 && /usr/local/bin/etcdctl --cert /tls/etcd-client.crt --key /tls/etcd-client.key --cacert /tls/etcd-client-ca.crt --endpoints https://kube-etcd-0000.kube-etcd.kube-system.svc:2379,https://kube-etcd-0001.kube-etcd.kube-system.svc:2379,https://kube-etcd-0002.kube-etcd.kube-system.svc:2379 defrag ` Validate the storage consumtion with the *show operation* above once the *defrag* finished.

## 1.4 Restore Cluster from Backup

Download from S3 the latest snapshot of your ETCD backup. The snapshot file typically looks something like this *3.1.8_0000000000676d59_etcd.backup*.

Add those two lines to your *terraform.tfvars* file ` recover_cluster = true etcd_backup_file = PATH_TO_ETCD_SNAPSHOT `

Before you run the restore operation first run a plan operation ` k8s/bin/plan -s ` You should see, that the compute nodes *storege* (which are the Master nodes) are marked with a +/- which means that they will be recreated.

To start the restore run the apply operation ` k8s/bin/apply -y ` If you run into any errors, re-run the apply operation until the master nodes have been replaced. Especially on OpenStack, sometimes the hypervisor placement (anti-affinity) is causing some trouble due to an OpenStack race condition. Here a re-run is the safest bet. In theory, if the first master [0] has been created, commenting out the 2 options from the *terraform.tfvars* file and a re-run of *k8s/bin/ks apply -y* will recreated the missing machines.

__IMPORTANT__: Remove/comment out the *recovery_cluster* and the *etcd_backup_file* flag from the *terraform.tfvars* file after the restore has been completed. Otherwise the cluster will be recovered every time you run terraform again.

Post Restore Check

After the cluster is restored, you need to make sure that the worker nodes have successfully joined the cluster. A restart of the worker nodes might be necessary

` k8s/bin/worker {0,1,2,...} sudo reboot `