
KubeCon / Tungsten Fabric Developers Summit


# Multi-Cloud Network Segmentation with Kubernetes and TF

**Will Stevens & Syed Ahmed**

December 10th, 2018

# Demo init...


Do we get to mess with him now?
No. We wait until the worst possible moment.
Demo
Rosalbob 2018



## About CloudOps

CloudOps is a services organization with a simple mission...

> *Help our customers own their destiny in the cloud.*

In order to realize this mission, we work to:

> *Deliver future-proof cloud solutions that leverage open source, key partners and CloudOps software, optimizing how cloud services are delivered, consumed, and connected.*

# Why Tungsten Fabric

Tungsten Fabric (TF) aligns well with our mission and vision in enabling organizations to consume and manage cloud services.

- **Open source** is a key enabler for organizations to own their destiny. It provides a level of visibility and control not achievable with proprietary software, and provides the flexibility to extend the platform with community collaboration.
- **Orchestrator integrations** with both OpenStack and Kubernetes (as well as others) creates strong ecosystem alignment which caters to both the existing tooling as well as future-proofing for the next wave of SDN deployments.
- **Service chaining** is an area in the SDN landscape which is not well covered by most implementations. However, TF stands out in this area with its elegant service chaining implementation.
- A **clean abstraction from the underlay network** enables TF to operate in a wide variety of contexts.

# Tungsten Fabric + Kubernetes

Leveraging Tungsten Fabric as a CNI for Kubernetes is a powerful combination, enabling network policies and the ability to extend the Kubernetes networking into other regions or underlay networks.

# Let the journey begin...

It turns out that getting Kubernetes and Tungsten Fabric to work together is non-trivial, so we wanted to give a little summary of our findings.

The Deploy Tungsten Fabric on Kubernetes in 1-step command tutorial is very useful, but we still ran into these issues.

- We needed the kernel version `3.10.0-862.14.4` instead of `3.10.0-862.3.2`.
- We needed 32GB of RAM and 50GB of disk on the Kubernetes master to have enough resources.
- The version of the *Deploy in 1-step* guide which is rendered on the website strips out the `sed` variable `{{ K8S_MASTER_IP }}`.

After we are back from KubeCon, I will open a PR to improve the documentation in these areas.

# The journey continues...

We also ran into these challenges when deploying Kubernetes and Tungsten Fabric together:

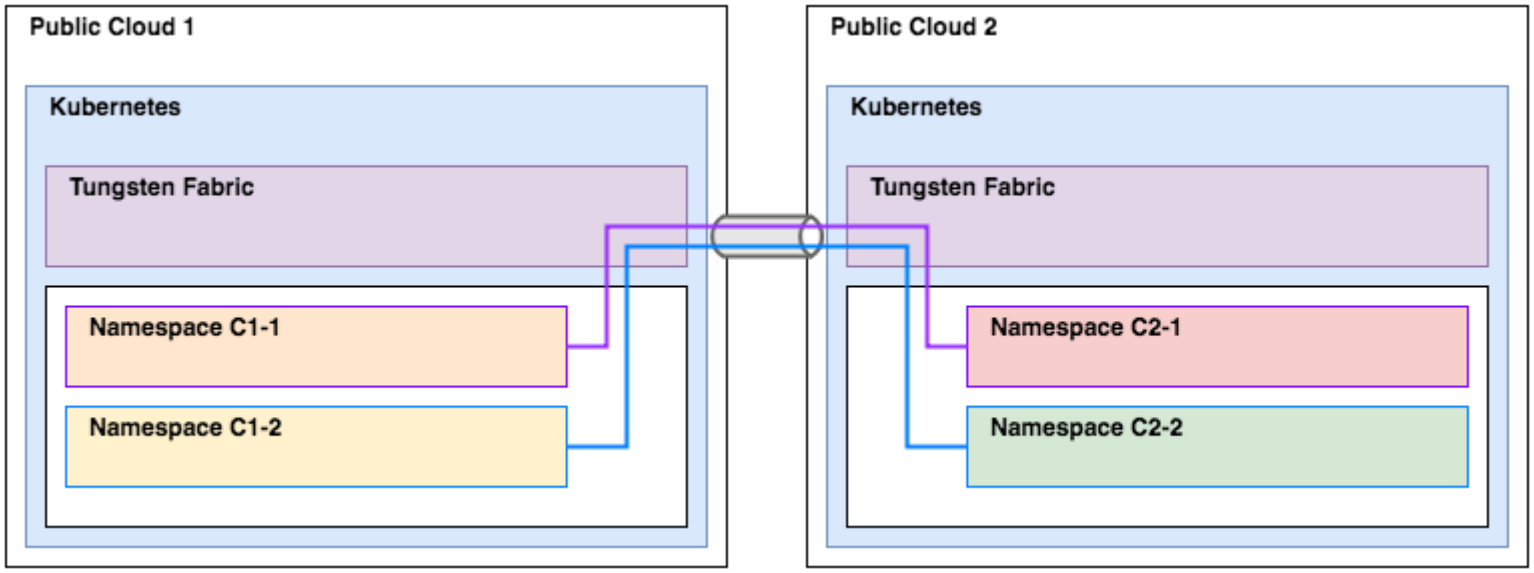
- We needed to change the `VROUTER_GATEWAY` variable in the helm chart to be the gateway of the underlay network and not the master IP (which is what is documented), otherwise when TF came up, we would lose network access to the master node.
- We needed to pin the Kubernetes version to `1.10.11`, as version `1.12` does not seem to be supported yet.
- We needed to pin a specific version of `docker` to use in order for everything to come up correctly.

At this point, we were able to consistently deploy Kubernetes and Tungsten Fabric together with the ability to launch workloads.

# Now the interesting stuff

So now that we can spin up K8s and TF together, we want to deploy two distinct stacks in two different public clouds.

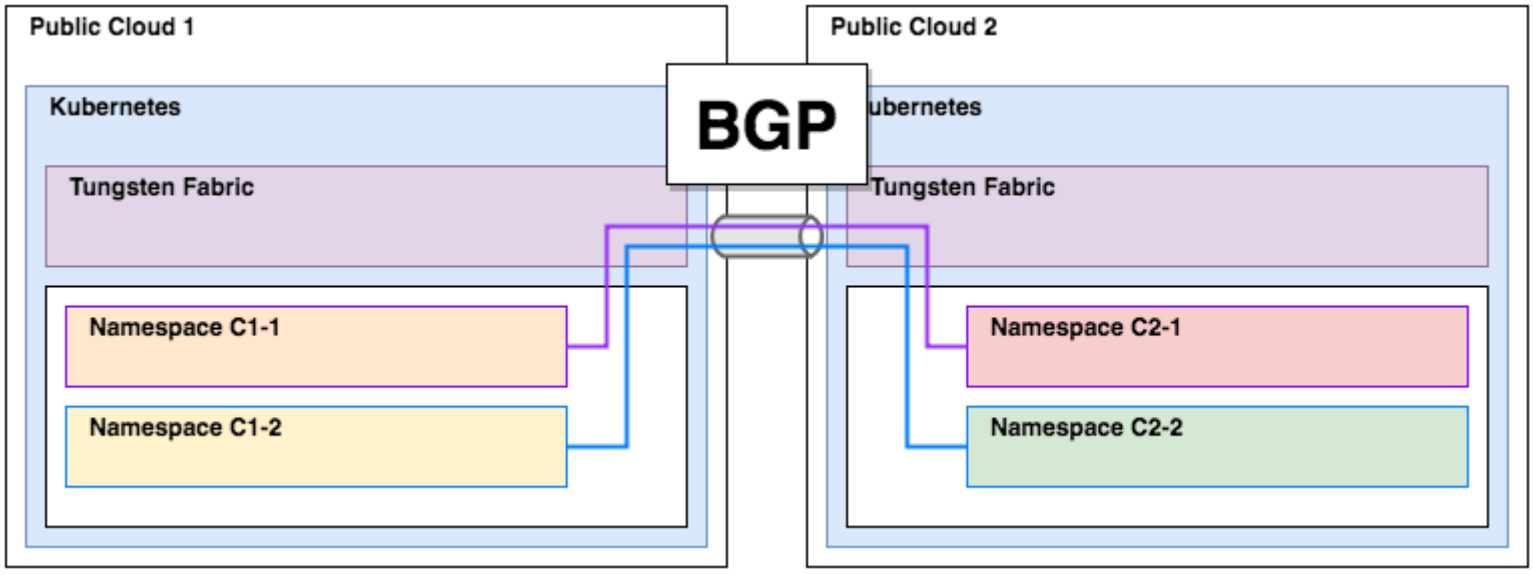
The goal of this is to be able to enable pods in a namespace in one K8s deployment to be able to communicate with a service in a namespace in a K8s deployment in a different cloud.



# Bridging the gap

An interesting thing about Tungsten Fabric is that it is a BGP speaker and is able to communicate with other BGP speakers.

Basically, if you have connectivity between two sites in the underlay network, such as a direct connect or a Site-to-Site VPN connection, Tungsten Fabric can route between them.



# Demo check-in...

# Can I use overlapping IP space?

By default, if we spin up both sites the same way, they will have overlapping IP ranges for the `pod`, `service` and `ip-fabric` IPAM ranges.

In attempting to modify the default ranges, we discovered that passing the `--pod-network-cidr` and `--service-cidr` flags to the `kubeadm init` command has mixed results.

- The `--service-cidr` flag correctly configures Kubernetes, but the configuration is not propagated to Tungsten Fabric.
- The `--pod-network-cidr` flag correctly sets the Kubernetes config, but it seems to have no effect on what is provisioned.

# Tungsten Fabric w/ custom IP ranges?

Lets modify the `kube-manager-config` ConfigMap to pass non-overlapping IP ranges to TF so the configuration defined in Kubernetes and Tungsten Fabric match.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-manager-config
  namespace: kube-system
data:
  KUBERNETES_API_SERVER: {{ K8S_MASTER_IP }}
  KUBERNETES_API_SECURE_PORT: "6443"
  K8S_TOKEN_FILE: "/tmp/serviceaccount/token"
  # --- additional config below --- #
  KUBERNETES_POD_SUBNETS: 10.48.0.0/12
  KUBERNETES_SERVICE_SUBNETS: 10.112.0.0/12
  KUBERNETES_IP_FABRIC_SUBNETS: 10.80.0.0/12
```

# Demo check-in...

# Periodically we have problems

Even though all the containers are up and running, sometimes the network does not come up. I believe this is due to a race condition.

Checking the `contrail-status`:

```
== Contrail control ==
control: initializing (No BGP configuration for self)
nodemgr: active
named: active
dns: active
```

and

```
== Contrail vrouter ==
nodemgr: active
agent: initializing (XMPP:control-node:10.177.192.154 connection down, No Configuratio
```

Delete the following pod so it is recreated:

```
$ kubectl delete pod contrail-controller-control-<hash> -n kube-system
```

# Are the demo gods smiling?

# And now...

# Network Segmentation

Right now, the networks are basically open to each other. We have not specified any policies to lock down communication between the different networks.

Review the table below to understand the different options available. Additional detail can be found in the architecture doc.

| Networking Mode | Network Policy | Effect |
|---|---|---|
| Kubernetes default | Any-to-any, no tenant isolation | Any container can talk to any other container or service |
| Namespace isolation | Kubernetes namespaces map to projects in Tungsten Fabric | Containers within a namespace can communicate with each other |
| Service isolation | Each pod is in its own virtual network and security policy is applied so that only the service IP address is accessible from outside the pod | Communication within a pod is enabled, but only the service IP address is accessible from outside a pod |
| Container isolation | Zero-trust between containers in the same pod. | Only specifically allowed communications between containers are enabled, even within a pod. Only specific pod to specific services may be enabled. |

# Required Reading

Recently the **Tungsten Fabric Architecture** documentation has been updated and it is a must read if you will be working with Tungsten Fabric.

It is extremely detailed and will help you orient if you run into problems.

## Tungsten Fabric Architecture

*Special thanks to **Stuart Mackie** for the effort he put into this.*

# Thank You


## Will Stevens

**CTO @ CloudOps**
https://github.com/swill

## Syed Ahmed

**Cloud Software Architect @ CloudOps**
https://github.com/syed



*Special thanks to Olivier Pilotte (Cloud Solutions Architect @ CloudOps)*
*https://github.com/olivierpilotte*

github.com/cloudops/k8s_tf_demo