

Calypso architecture

Physical Infrastructure

It is currently composed of the following machines :

- 1 DELL R740XD (Master)
- 15 DELL R630 (6 currently active)
- 3 DELL R630 (1 spare in 23N321, storage 1)

Network

Calypso is located in an isolated separate network inside the school. It can be accessed via a [Wireguard VPN](#) (ask Rémi for access).

Inside this network you have a simple setup :

```
192.168.88.0/24 : Network appliances subnet
192.168.89.0/24 : VPN subnet
192.168.90.0/24 : Server's IDRACs subnet
192.168.91.0/24 : Servers subnet
```

Currently there are 6 servers running in the cluster that are accessible to students for their labs :

```
calypso0 : 192.168.91.10
calypso1 : 192.168.91.11
calypso2 : 192.168.91.12
calypso3 : 192.168.91.13
calypso4 : 192.168.91.14
calypso5 : 192.168.91.15
calypso6 : 192.168.91.16
```

The Calypsomaster node is not available for connection to students, it contains the Kubernetes control plane :

```
calypsomaster : 192.168.88.248
calypsomaster IDRAC : 192.168.88.249
```

To work on it, see with your teacher which one are allocated to you, or if you need to run jobs on all nodes (via SLURM) you can pick any of them to run.

NAS NFS share

On the Calypso infrastructure, there is a shared storage mainly for SLURM needs : it gives students a shared storage between all Calypso nodes to run jobs on any node and have a common filesystem to

run jobs / get results on

```
nas (NAS appliance) : 192.168.88.250
```

The filesystem is mounted from **nas:/volume1/calypso_homes/homes/firstname.lastname** , to each student's shared directory **/exports/firstname.lastname/**

On each student's home, there is a symlink to it : **~/nas_home**

DNS server / Gateway

In this isolated network, the Sinf provides us only their gateway as the only DNS server :

```
DNS / Gateway : 172.30.7.1
```

Software Architecture

User accounts

User access is SSH based for now, managed by Rémi.

Container Runtimes

To run containers on Calypso, you can use :

- [Apptainer](#)
- [Docker](#)
- [Containerd](#) (in [Kubernetes](#))

SLURM Cluster

There is a [SLURM](#) cluster on all Calypso worker nodes :

```
calypsomaster : no SLURM
calypso0 : SLURM controller + accounting DB
calypso[1-6] : SLURM workers
```

Configuration

TODO : redeploy from ISC compute center configuration

Kubernetes Cluster

The Kubernetes control plane is on calypsomaster, you don't have access to it, and this node can't run pods, it's just administrative for the operation of the cluster. The other nodes are calypso0 to 5, they are all capable of running pods.

```
# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
calypso0	Ready	<none>	221d	v1.31.1
calypso1	Ready	<none>	221d	v1.31.1
calypso2	Ready	<none>	221d	v1.31.1
calypso3	Ready	<none>	221d	v1.31.1
calypso4	Ready	<none>	221d	v1.31.1
calypso5	Ready	<none>	221d	v1.31.1
calypso6	Ready	<none>	18d	v1.31.1
calypsomaster	Ready	control-plane	221d	v1.31.1

There are 2 usable namespaces as of now : the default one where teachers can do tests, and the isc3 namespace for students, the others are administrative.

```
kubectl get namespaces
```

NAME	STATUS
default	Active
isc3	Active
kube-flannel	Active
kube-node-lease	Active
kube-public	Active
kube-system	Active

The K8s cluster was created with containerd as the main container engine: there is also a docker-engine + docker-compose that are installed on the servers for testing.

Users

Teacher users are administrators, they have some clusterroles permissions (cluster-admin-role), and large roles permissions on default and isc3 namespaces (default-admin-role, isc-admin-role).

Students have only a role permission, named student-role to do all needed operations on pods, services, deployments etc...

Storage

The PersistentVolumes (PV) have already been created locally to reflect the local storage capacity, which is NVMe SSD storage: the trade-off is that the PersistentVolumeClaims (PVC) being only assignable 1:1 to a PV, you can't create additional PVCs, it will never be bound to a PV since they are already taken statically: this is due to the local volume type which requires the binding to use the WaitForFirstConsumer policy.

```
kubectl get storageclasses
```

NAME	PROVISIONER	RECLAIMPOLICY
VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	
local-node-storage	kubernetes.io/no-provisioner	Delete
WaitForFirstConsumer	false	


```
kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY
STATUS CLAIM	STORAGECLASS	VOLUMEATTRIBUTESCLASS	
large-local-calypso0-volume	1500Gi	RW0,RWX	Retain
Bound default/claim1500g	local-node-storage	<unset>	
local-node-volume	300G	RW0,RWX	Retain
Bound default/claim300g	local-node-storage	<unset>	

The PVCs were created with Access Mode ReadWriteOnce and ReadWriteMany: which allows pods created by several people to be able to reuse the same PV in a concurrent way.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
STORAGECLASS		VOLUMEATTRIBUTESCLASS		
claim1500g	Bound	large-local-calypso0-volume	1500Gi	RW0,RWX
local-node-storage		<unset>		
claim300g	Bound	local-node-volume	300G	RW0,RWX
local-node-storage		<unset>		

Rules for persistent storage:

- You should be aware that calypso0 is the only one to have a 1.5T SSD, and on all the other nodes there is 300G of space allocated to K8s. So to create a pod requiring for example a non-ephemeral database, you need to specify either the PersistentVolumeClaim claim300g or the claim1500g in the deployment yaml, and it will be dispatched to a corresponding node.
- Unless you need to test a large volume of data, please use the claim300g as a priority
- The pods will automatically launch on the node that has the right PV/PVC pair.

```
kubectl get pods -n isc3 -o=custom-  
columns=NAME:.metadata.name,STATUS:.status.phase,NODE:.spec.nodeName
```

NAME	STATUS	NODE
bigwww	Running	calypso0
bigwww2	Running	calypso0
www	Running	calypso4
www2	Running	calypso3

Ressources GPU

Each node in the cluster has Nvidia Container Toolkit installed, which allows you to use their Nvidia Tesla T4 GPU from the containers.

Registry Docker

To avoid saturating the network link with the builds there is a local registry that can be used on calypsomaster: you already have in your configuration daemon.json of each node a local insecure registry of settings: "insecure-registries": ["192.168.88.248:5000"]

TODO : Recreate the registry with a certificate from local PKI, distribute CA certificates / registry certificate to every node to trust the local authority.

From:

<https://wiki.isc-vs.ch/> - **The ISC wiki**

Permanent link:

<https://wiki.isc-vs.ch/doku.php?id=infra:calypso&rev=1748615544>

Last update: **2025/05/30 14:32**

