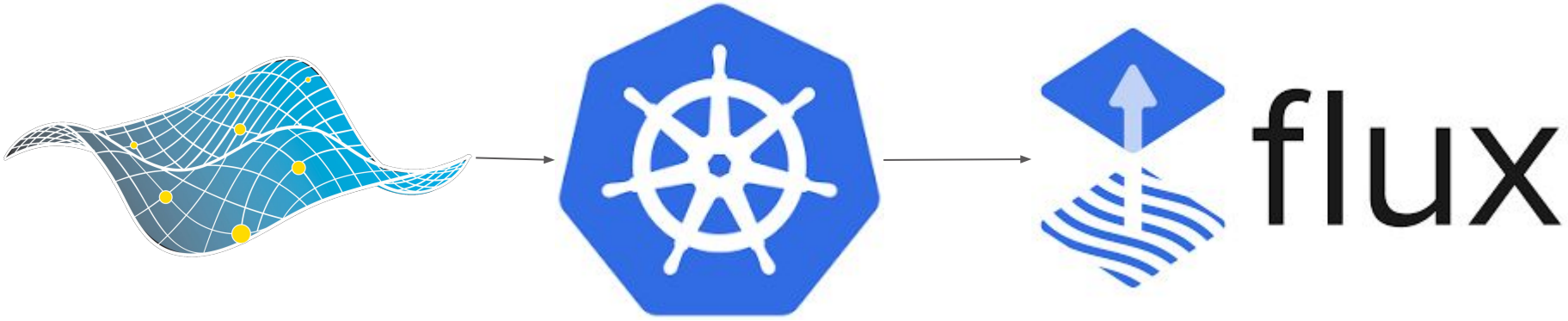
Service Deployment in FABRIC at CERN

Fengping Hu
Enrico Fermi Institute
University of Chicago

Throughput Computing 2023

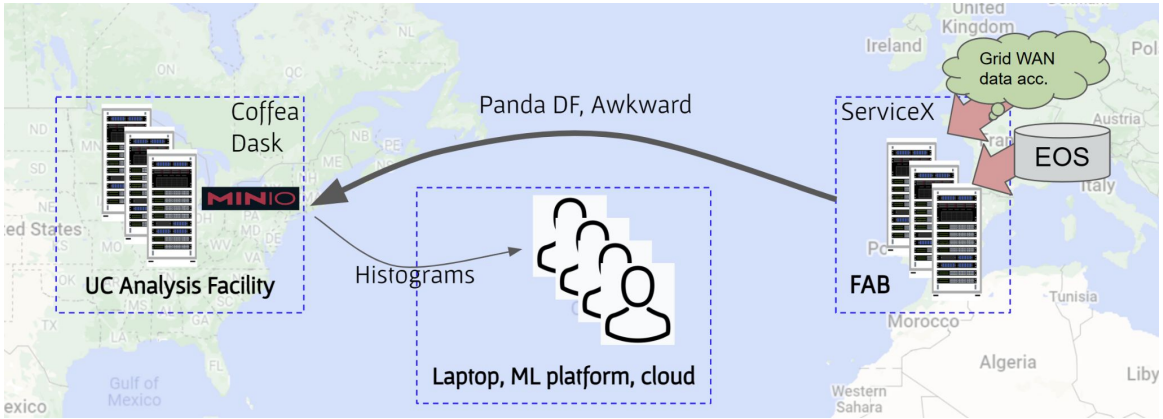
7/12/23

Overview



A demonstrator to inform future LHC computing models

- Deploy ServiceX at CERN (to filter and reformat data on the Tier0)
- Deliver only columnar data objects to analysis facilities
- Examine resulting 1) turn around time and 2) transatlantic bandwidth reduction



A bit of background

- FABRIC is an NSF funded network testbed operated by ESnet where one can run experiments in areas of networking, distributed computing, storage, ML, etc.
- FAB (FABRIC Across Borders). It added five international sites to the FABRIC testbed, including CERN.

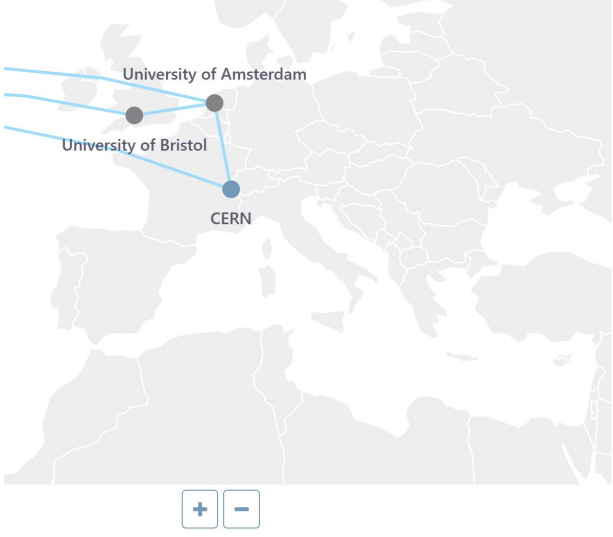
Main components:

- an everywhere programmable network interconnected by dedicated optical links
- cutting-edge infrastructure for computer science, AI, data-intensive research
- software and support

FAB resources @ CERN

Nodes:

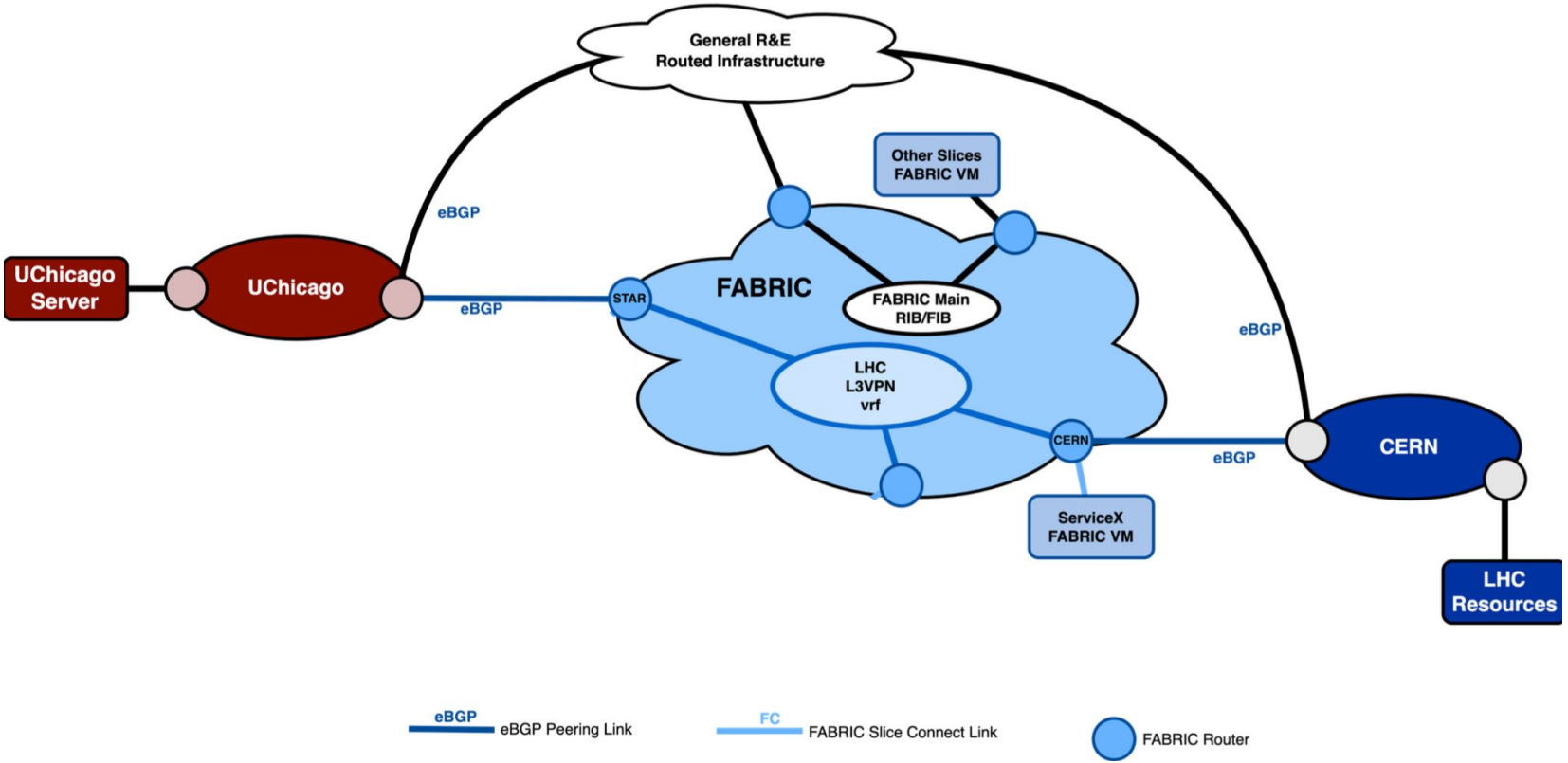
- 1 head
  - R7515, single AMD 7532
- 2 slow net
  - R7525, dual AMD 7532, 512 GB RAM
  - 2x 100Gbps, 4x25Gbps
  - 2x NVidia T4
- 3 fast net
  - Same as slow but with 6x 100 Gbps and 2x SN1000 FPGA
- 1 GPU
  - Same as slow net but with 2x NVidia RTX6000



CERN (CERN)

Status	Active
Cores	120/384
Disk (GB)	74480/75000
RAM (GB)	1556/3072
GPU	6/7
NVME	22/22
SmartNIC	10/10
SharedNIC	742/762
FPGA	0/0

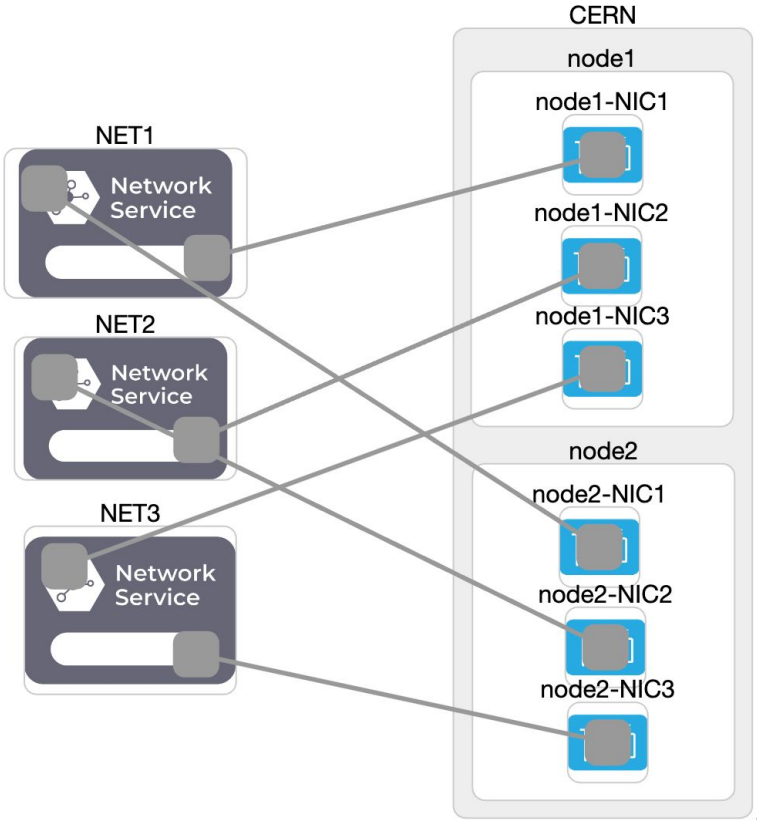
Network layout



Provision and manage resource in FABRIC

- Create Slice in Portal (Slice Builder UI is still a work in progress)
- Create Slice in JupyterHub
  - a private JuypyterHub environment will be built on first login
  - User's FABRIC experiment notebooks will be persistently stored
  - FABRIC includes a set of example notebooks that demonstrate the use of the FABRIC Python API.

The ServiceX slice



- VMs are created with 3 l3network
  - NET1 – IPv4(IPv6 only kubernetes isn't well supported by kubespray)
  - NET2 – IPv6 peering for ServiceX data network
  - NET3 – IPv6 public for ServiceX frontend
- 10 nodes
  - Site: CERN
  - Host: No selection
  - Cores: 62
  - RAM: 128G
  - Disk: 500G
  - VM Image: default_ubuntu_20

Customizing the nodes(node.execute)

- Rename Network Interface so the name to network mapping is consistent across nodes
- Configure netplan to disable RA and DHCP on IPv6 links ens8 and ens9
- Configure IPs for the NET1 and NET3. The IPs for the NET2 is not configured on the node but managed in Kubernetes
- Configure policy based routing to avoid asymmetric routing(change default route to use the NET3 gateway and add rules for the management network)
- Config node to use NAT64 – Access non-IPv6 services (i.e. GitHub) from IPv6 FABRIC nodes

Kubernetes installation with Kubespray

Kubespray is a composition of Ansible playbooks, inventory, provisioning tools, and domain knowledge for generic OS/Kubernetes clusters configuration management tasks

```
all:
  children:
    kube_control_plane:
      hosts:
        node1:
    k8s-cluster:
      children:
        kube_control_plane:
        kube_node:
      vars:
        dns_min_replicas: 1
        supplementary_addresses_in_ssl_keys: ('2602:fcfb:1d:2::2')
        ansible_ssh_common_args: '-F  ssh_config  -J uid@bastionhost  -i slice_key'
  hosts:
    node1:
      access_ip: 10.143.1.2
      ansible_host: 2001:400:a100:3090:f816:3eff:fe1c:385f
      ip: 10.143.1.2
      ip6: 2602:fcfb:1d:2::2
```

- Inventory file generated with jinja templating from the slice creation notebook
- One command to install Kubernetes
  `ansible-playbook -i inventory/fabric/hosts.yaml --become --become-user=root -u ubuntu cluster.yml`
- Enable natoutgoing for the IPv6 Kubernetes cluster network
  `Kubectl edit ippool default-pool-ipv6 (natOutgoing: true)`

Kubernetes installation with kubespray II

```
diff -r fabric/group_vars/k8s_cluster/k8s-cluster.yml
sample/group_vars/k8s_cluster/k8s-cluster.yml
< kube_network_plugin_multus: true
---
> kube_network_plugin_multus: false
< enable_dual_stack_networks: true
---
> enable_dual_stack_networks: false
< kube_proxy_strict_arp: true
---
> kube_proxy_strict_arp: false
< enable_nodelocaldns: false
---
> enable_nodelocaldns: true
diff -r fabric/group_vars/k8s_cluster/k8s-net-calico.yml
sample/group_vars/k8s_cluster/k8s-net-calico.yml
< calico_ip6_auto_method: "kubernetes-internal-ip"
```

- Configure IP autodetection for Calico nodes to ensure the correct IP address is used for routing
- Configure arp_ignore and arp_announce to avoid answering ARP queries from kube-ipvs0 interface for MetalLB to work.

  kube_proxy_strict_arp: true
- Addons
  - Metallb, ingress controller, certmanager ...

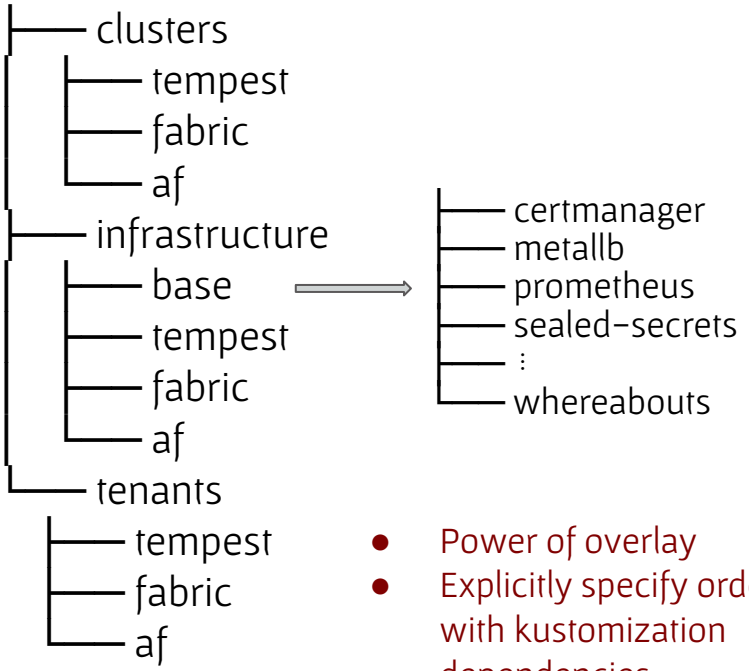
Apps and infrastructure deployment with Flux

Flux is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible

- GitOps for apps and infrastructure
- Declarative & Automated
- Auditable

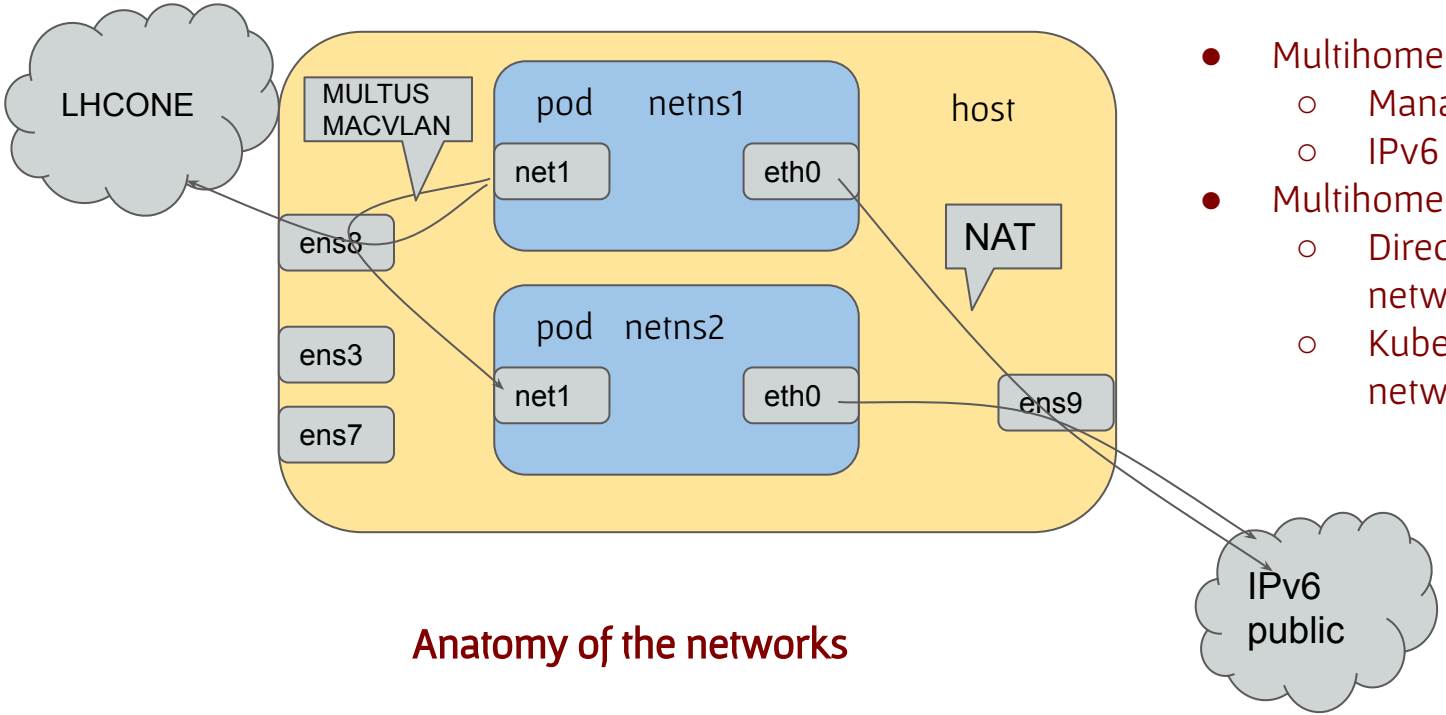
Repository structures

- Platform admin repo – **Shared** by a fleet of clusters
  - clusters dir contains the Flux configuration per cluster
  - infrastructure dir contains common infra tools such as admission controllers, CRDs and cluster-wide policies
  - tenants dir contains namespaces, service accounts, role bindings and Flux custom resources for registering tenant repositories



- Power of overlay
- Explicitly specify order with kustomization dependencies

Frontend network and data network for ServiceX



Anatomy of the networks

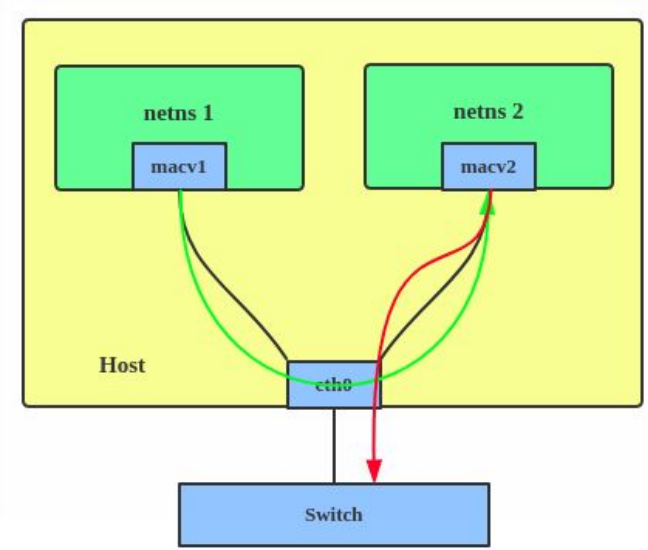
- Multihomed host
  - Management network(ens3)
  - IPv6 public network(ens9)
- Multihomed pod
  - Directly attached peering network(net1, ens8)
  - Kubernetes cluster network/master plugin(eth0)

Attach data network to pod with Multus and Whereabout


```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-conf
spec:
  config: '{
      "cniVersion": "0.4.0",
      "type": "macvlan",
      "master": "ens8",
      "mode": "bridge",
      "ipam": {
        "type": "whereabouts",
        "range": "2602:FCFB:0100:0:10::-2602:FCFB:0100:0:20::/64",
        "gateway": "2602:FCFB:0100::1",
        "routes": [
          {"dst": "2605:9a00:10:200a::/64",
          "gw": "2602:FCFB:0100::1"},
          {"dst": "2001:1458:d00::/48",
          "gw": "2602:FCFB:0100::1"},
          {"dst": "2001:1458:301::/48",
          "gw": "2602:FCFB:0100::1"},
          {"dst": "2001:1458:303::/48",
          "gw": "2602:FCFB:0100::1"}
        ]
      }
    }'
```

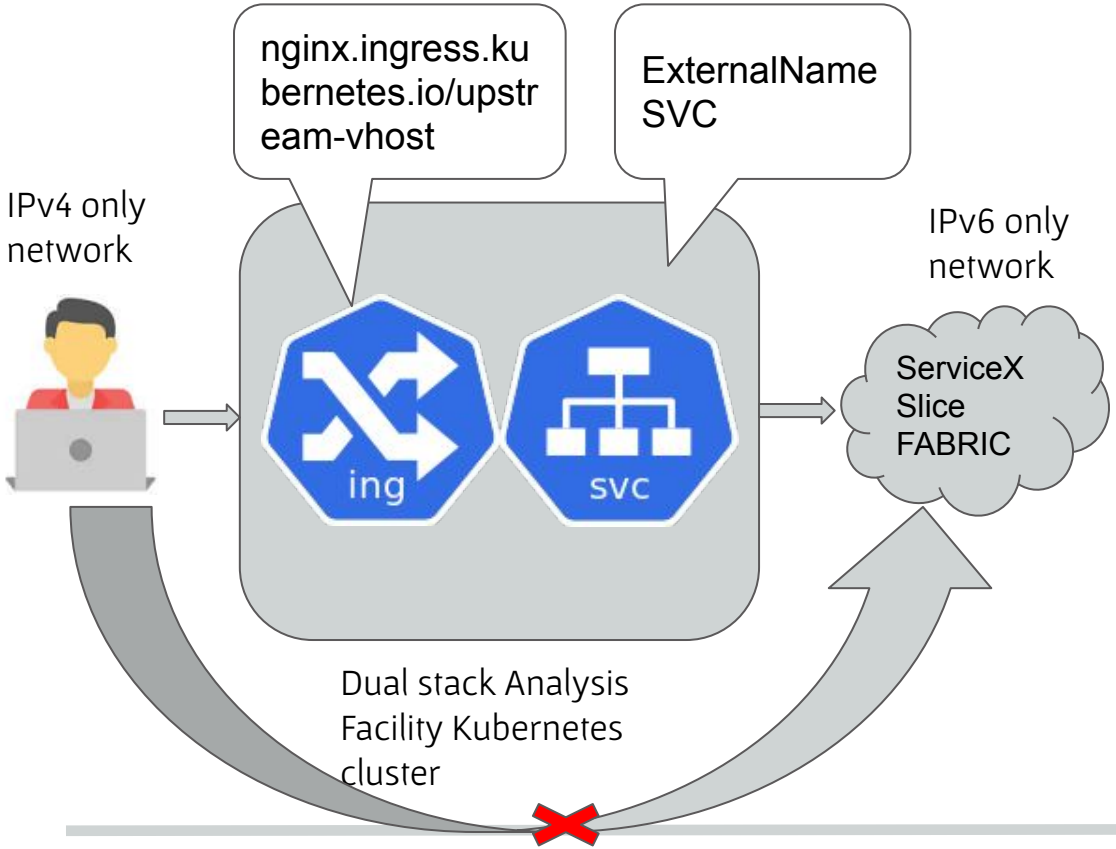
- Multus CNI is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods.
- Whereabouts- An IP Address Management (IPAM) CNI plugin that assigns IP addresses cluster-wide.

MACVLAN for container and caveats



- With MACVLAN, you can create multiple interfaces with different Layer 2 (that is, Ethernet MAC) addresses on top of a single one
- Containers comes and goes quickly which would result in rapid change of the MAC address associated with the same IP address. We observed a problem with routers due to this.
- The symptom is that we can't ping the container IP from outside in the first few minutes, but reaching out from the container is fine. Not really a problem for us because we initiates connection from container.

Access frontend from IPv4 network



- Limited IPv6 availability cause inconvenience for frontend access
- We bridge the gap with our on-prem dual stack cluster

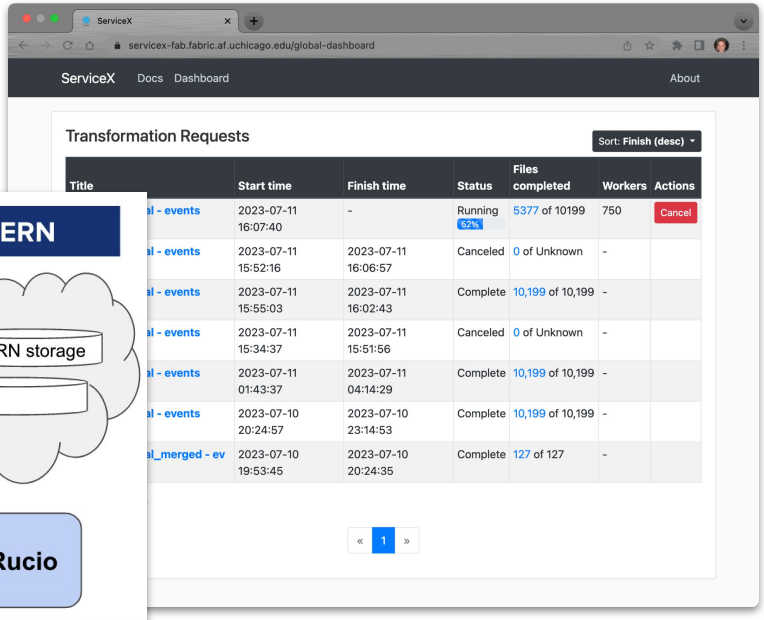
Steps to redeploy everything

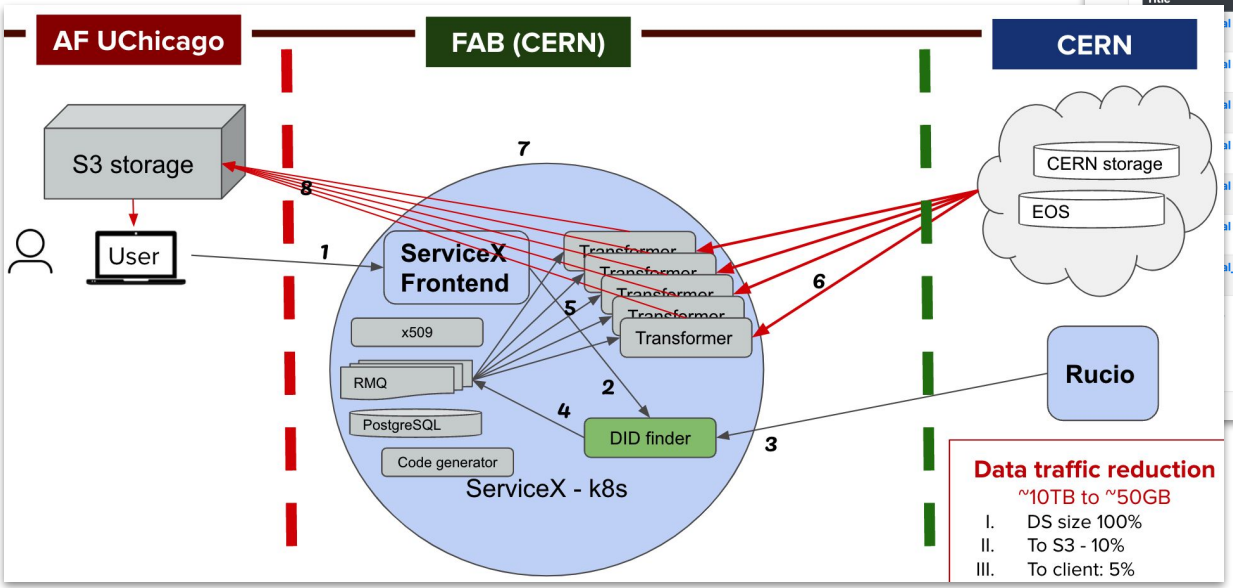
1. Run the Jupyter notebook to create a FABRIC slice.
2. Run the Jupyter notebook to install Kubernetes
3. Bootstrap Flux on the cluster connected to the GitHub repository
4. Restore the SealedSecrets from the S3 backup
5. Watch everything get recreated and access the ServiceX web frontend to verify

Ready to Deploy ServiceX

[Server Side Data Delivery using FAB at CERN]

(results run yesterday)





Conclusion

- We used the FAB extension of the testbed to CERN
- The FABRIC interface allowed us to provision and manage a data delivery service where we normally cannot
  - Here, we put the service close to the storage origin to test a new LHC analysis facility configuration
- We plan to scale this up and **extend** IRIS-HEP analysis grand challenges this Fall

Thank you!

This work supported in part by NSF awards:

- #1935966 Mid-Scale RI-1 (M1:IP): FABRIC: Adaptive Programmable Research Infrastructure for Computer Science and Science Applications
- #2029261 Collaborative Research: IRNC: Testbed: FAB: FABRIC Across Borders
- #1836650 S2I2: Institute for Research and Innovation in Software for High Energy Physics (IRIS-HEP)