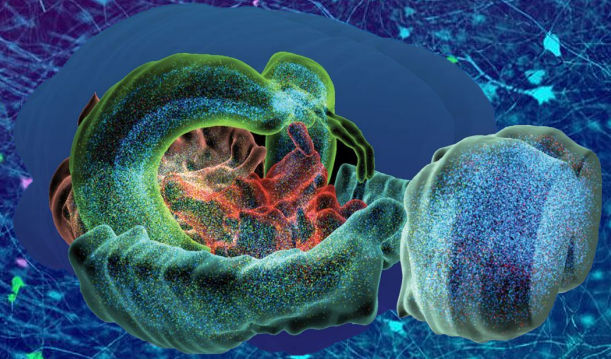
EPFL Blue Brain Project



Kubernetes from Scratch for Neuroscientific Research

Carolina Lindqvist & Daniel Fernández

KubeCon | CloudNativeCon
Europe 2023

Agenda

- About us
- Target audience
- How did our interest in Kubernetes start
- Prototyping Kubernetes
- Blue Brain Project infrastructure overview
- Cluster running and maintenance
- Kubernetes use cases within the Blue Brain Project
- The process of migrating to Kubernetes
- A Kubernetes user’s point of view

About us

Carolina Lindqvist
SRE @ EPFL

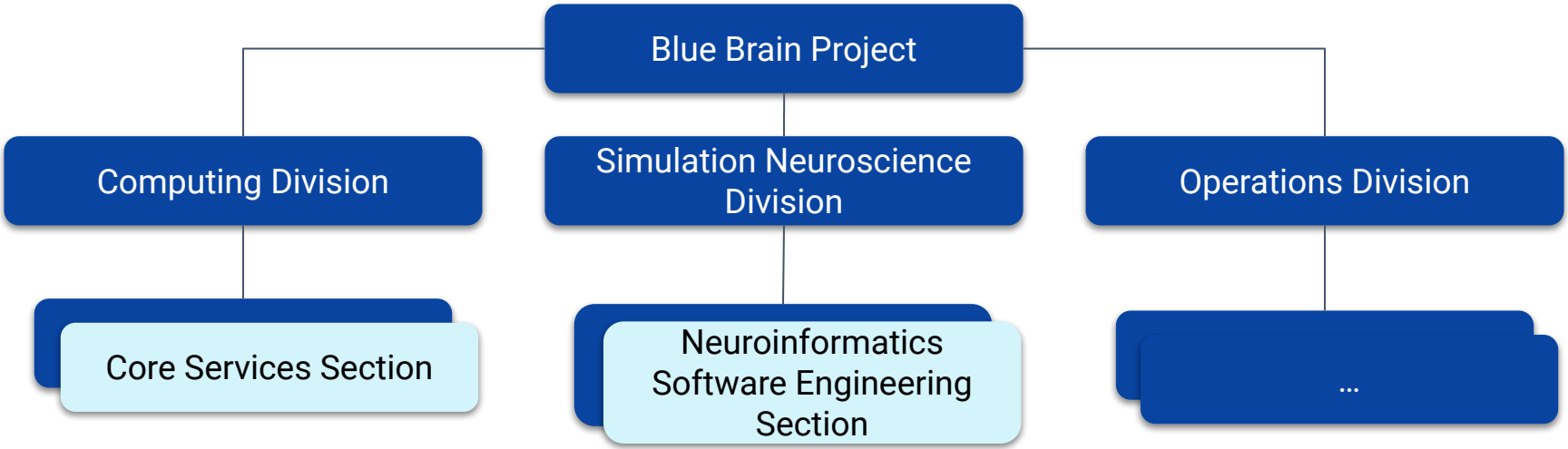
carolina.lindqvist@epfl.ch
https://github.com/carolili

Daniel Fernández Rodríguez
SRE @ EPFL

daniel.fernandez@epfl.ch
https://github.com/danifr

About the Blue Brain Project

- Goal: Digitally reconstructing and simulating a mouse brain
- Pioneering simulation neuroscience
- Better understanding of the human brain with potential applications in health and disease research



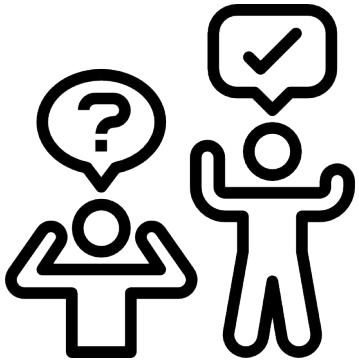
Content

- Ideas for how to approach an on-premise cluster setup
- The ideas can also be applied to a cloud environment
- Where to start

Useful prerequisites

- Basic knowledge about Kubernetes architecture
- Basic knowledge about some resource types
- Knowledge of Helm and Helm chart is a plus

How did our interest in Kubernetes start



OPENSHIFT

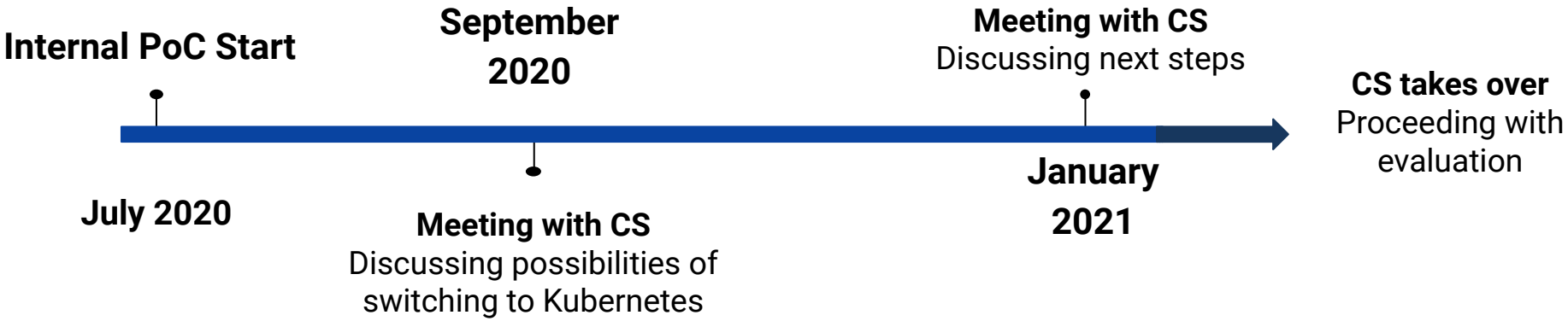
NGINX
Part of F5

kubernetes

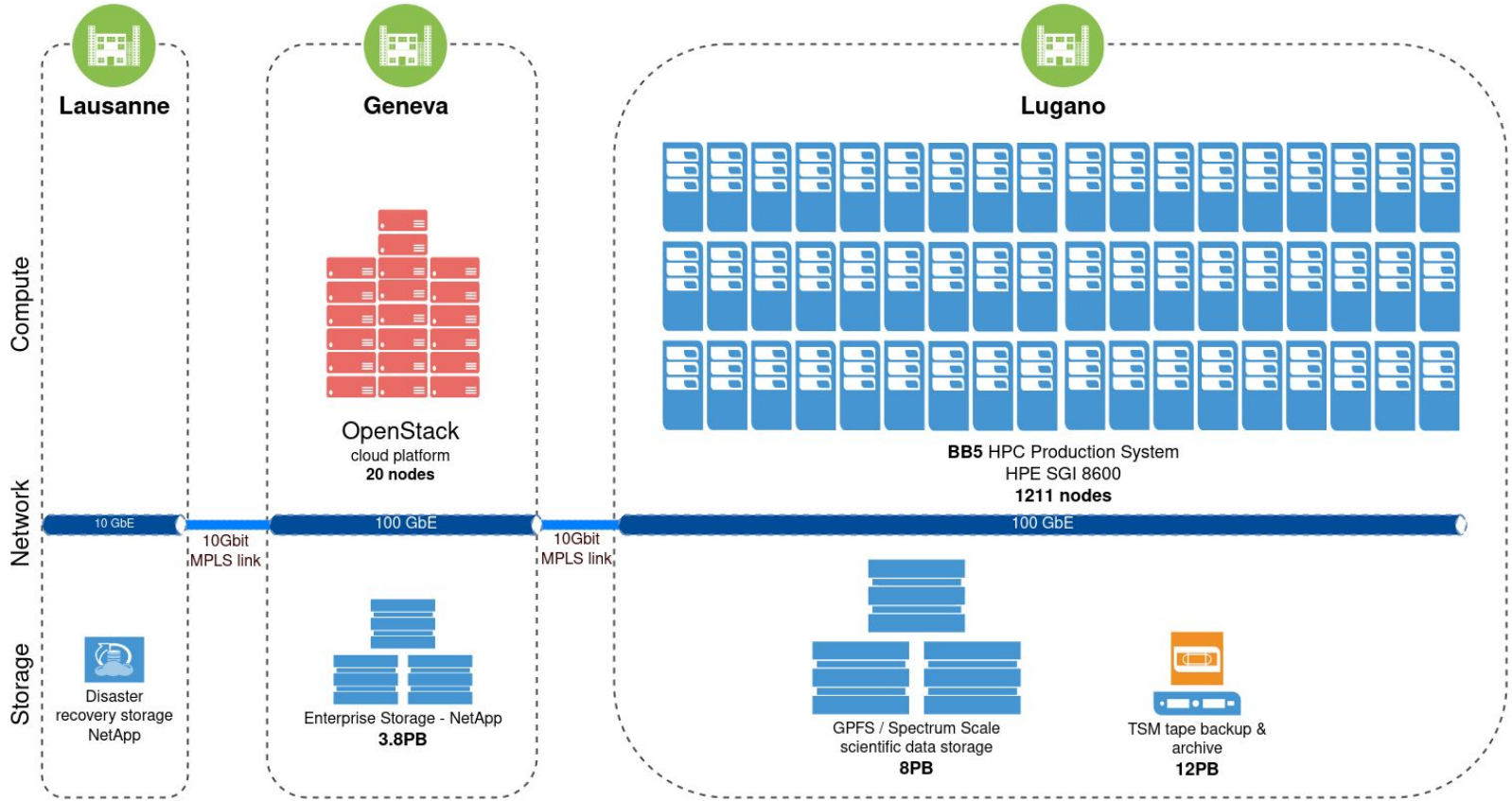
HELM

Prototyping

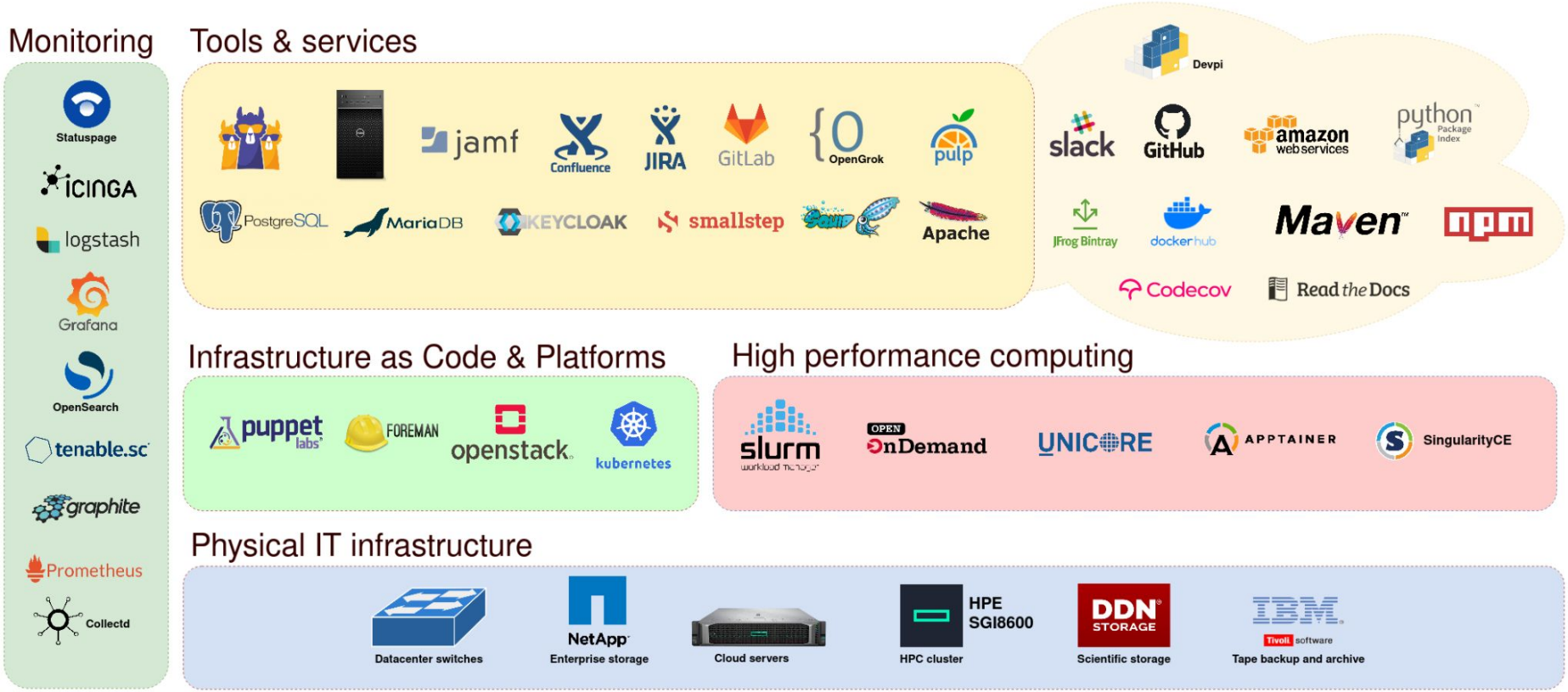
- How to deploy Kubernetes on Openstack
- Creating an Ubuntu-based Kubernetes cluster for internal testing
- Documenting the installation and deployment process
- Presenting the test results to Daniel's team and discussing the next steps



About Blue Brain Project Infrastructure



About Blue Brain Project Infrastructure

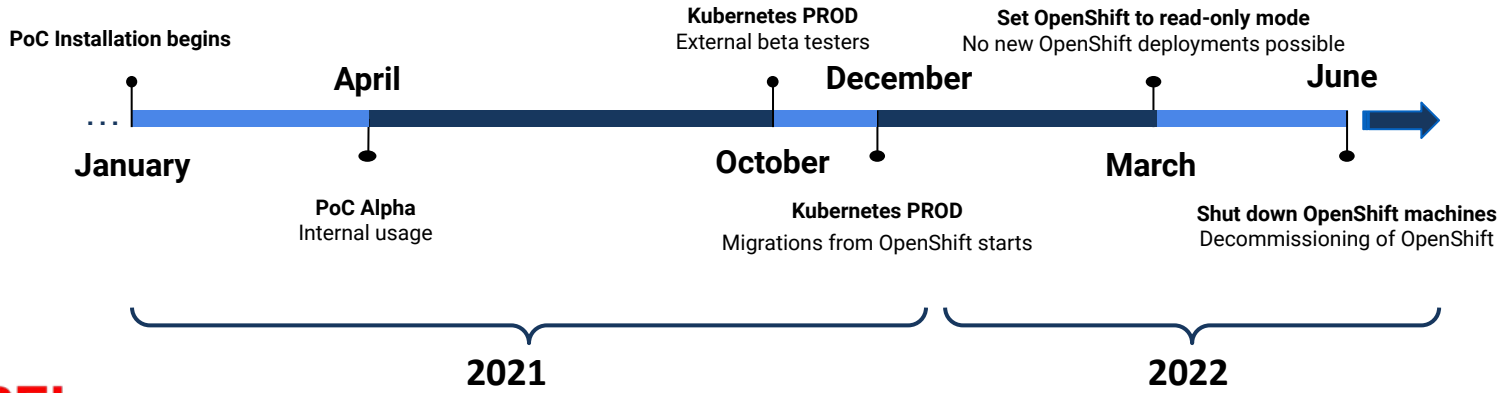


Pre-Kubernetes (2021)

- OpenShift Origin 3.11 (Based on Kubernetes 1.11)
  - Released in Oct 11, 2018
- People who worked on its deployment had left the team
- Deployment was relying on an Ansible playbook
- Day-to-day operations were tedious and cumbersome:
  - Adding new worker nodes
  - Changing the quota of a project
- Lack of support for new features which users were requesting:
  - Helm
  - GitOps
  - Ingress

Proof Of Concept - Kubernetes (2021)

- Kubernetes 1.20 (containerd/runc)
- On OpenStack VMs
- RHEL operating system
  - SElinux enabled and set to enforcing :)
- Installation and initial cluster configuration: Puppet
- GitOps to automate the configuration of the clusters



Features

Kubernetes 1.21

Calico

Ingress Nginx controller

Kyverno

GitLab Registry

Flux

mozilla/sops

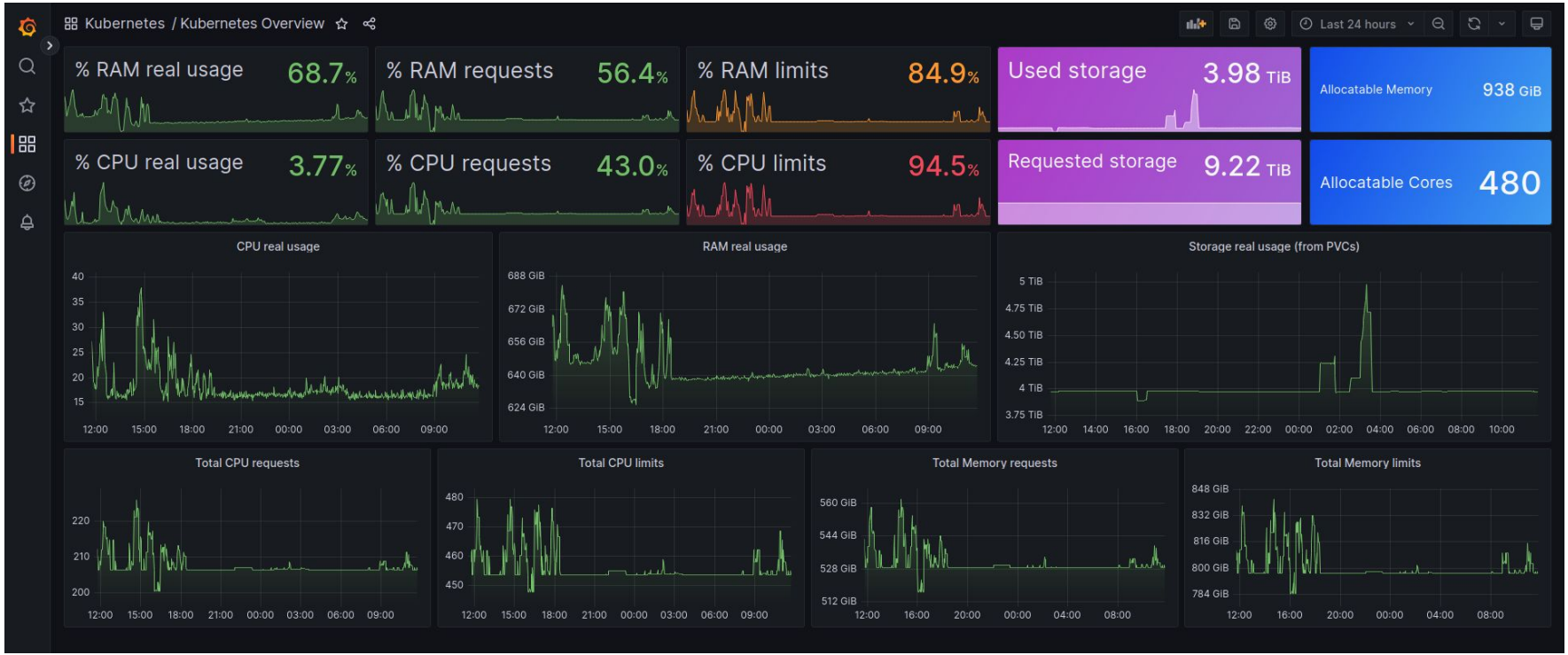
Migration from OpenShift

- We audited everything running on OpenShift and contacted those in charge
- Majority of users managed their migrations themselves
  - In most cases: `oc get <resource> -o yaml` and then re-apply those yaml on Kubernetes
- No way to automatically migrate OpenShift volumes to Kubernetes
  - The data migration process depended upon the app, but usually it was a matter of:
    - `oc cp / kubectl cp`
- Migration from OpenShift Container registry to GitLab Registry:
  - Push image to new registry

Cluster deployment

- 100% on-premises
- Kubernetes on OpenStack
- RHEL8 (started with RHEL7)
- Current version: 1.25
  - 5 upgrades in ~2 years
- 2 clusters: **PROD** and **DEV**
  - 3 control plane nodes + ‘N’ worker nodes
  - PROD & DEV have the same configuration (enforced by Flux)
    - DEV: fewer and smaller worker nodes
  - We test every change / upgrades first on DEV

Kubernetes usage overview



Multitenancy

- Heavily based on: flux2-multi-tenancy although we ended up doing our own thing
- Each team has its own namespace
  - Defined ResourceQuotas
  - RoleBinding references an LDAP group
    - Each member has admin access within the namespace
    - User groups are passed in the OIDC token
  - Pods within a namespace are by default isolated (NetworkPolicy)
    - Ingress and egress traffic blocked unless specified otherwise
  - Optional Flux integration

Installation

- Installation with puppetlabs-kubernetes:
  - caveats: etcd/Kubernetes certificates are generated before installation
    - They are tied to node's IPs/hostnames

```
$ docker run --rm -v /tmp/k8s/production:/mnt
  -e OS=redhat
  -e VERSION=1.22.2 -e CONTAINER_RUNTIME=cri_containerd
  -e CNI_PROVIDER=calico-tigera -e CNI_PROVIDER_VERSION=3.20.1
  -e ETCD_INITIAL_CLUSTER="<node1>:<node1_ip>,<node2>:<node2_ip>,<node3>:<node3_ip>"
  -e ETCD_IP="%{::ipaddress}"
  -e KUBE_API_ADVERTISE_ADDRESS="<cluster_ip>"
  puppet/kubetool:7.0.0

$ ls -l /tmp/k8s/production
total 52K
-rw-r--r-- 1 root root 9.1K Oct  4  2021 node1.yaml
-rw-r--r-- 1 root root 9.1K Oct  4  2021 node2.yaml
-rw-r--r-- 1 root root 9.1K Oct  4  2021 node3.yaml
-rw-r--r-- 1 root root  15K Oct  4  2021 Redhat.yaml
```

Installation: Redhat.yaml


```
kubernetes::kubernetes_version: 1.22.2
kubernetes::container_runtime: cri_containerd
kubernetes::cni_network_provider: https://docs.projectcalico.org/archive/3.20.1/manifests/calico.yaml
kubernetes::cni_pod_cidr: 192.168.0.0/16
kubernetes::cni_provider: calico-tigera
kubernetes::etcd_initial_cluster:
<node1>=https://<node1_ip>:2380,<node2>=https://<node2_ip>:2380,<node3>=https://<node3_ip>:2380
kubernetes::etcd_peers:
 - <node1_ip>
 - <node2_ip>
 - <node3_ip>
kubernetes::etcd_ip: "%{::ipaddress}"
kubernetes::kube_api_advertise_address: <cluster_ip>
kubernetes::token: bf1bce.d0ef29da7bfc92658
kubernetes::etcd_ca_crt: |
  -----BEGIN CERTIFICATE-----
  MIIC7jCCAdagAwIBAgIUTWX6UV38XofpTGFKBtdCrM36d9IwDQYJKoZIhvcNAQEL
  [...]
kubernetes::kubernetes_ca_crt: |
  -----BEGIN CERTIFICATE-----
  MIIC+jCCAeKgAwIBAgIUBbK3L5d9H/mNwZ+rNgJ68z1qGM8wDQYJKoZIhvcNAQEL
  [...]
```

Installation

- Ensure the certificates are created on the nodes
- Bootstrap an initial etcd cluster
- Install Kubernetes via Kubeadm (init/join)
- Install and configure CNI Calico-Tigera

- Flux bootstrapping and initial configuration
- Enable OIDC auth by adding the corresponding flags in API server

- Kyverno + ClusterPolicy
- cert-manager
- Trident (CSI maintained by NetApp)
- Kubernetes dashboard
- ingress-nginx
- gitlab-runners
- …

Cluster running and maintenance

- Quota changes managed with Flux
- Adding extra worker nodes
  - Create new VM, put VM in the correct hostgroup, and run Puppet
- Cluster upgrades
  - In-place
  - Puppet and some helper scripts
    - Update Hiera variable with new Kubernetes version
    - Push changes and run Puppet
      - `kubeadm kubectl kubelet` packages will be upgraded
  - Control plane nodes
    - `kubeadm upgrade apply --certificate-renewal=false v1.25.8`
  - Worker nodes
    - drain, run Puppet, uncordon

Core Services migration to Kubernetes and future

- Applications:
  - GitLab Runners
  - Dummy applications to verify cluster status
  - Slack bots
  - OpenSearch
  - Various Prometheus exporters
    - Pgpool
    - Foreman
    - Blackbox
- Future plans:
  - Cluster Autoscaler
  - Service Mesh with Linkerd

Overall experience

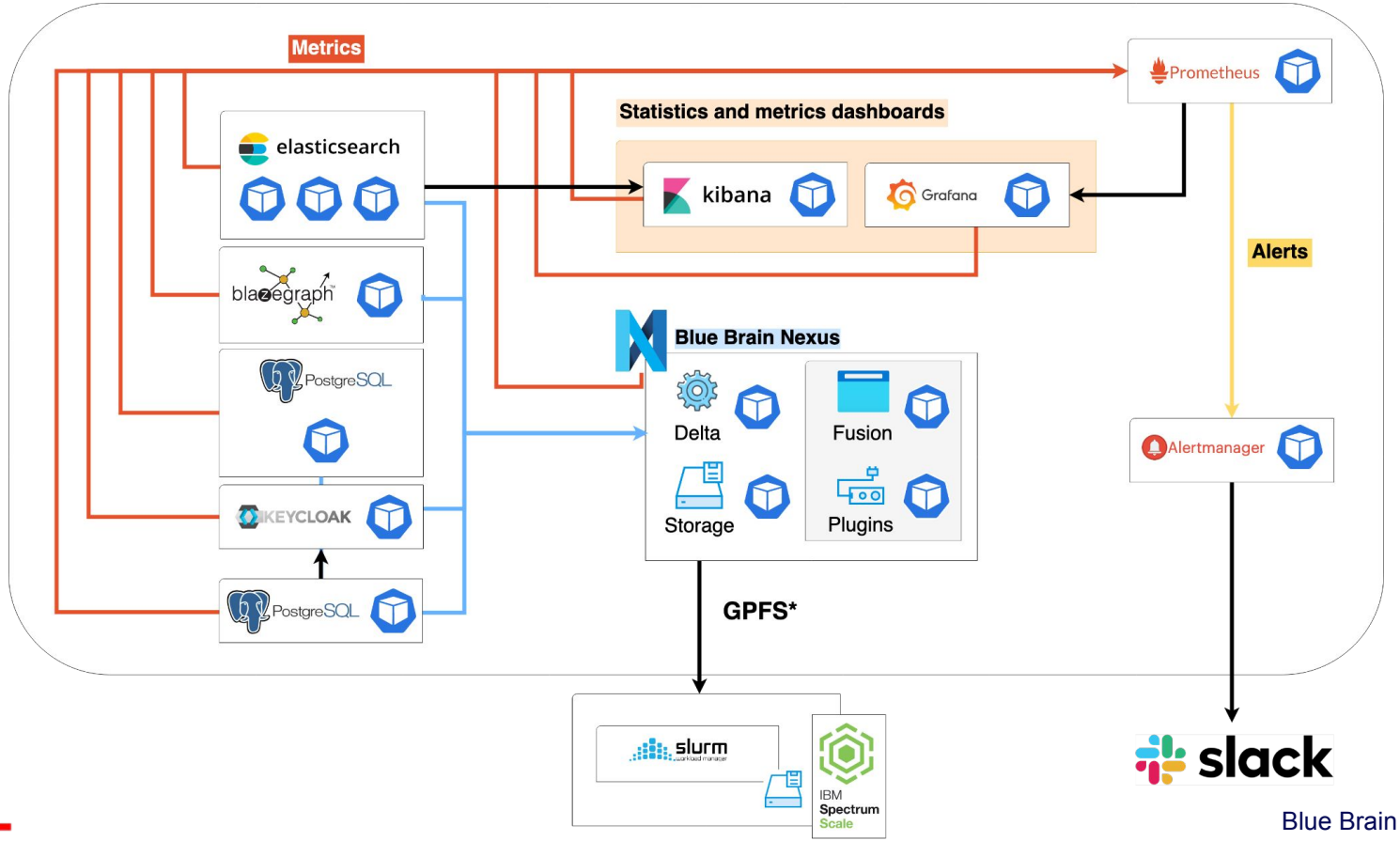
- We are very happy with Kubernetes
- Huge YAMLs and Flux core concepts can be intimidating at first
- People were using OpenShift web UI - we thought they would miss it
  - End users interact with the cluster via: kubectl CLI, Lens, k9s, Kubernetes dashboard
- For end-users we wrote easy to follow how-to guides:
  - Connect to cluster (via kubelogin)
  - Use the Kubernetes dashboard
  - Change pod network policies to allow ingress/egress traffic
  - Expose applications via ingress
  - Quick start with Flux

Example: Blue Brain Nexus

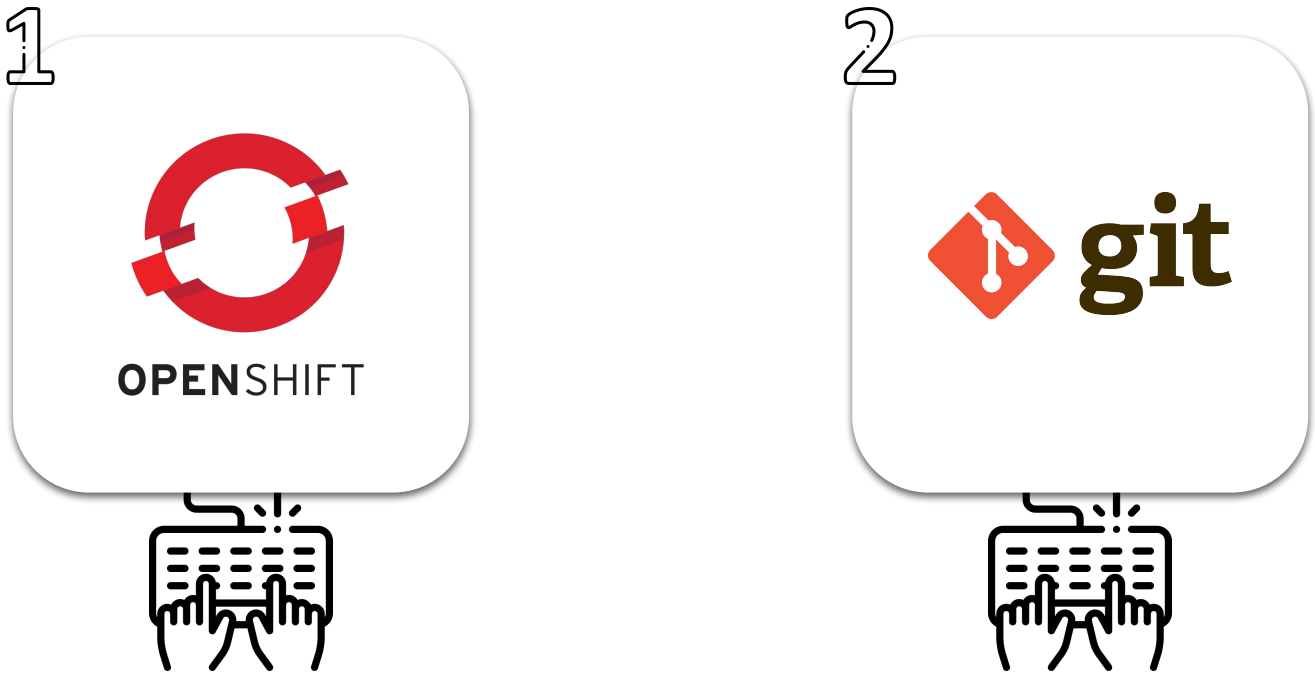
- Main data management platform at BBP to organise all neuroscience data
- Backend for knowledge graph management
- Web interface to provide user-friendly access
- Fully deployed in Kubernetes

https://bluebrainnexus.io/

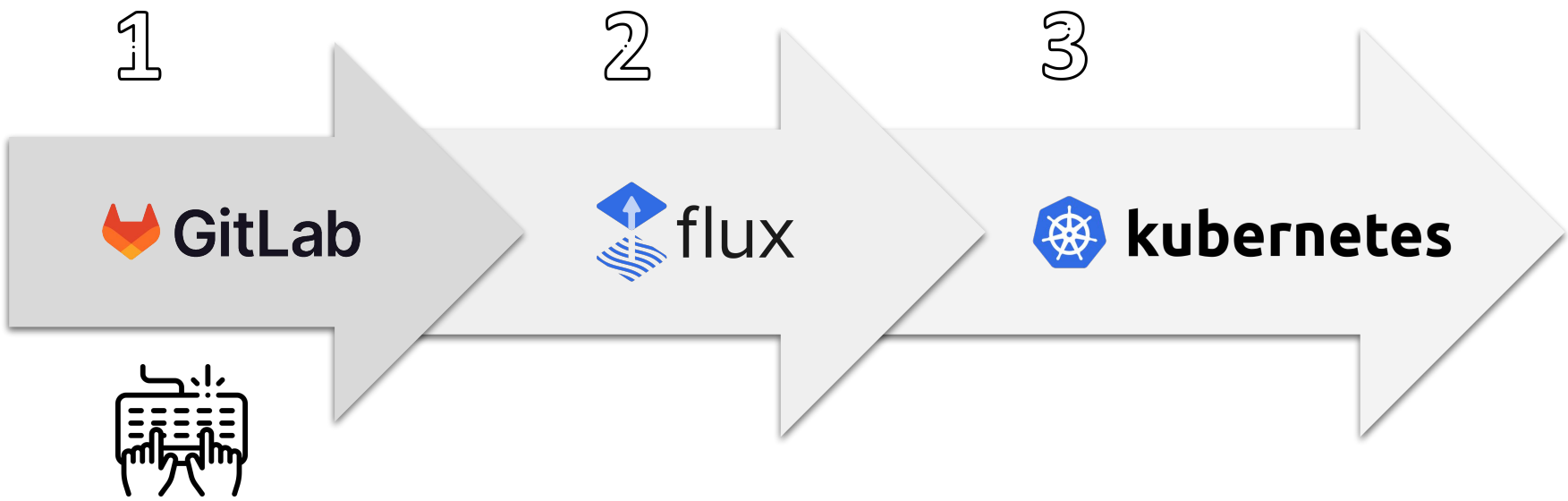
Blue Brain Nexus infrastructure



Changing the infrastructure (before)



Changing the infrastructure



New use cases possible with Kubernetes

Git as source of truth

flux

Automation for infrastructure

HELM

Helm charts

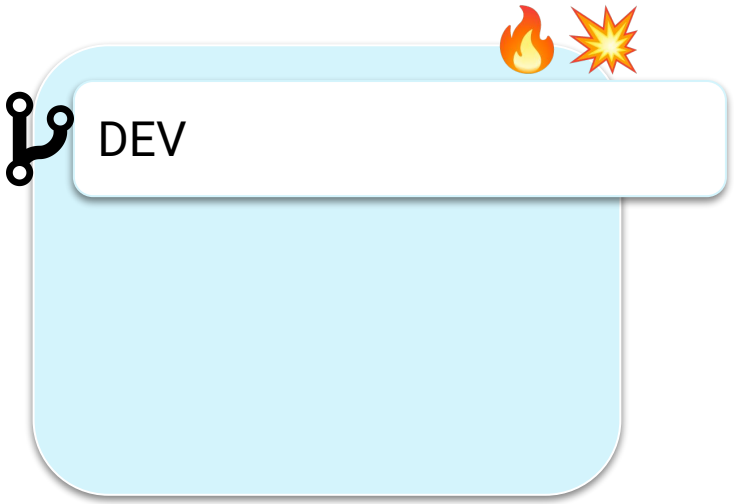
Flexible ingress configuration

mozilla/**sops**

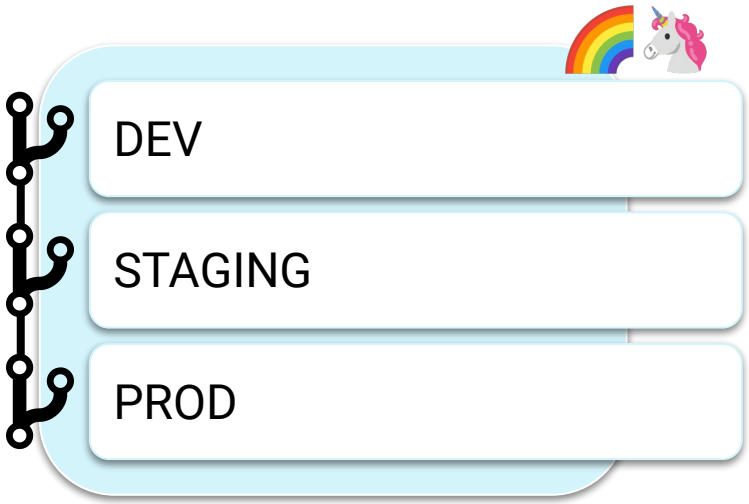
Encrypted secrets stored in Git

Daily operations

Kubernetes dev cluster



Kubernetes production cluster



Example: Porting an application to Kubernetes

Decide how to deploy your application, some alternatives are:

- Find a Helm chart
- Create a Helm chart
- Create the necessary Kubernetes manifests (Deployment, StatefulSet, Service, Ingress…)

Example: Porting an application to Kubernetes

Decide how to deploy your application, some alternatives are:

- Find a Helm chart
- Create a Helm chart
- Create the necessary Kubernetes manifests (Deployment, StatefulSet, Service, Ingress…)

Configure and test the Helm chart or Kubernetes manifests

- Deploy them in your dev cluster or dev environment
- Keep resolving any errors that may appear when deploying a Helm chart or manifest
- Ensure that the storage and network configuration works, add test data if needed

Example: Porting an application to Kubernetes

1

Decide how to deploy your application, some alternatives are:

- Find a Helm chart
- Create a Helm chart
- Create the necessary Kubernetes manifests (Deployment, StatefulSet, Service, Ingress…)

2

Configure and test the Helm chart or Kubernetes manifests

- Deploy them in your dev cluster or dev environment
- Keep resolving any errors that may appear when deploying a Helm chart or manifest
- Ensure that the storage and network configuration works, add test data if needed

3

Apply your working configuration to other environments

- Storing your manifests and charts in git makes it easier
- Generate new secrets for the new instances of your applications
- Using Helm charts and Flux makes it easy to scale your application to new environments

The migration process for a user

Learn about Kubernetes

- Learn about Kubernetes and its architecture
- Learn about Kubernetes resource types: RBAC, Network policies…

The migration process for a user

Learn about Kubernetes

- Learn about Kubernetes and its architecture
- Learn about Kubernetes resource types: RBAC, Network policies…

Testing and development

- Find or create a Helm chart or manifests for every application you need
- Deploy and test every application
- Decide how to use your available namespaces and Kubernetes clusters

The migration process for a user

1

Learn about Kubernetes

- Learn about Kubernetes and its architecture
- Learn about Kubernetes resource types: RBAC, Network policies…

2

Testing and development

- Find or create a Helm chart or manifests for every application you need
- Deploy and test every application
- Decide how to use your available namespaces and Kubernetes clusters

3

Plan, test and execute data migration

- Back up your applications
- Migrate existing data to the new environments – this may require downtime
- Keep a backup of old data in case it's needed

The migration process for a user

1

Learn about Kubernetes

- Learn about Kubernetes and its architecture
- Learn about Kubernetes resource types: RBAC, Network policies…

2

Testing and development

- Find or create a Helm chart or manifests for every application you need
- Deploy and test every application
- Decide how to use your available namespaces and Kubernetes clusters

3

Plan, test and execute data migration

- Back up your applications
- Migrate existing data to the new environments – this may require downtime
- Keep a backup of old data in case it's needed

4

Switch to the new environments

- Verify that the new deployment is working with the migrated data
- Notify users in case of downtime
- Route traffic to the new deployment

Caveats throughout the process

- A lot of time spent on prototyping and testing
- A Kubernetes version has ~1 year of support
- Make sure to consider and test existing use cases – talk to users
- Allocate time to train your team(s)
- Ensure that it is easy to roll back changes if needed – use GitOps
- Network policies, RBAC and Ingress configurations may need adjustment
- Beware of internal networks running behind a proxy – $HTTP_PROXY
- Helm chart quality varies a lot
- Set up monitoring, alerts and dashboards to keep an eye on the applications

Summary

- Start prototyping as early as possible
- Set up a collaboration between the users and operations teams
- Evaluate decisions against the needs of your team or organization
  - Kubernetes clusters vs. namespaces
  - Which CNI (Container Network Interface)?
  - What permissions do users have in the cluster?
  - …
- Use standard alternatives unless you have a reason not to
- Ask for help in the Kubernetes and CNCF Slack instances
- Benefit from the automation possibilities that the Kubernetes ecosystem offers
- Become a Kubernaut and present what you did at KubeCon + CloudNativeCon!

EPFL

Blue Brain Project

The EPFL Blue Brain Project is recruiting – several positions available

For more information and how to apply, visit
go.epfl.ch/bluebrain-careers