Introduction to the new generation EGI container execution platform

Adrián Rošinec (adrian@ics.muni.cz)
Cloud Engineer at CESNET/Masaryk University

24/01/2024
EGI Webinar 2024

What to expect

- **Introduction to the EGI Container platform**
  - **New features making things easier**
- Containers 101
  - Building, storing and running containers
- Kubernetes 101
  - Basic principles, running containers is k8s
- Short demo
  - Build container and deploy the application to **EGI Cloud Container Compute**
- Discussion

EGI Cloud Container platform

- Managed environment to execute and store containers
- Built on Kubernetes (compute) and Harbor (container registry)
  - is provided by the team from the Czech e-infrastructure
    Lukáš Hejtmánek, Adrián Rošinec, Viktoria Spišáková, Klára Moravcová and Kristián Kováč
- Users are provided only with project and quota on resources
- Ready to be integrated with additional resource providers
- Integrated with the **EGI Check-in** for seamless login experience
- Access to the platform
  - "Free tier" via **vo.access.egi.eu**, few resources, "cpu-time" limited to the 3 months
  - Guaranteed resources - reach out to get SLA, see open calls on EGI website

New Generation – New Features

- Fully managed Kubernetes service
  - Users will obtain project and quota
- Web GUI to manage Kubernetes project – Rancher
  - Simplify container operations
  - https://rancher.cloud.e-infra.cz
- Catalog with prepared applications
  - MinIO, Virtual Desktop (VNC/WebRTC), Rstudio, Matlab, Scipion, ...
  - PostgreSQL operator (for HA databases)
- Dynamic DNS in ***.dyn.cloud.e-infra.cz** domain
- Let's Encypt certificates + auto-renewal
- Load Balancer (to route traffic from the Internet)
- Persistent storage (NFS)

What is available?

Cluster size

- 3456 CPU
- 19.5TB RAM
- GPU Acceletators
  - 22 NVIDIA A40
  - 6 NVIDIA A10
  - 12 NVIDIA A100 (80GB variant)
- 500 TB all-flash of persistent storage

Broader context

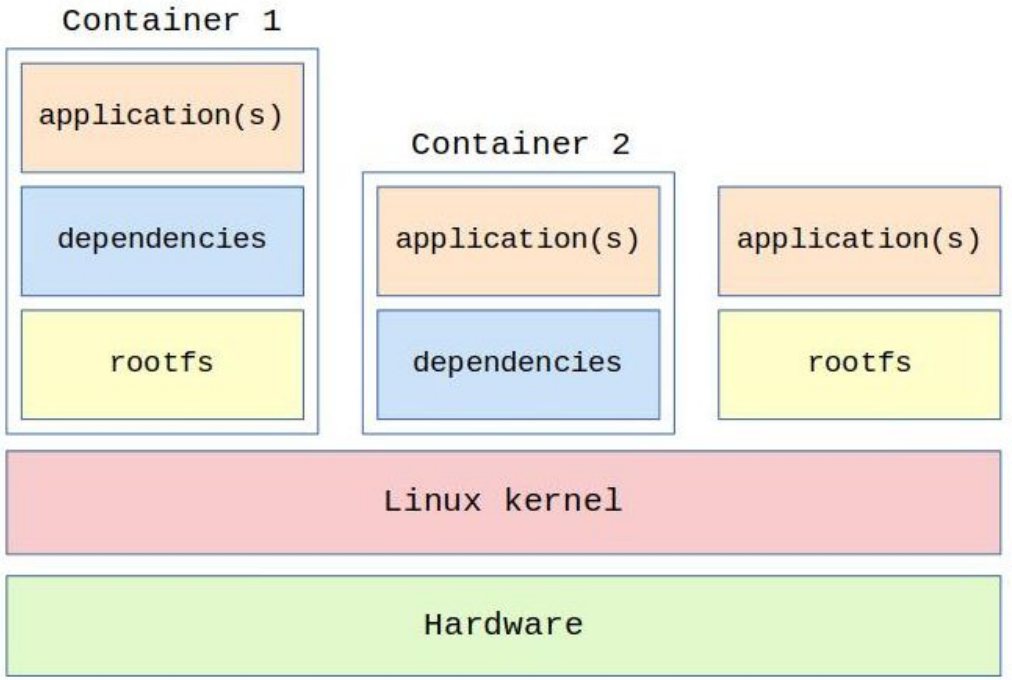
Comparison to other services

	VMWare	OpenStack	Slurm/OpenPBS	Kubernetes
Cloud type	Virtualization	IaaS	Batch system	CaaS (almostPaaS)
Basic Object	Virtual machine	Virtual machine	Job (script, software, …)	Container
Container workload	Within Docker/…	Within Docker/…	Singularity	Native entity
HPC / GPU – native support	Not usual	Yes	Yes	Yes
Reliability, high-availability	Yes, live migrations	Yes, user's responsibility	N/A	Yes, scaling and life-cycle on k8s
User's interface	GUI, API from CLI	GUI, API, CLI	CLI	GUI, API, CLI
Ease of scaling up the application	manual, deploy more VMs (API)	manual, deploy more VMs (API)	manual, amount of jobs	Semi-automatic, specify number of replicas

- Introduction to the EGI Container platform
  - **New features making things easier**
- **Containers 101**
  - **Building, storing and running containers**
- Kubernetes 101
  - Basic principles, running containers is k8s
- Short demo
  - Build container and deploy the application to **EGI Cloud Container Compute**
- Discussion

Containers 101

- Containers – very popular way to run application
- Minimal filesystem with required application files and dependencies-libs
- Benefits
  - Helps with software distribution
  - Isolation from underlying system = kernel is shared
  - Lightweight - not as heavy as virtual machine
  - Users doesn't have to manage any missing dependencies
  - Isolation of data and application
- Docker
  - the most popular management tool for containers
  - "Docker container" could be run on local PC, on VM in cloud or container platform as Kubernetes



Source: https://sergioprado.blog/introduction-linux-containers/

Building containers

- Using dockerfile
  - Directives like:
  - COPY, RUN, WORKDIR, USER, CMD
- Manual `docker build`
- Automatically
  - via CI - Github Actions or Gitlab Pipelines
  - New container image is built on code push
- Several important steps
  - Prepraing the software dependencies
  - Own software
  - User
- Result is ideally non-root container

```
# Use an official Python runtime as a base image
FROM python:3.9

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy the requirements file into the container at /usr/src/app
COPY requirements.txt .

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Copy the current directory contents into the container at /usr/src/ap
COPY . .

# Specify the command to run on container start
CMD ["python", "./your_script.py"]
```

See https://docs.cerit.io/docs/dockerfile.html

Storing containers

- Container images are stored in **container registries**
- Types
  - **Local** - on your own computer
    - $ docker images
  - **Remote** - service provided by e-infrastructure / private company
    - Public - access to the registry is open for read, closed for write - hub.docker.com, github.com, quay.io, ...
    - Private - provided by e-science center such as EGI – **cerit.io**

See https://docs.cerit.io/docs/harbor.html

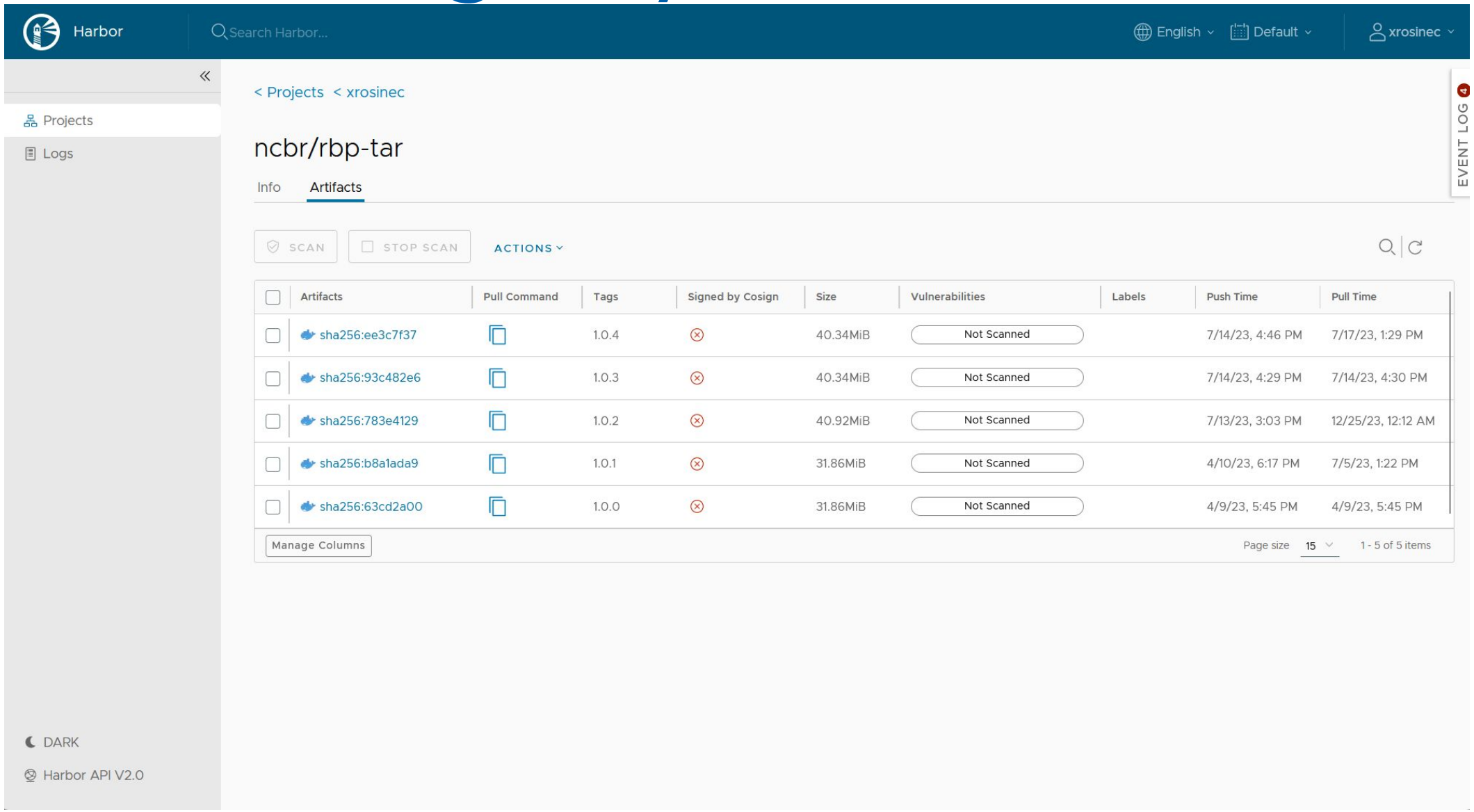
EGI Container Registry

New service Harbor registry

- Own registry, integrated with EGI Cloud Container platform
- For now, located at https://cerit.io
- Access and project granted with container platform project on request
- Image URL: cerit.io/xrosinec/gromacs
- Uploading an image:
  - $ docker login –u xrosinec cerit.io
  - $ docker tag gromacs:adrian-patch cerit.io/xrosinec/gromacs:adrian-patch
  - $ docker push cerit.io/xrosinec/gromacs:adrian-patch

See https://docs.cerit.io/docs/harbor.html

Harbor registry



Running containers

- Locally using command `docker run`
  - ```
    docker run -it \
    -v /home/adrian/gromacs/sample-experiment:/home/user/experiment \
    cerit.io/gromacs \
    /bin/bash
    ```
- Remote
  - Within the container platform
  - In Kubernetes using manifests or kubectl (cli)
    ```

Few tips…

- If non-root it is not possible to install additional software
  - Container has to be prepared with all required sw and libs in build phase
  - Installations only to home (software needs to be prepared for that)
- Writing is generally possible to /tmp and /home dirs.
  - If not specified otherwise in build phase via chown and chmod
- Data modified within the container aren't persistent = can't modify image
  - Need to attach external storage where changes would be persistent

See https://docs.cerit.io/docs/dockerfile.html

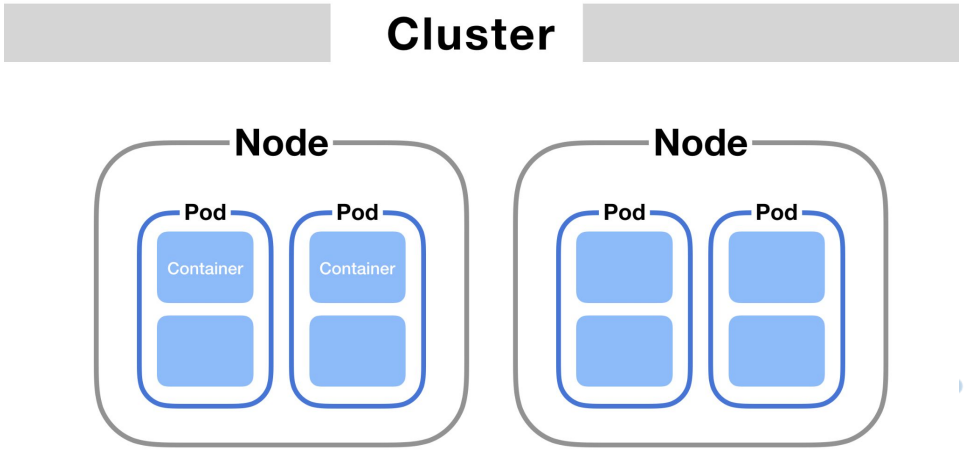
- Introduction to the EGI Container platform
  - **New features making things easier**
- Containers 101
  - Building, storing and running containers
- Kubernetes 101
  - Basic principles, running containers is k8s
- Short demo
  - Build container and deploy the application to **EGI Cloud Container Compute**
- Discussion

Kubernetes 101

- Orchestrator - basically docker on steroids
  - Tool for management of running containers
  - Developed by Google
- What it does
  - Download of container image
  - Run of container and management of container's state
  - Makes sure that container is running and replicated (if needed)
  - Networking
  - Access to the network storage
- What it is not capable of
  - Can't handle order of running containers
    - Only simple pre-start e.g. to initialize state of the application on first run
  - For dependent jobs need to use additional workflow manager such as NextFlow or SnakeMake

Kubernetes principles

- Kubernetes are organized to clusters
  - control plane and worker nodes
- Kubernetes cluster
  - has physical servers
  - has logical namespaces - user space with quotas, equivalent to the project or unix group
- Notable objects
  - Pod - the smallest deployable units of computing in k8s
  - Deployment - most common way to get your app on k8s
  - PersistentVolumeClaim – persistent storage for container
  - Ingress – get traffic from the Internet
- HTTP API
  - And clients on top of API: kubectl or Rancher, ...



Kubernetes actions

- Running workload
  - Using **manifest file** with the specification of:
    - what to run – pointer to container image
    - what context – command, environmental variables, mapped storage
  - The job/deployment doesn't have explicit wall-time
- Monitoring of the workload
  - kubectl logs – show stdout of running container
  - kubectl top – show current usage of CPU/MEM
- Modify state of container
  - Kubectl cp
- Management of the life-cycle
  - Scaling up/down, restarting, enforcing container's state

Simple Manifest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-kubernetes
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-kubernetes
  template:
    metadata:
      labels:
        app: hello-kubernetes
    spec:
      securityContext:
        runAsUser: 1000
        runAsNonRoot: true
        seccompProfile:
          type: RuntimeDefault
      containers:
      - name: hello-kubernetes
        image: paulbouwer/hello-kubernetes:1.9
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
        ports:
        - containerPort: 8080
```

Kubernetes storage

- Several options where to store data
- /tmp — implicit, no need to specify in manifest – content is deleted with restart
  - Can't be shared between more running containers
- emptyDir — need to specify in manifest, content is ephemeral
  - In physical machine memory or on local, typically faster disks
  - emptyDir can be shared between containers within the one manifest
- PVC — need to be specified in manifest, persistent
  - Typically, as network storage e.g. NFS, CIFS, S3, … (automatically mounted)
  - PVC's could be shared between containers

See https://docs.cerit.io/docs/pvc.html

Kubernetes PVC

- Persistent Volume Claim
  - way to get persistent storage via Storage Class
  - Types
    - ReadWriteOnce - can only mount to one Pod
    - ReadWriteMany - can be mount to many Pods
- Storage Class – represents the storage type (NFS, Block storage,...)

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my-storage-class
  resources:
    requests:
      storage: 1Gi
```

```yaml
  spec:
    containers:
    - name: my-container
      image: nginx:latest
      volumeMounts:
      - name: my-persistent-storage
        mountPath: /app/data
volumes:
- name: my-persistent-storage
  persistentVolumeClaim:
    claimName: my-pvc
```

See https://docs.cerit.io/docs/pvc.html

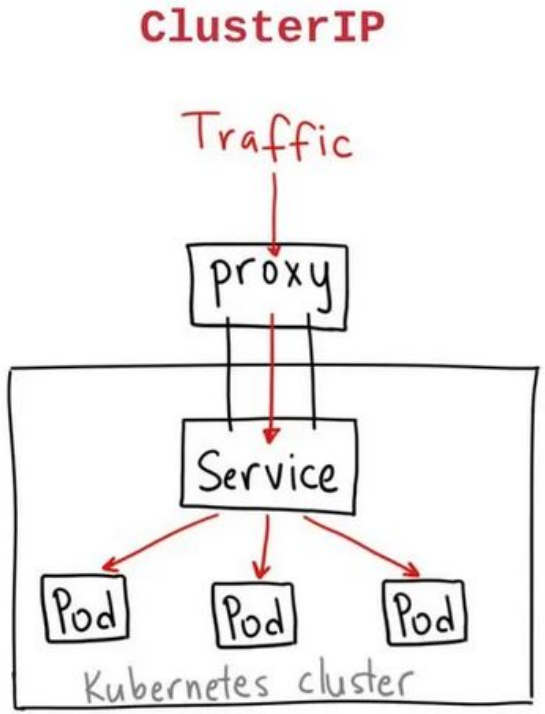
Storage Classes Available

- Local flash
  - /tmp or emptyDir
- nfs-csi
  - 500 TB all flash network storage
- sshfs
- webdav
- onedata – could be used with EGI DataHub
  - Mounts Onedata Space/Dataset into the container

See https://docs.cerit.io/docs/pvc.html

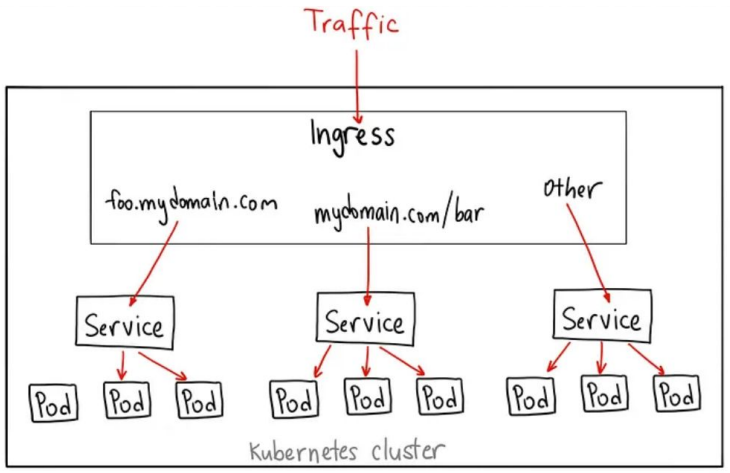
Kubernetes networking

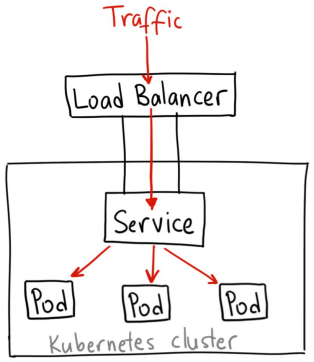
- Each pod has non-static IP address
- To communicate with deployment which consists multiple pods we use Services with unique static IP address



Accessing from the Internet

- **Ingress**
  - exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
  - can be configured with externally-reachable URLs
- LoadBalancer
  - For other types of services
  - each loadbalancer has its own IP address






```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-service
            port:
              number: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalancer-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

See https://docs.cerit.io/docs/kubectl-expose.html

Exposing applications

- LoadBalancer
  - kubernetes.io/ingress.class: "nginx“
- Certificate manager
  - cert-manager.io/cluster-issuer: "letsencrypt-prod"

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: egi-webinar-application-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    kubernetes.io/tls-acme: "true"
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  tls:
    - hosts:
        - "egi-webinar.xrosinec.dyn.cloud.e-infra.cz"
      secretName: egi-webinar-xrosinec-dyn-clout-e-infra-cz-tls
  rules:
  - host: "egi-webinar.xrosinec.dyn.cloud.e-infra.cz"
    http:
      paths:
      - backend:
          service:
            name: egi-webinar-application-service
            port:
              number: 8080
        pathType: ImplementationSpecific
```

Examples

- Scientific applications
  - https://rbp-tar.ncbr.dyn.cloud.e-infra.cz/
  - https://omero-test.dyn.cloud.e-infra.cz
- Fully scalable Jupyter Hub
  - https://hub.cloud.e-infra.cz/
- AlphaFind
  - https://alphafind.dyn.cloud.e-infra.cz/search
- AlphaFold as a service
  - https://alphafold.cloud.e-infra.cz
- Workflow managers – Nextflow pipelines
  - https://docs.cerit.io/docs/nextflow.html
  - “cloud bursting principles”
- Services
  - Indico, Limesurvey, Outline (Knowledge management), Alternative to Doodle