Sylva: CNF as an Unit

Whitestack - 2026

whitestack

About Whitestack

Our mission

To deploy hyper-scalable digital infrastructure, based on open technologies and with the highest industry standards.

To bring unprecedented value to the IT and telecommunications industry by using open source in an efficient and innovative way, oriented to large-scale deployment and leveraging the contributions of the open source community.

Our vision

Our goal is to lead the industry in creating innovative and highly efficient digital infrastructure worldwide.

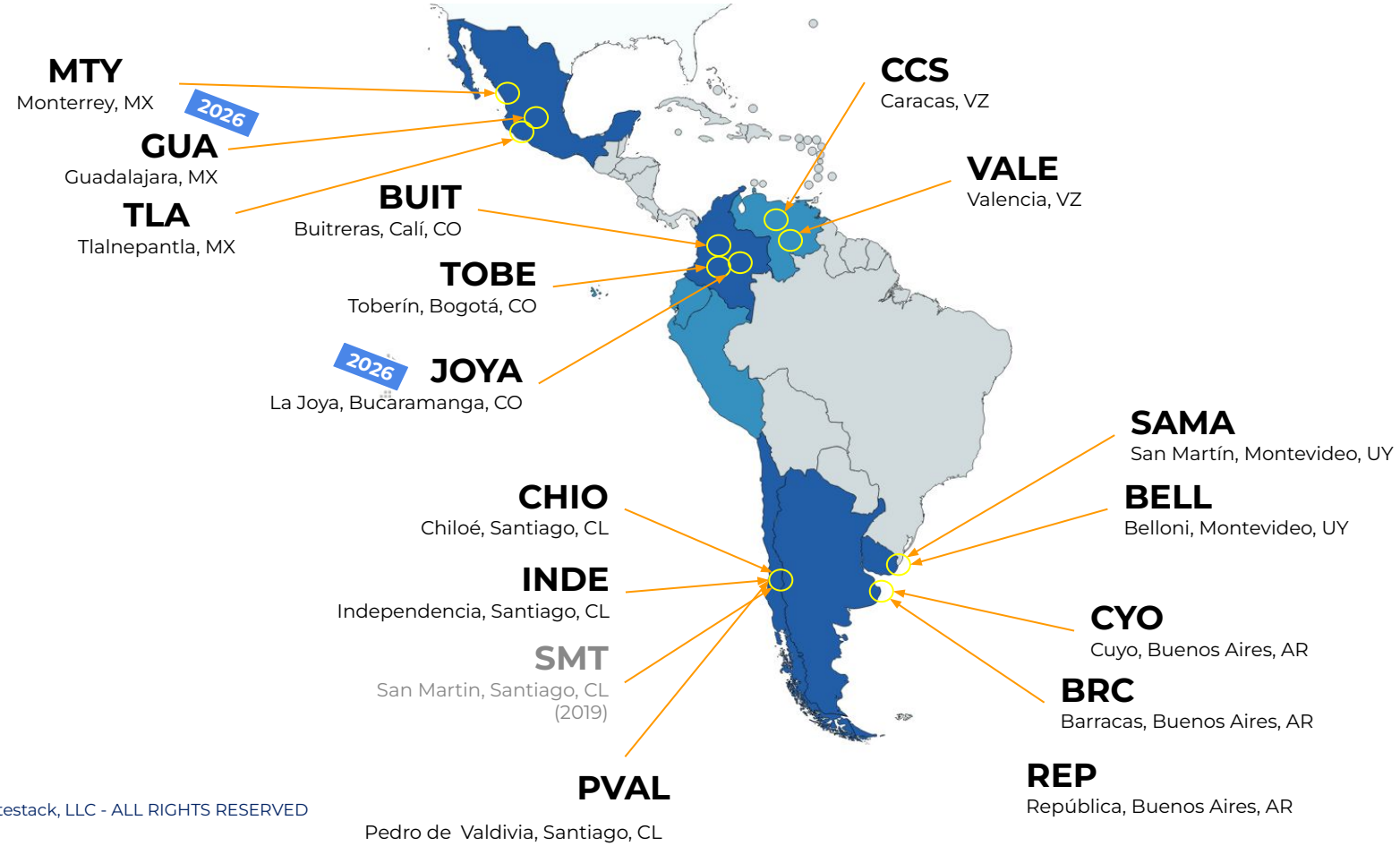
To provide agile, efficient, and universally accessible telecommunications services, transcending geographical barriers and social disparities.

To empower digital migrants in embracing a transformative mindset, inspiring them to innovate their processes for the ultimate benefit of their users.

Telco Cloud

A complete cloud solution
to support multivendor workloads
of Telco / VAS / IT,
based on open components,
deployed/supported from LATAM

Telco Clouds LATAM



Workloads deployed

Proveedor	VNF
ORACLE	vSBC
ORACLE	vDRA
ORACLE	vSR
HACOM Generando Innovación	vSMSC
THALES	vOTA
comviva A TECH MAHINDRA COMPANY	vEIR
SECURE64	vDNS
MQUEST TECHNOLOGIES	vTIR
MQUEST TECHNOLOGIES	vVMS

Proveedor	CNF
ORACLE	PCRF
ORACLE	UDR
MAVENIR	IMS
NOKIA	EPC
TITANIUM	ENUM/DNS
Radisys	MRF
MAVENIR	PCRF

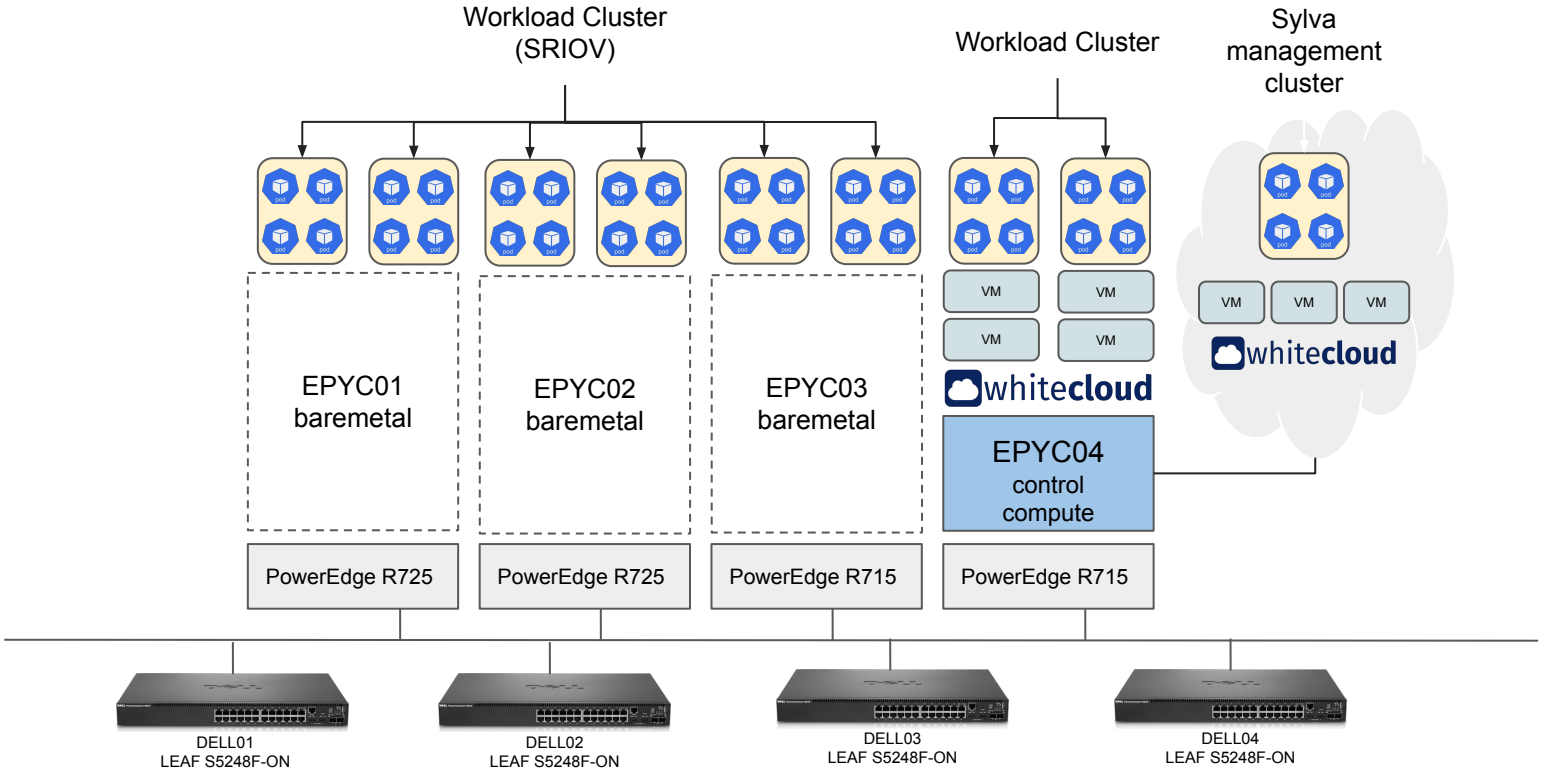
With Sylva:

WHI-1: Whitestack Validation Center

Physical location

Santiago, Chile

Architecture



¿What are Units?

The definition in the documentation is:

“Units are modular platform features (such as Rancher, Keycloak, Harbor, Vault, monitoring, etc.) that can be enabled or disabled as needed. […]”

1. Identify CNF Prerequisites

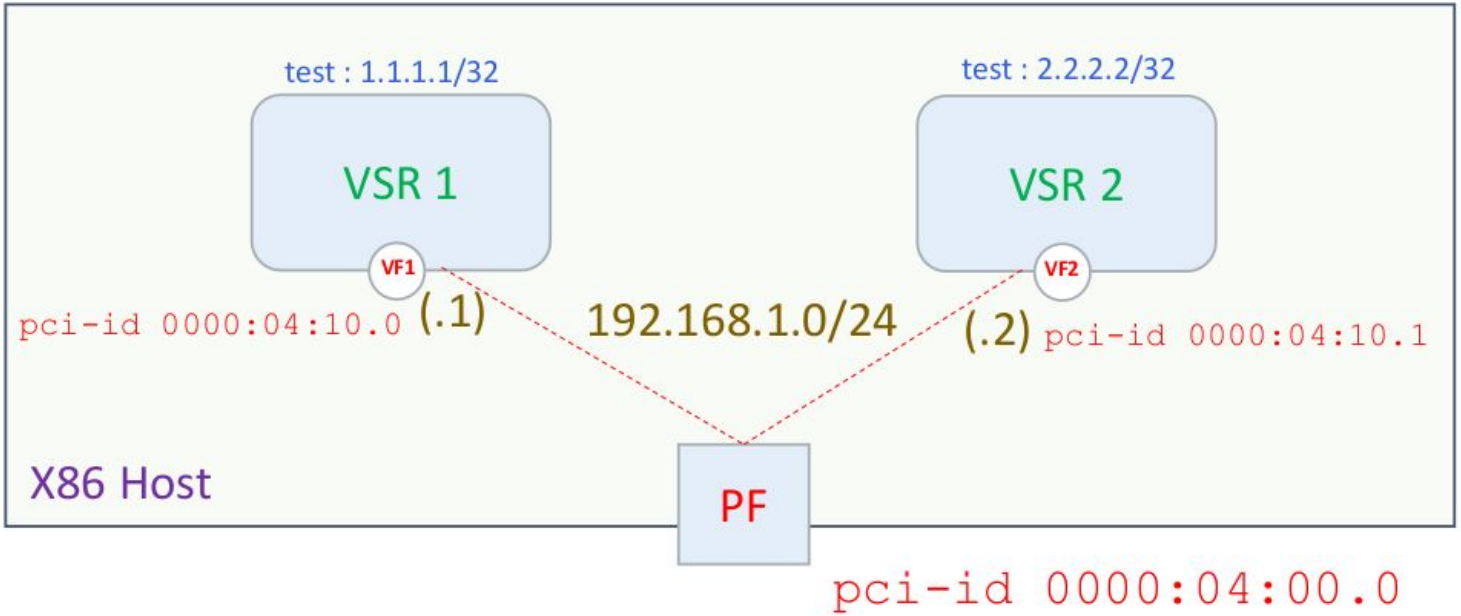
Prerequisites

- They vary from one CNF to another.
- They can be either software or hardware based.
- Many of these can be satisfied at the time of deploying the Workload Cluster.
- The vendor should be the one to specify all the prerequisites in order to streamline the validation process.

Prerequisites: 6Wind-VSR

- Host kernel compiled with features required by 6WIND fast path engine (exception path, MPLS, IPsec ...)
- Load all required kernel modules on host (vfio-pci, ip_tables, mpls_router, mpls_iptunnel, ppp_generic ...)
- Allocate hugepages on host
- Tune the kernel stack (conntracks, bridge filtering, ARP/NDP entries, logs, receive socket buffer size)
- Prepare the network devices
  - Create and configure VFs
  - Force bind Intel devices to vfio-pci driver
  - Mellanox devices must stay bound to their mlxN_core driver

Prerequisites: Success Conditions



1. An IPSec tunnel is established between the two VSR over the IPv4 network 192.168.1.0/24 which uses the VF to VF communication channel
2. The clear traffic from VSR1 1.1.1.1 to VSR2 2.2.2.2 is cyphered over that ipsec tunnel

2. Packing the CNF

Packing the CNF

- Core CNF manifests must be provided by the vendor. These can be delivered in two formats:
  - RAW
  - Helm Chart
- For both cases, we need to create a git repository to host the CNF. In the RAW case, you can manually create the Helm chart or deploy it using Flux Kustomize. In both scenarios, the resources must be organized in a specific way to ensure the deployment works as intended.
- Depending of the prerequisites there will be cases where additional manifests are required for the CNF to function correctly. For example, in VSR case, the vendor provided the helm chart for the CNF, but we needed to create the manifest to configure SR-IOV to work as intended.
- Also, if you are working with a hardened version of Sylva you need to create the Namespace to deploy the CNF. We did it using a manifest, so we can add the necessary policies to create resources in the cluster.

Packing the CNF: VSR Chart

```
┌─luis at luis-Inspiron-15-5510 in ~/Whitestack/vsr
└─o tree
.
├── Chart.yaml
├── templates
│   └── deployment.yaml
└── values.yaml

1 directory, 3 files


# As you can see, we only have one deployment file and one values file.
# We need to deploy at least 2 replicas that can interact between them to
# reach the Success Conditions.

# We also need to configure the VSR as intended. To do that, we will add
# configmaps and edit the deployment to read the configuration from them.
```

Packing the CNF: VSR Chart

```
ubuntu@sylva-deployer:~/6wind-vsr$  tree
.
└── chart
    ├── Chart.yaml
    ├── README.md
    ├── templates
    │   ├── configmap-2.yaml
    │   ├── configmap.yaml
    │   ├── deployment-2.yaml
    │   └── deployment.yaml
    └── values.yaml

# Inside our repository, we create a chart directory. Here we will put
# everything related to the VSR chart.

# We decided to add another deployment and the configmaps so we can
generate
# two replicas with different configurations working together. This is a
# design decision so we only deploy the pair of VSR as one Unit; you could
# also deploy the same chart two times (and different configurations) to
get # the same result.
```

Packing the CNF: Common Elements

```
ubuntu@sylva-deployer:~/6wind-vsr$  tree
.
├── chart
│   ├── Chart.yaml
│   ├── README.md
│   ├── templates
│   │   ├── configmap-2.yaml
│   │   ├── configmap.yaml
│   │   ├── deployment-2.yaml
│   │   └── deployment.yaml
│   └── values.yaml
├── common-elements
│   ├── namespace.yaml
│   ├── network-attachment.yaml
│   └── secret.yaml
└── kustomization.yaml

# Now that we have the VSR chart ready, we add to our repository the “common-elements”
# we need to deploy the unit. In the VSR case, we need to create the namespace, the SR-IOV
# NetworkAttachmentDefinition and the secret we are using to pull the images for the
# deployment. We also need to add the kustomization.yaml file so Sylva knows what
# resources to deploy.
```

Packing the CNF: Common Elements

```
# namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: six-wind-vsr
  labels:
    pod-security.kubernetes.io/enforce:  "privileged"
    pod-security.kubernetes.io/warn:  "privileged"
    pod-security.kubernetes.io/audit:  "privileged"
- - -
# namespace.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: six-wind-vsr

resources:
- ./common-elements/namespace.yaml
- ./common-elements/secret.yaml
- ./common-elements/network-attachment.yaml
```

3. Defining the Workload Cluster values

Defining the Values: Nodes

```
k8s_version_short: "1.30"
cluster:
 name: 6winds-cnf
 baremetal_host_default: {}
 node_classes:
   generic:
     kernel_cmdline:
       hugepages:
         enabled: true
         hugepagesz_2M: 6114
         hugepagesz_1G: 0
         default_size: 2M
       extra_options: "intel_iommu=on iommu=pt"
     non_hugepages_minimum_memory_gb: 8
     kubelet_extra_args:
       allowed-unsafe-sysctls: net.*
     kubelet_config_file_options: {}
     nodeTaints: {}
     nodeAnnotations: {}
     additional_commands:
       pre_bootstrap_commands:
         - sysctl -w net.ipv4.conf.all.rp_filter=2
         - sudo sed -i 's/^net\.ipv4\.conf\.all\.rp_filter=.*/net.ipv4.conf.all.rp_filter=2/' /etc/sysctl.conf
         - sudo sed -i -e 's/AllowGroups sylva-ops/\#AllowGroups sylva-ops/g' /etc/ssh/sshd_config
         - sudo systemctl restart sshd
         - sudo apt update && sudo apt install -y pciutils nano arping iputils-ping
     nodeLabels:
       feature.node.kubernetes.io/network-sriov.capable: "true"
```

Defining the Values: SR-IOV

```
sriov:
 node_policies:
   sriov-network-policy:
     nodeSelector:
       feature.node.kubernetes.io/network-sriov.capable: "true"
     resourceName: "sriov-network"
     numVfs: 1
     needVhostNet: true
     deviceType: "vfio-pci"
     nicSelector:
       deviceID: "1592"
       vendor: "8086"
```

Defining the Values: VSR as an Unit

```
# ./environment-values/workload-clusters/6wind-vsr/values.yaml
units:
 6wind-vsr-common:
   enabled: false
   info:
     description: Namespace with privileges for 6wind
   repo: 6wind-vsr-source
   unit templates: []
   depends on:
     cluster-ready: true
   kustomization spec:
     targetNamespace: six-wind-vsr
     path: .
 6wind-vsr:
   enabled: true
   info:
     description: 6Wind VSR
   repo: 6wind-vsr-source
   unit templates: []
   depends_on:
     sriov: true
     6wind-common: true
   helmrelease spec:
     targetNamespace: six-wind-vsr
     chart:
       spec:
         chart: ./chart
     values:
       image:
         repository: gcr.io/whitestack-private/sylva/6wind-vsr
         tag: 3.10.1.3
       imagePullSecrets:
       - name: ws-private-image-pull-secrets
```

```
source templates :
 6wind-vsr-source :
   kind: GitRepository
   spec:
     url:
https://sylva-deployment:<token>@github.com/luisvega23/6wind-vsr.git
     ref:
       branch: master
```

ANEX: What happen if the CNF are RAW manifests ?

SummaNetworks - 5GUAT

- Deployment of a 5G UAT core, production-deployed by SummaNetworks in USA, Canada, Norway (in deployment), France (PoC)
- Components:
  - **AUSF**: Authentication Server Function (1 Pod)
  - **UDM**: Unified Data Management (1 Pod)
  - **UDR**: Unified Data Repository (1 Pod)
- Database layer needed for the 5G Core:
  - Percona-DB (1 Pod for the database, 1 for the HA Proxy)
  - Percona-Operator (1 Pod)
  - Redis (1 pod for the operator, 1 pod for the db)
- The CNF does not have any specific networking requirements (e.g., SR-IOV, dedicated network cards, etc.), but it does require deployment on a high-speed storage unit (SSD).
- Summa Networks does not provide a Helm chart; instead, the deployment is done directly using RAW Kubernetes manifests.

Packing the CNF: SummaNetworks

```
ubuntu@sylva-deployer:~/summanetwork$ tree
.
├── CNFs
│   └── summanetworks
│       ├── AUSF
│       │   ├── configmap.yaml
│       │   ├── deployment.yaml
│       │   ├── kustomization.yaml
│       │   └── service.yaml
│       ├── UDM
│       │   ├── configmap.yaml
│       │   ├── deployment.yaml
│       │   ├── kustomization.yaml
│       │   └── service.yaml
│       ├── UDR
│       │   ├── component
│       │   │   ├── configmap.yaml
│       │   │   ├── deployment.yaml
│       │   │   ├── kustomization.yaml
│       │   │   └── service.yaml
│       │   └── updates
│       │       ├── jobs.yaml
│       │       └── kustomization.yaml
│       └── kustomization.yaml
…
# Inside our repository, we create a directory for any component of the
CNF. # In this case the CNF isn't 1:1 with the Unit concept.
```

Packing the CNF: SummaNetworks

```
ubuntu@sylva-deployer:~/summanetwork$ tree
.
├── CNFs
│   └── summanetworks
│       ├── …
├── common-elements
│   ├── kustomization.yaml
│   └── suma5g-namespace.yaml
├── redis
│   ├── redis-cluster
│   │   ├── cluster.yaml
│   │   └── kustomization.yaml
│   ├── redis-database
│   │   ├── db.yaml
│   │   └── kustomization.yaml
│   └── redis-operator
│       ├── deployment.yaml
│       ├── kustomization.yaml
│       ├── redis-operator-crd.yaml
│       ├── role.yaml
│       └── service.yaml
…
# More components to the CNF.
```

Packing the CNF: SummaNetworks

```
# environment-values/workload-clusters/summanetworks-5guat/values.yaml
source_templates:
 summanetwork:
   kind: GitRepository
   spec:
     url: https://sylva-deployment:token@github.com/whitestack/summanetwork.git
     ref:
       branch: main
 percona:
   kind: GitRepository
   spec:
     url: https://github.com/percona/percona-helm-charts.git
     ref:
       branch: main
```

Packing the CNF: SummaNetworks

```
# ./environment-values/workload-clusters/summanetworks-5guat/values.yaml
units:
 summa5g-deployment-namespace:
   …
   repo: summanetwork
   …
   kustomization_spec:
     targetNamespace: summa5g-deployment
     path: ./common-elements/
 pxc-operator:
   …
   repo: percona
   depends_on:
     summa5g-deployment-namespace: true
   helmrelease_spec:
     targetNamespace: summa5g-deployment
     chart:
       spec:
         chart: ./charts/pxc-operator
 udr:
   enabled: true
   …
   depends_on:
     redis-database: true
     pxc-db: true
     udr-job: true
   repo: summanetwork
   kustomization_spec:
     targetNamespace: summa5g-deployment
     path: ./CNF/summanetworks/UDR/component/
```

Questions ?

Contact email:
lvega@whitestack.com

Sylva: CNF as an Unit

Whitestack - 2026

whitestack