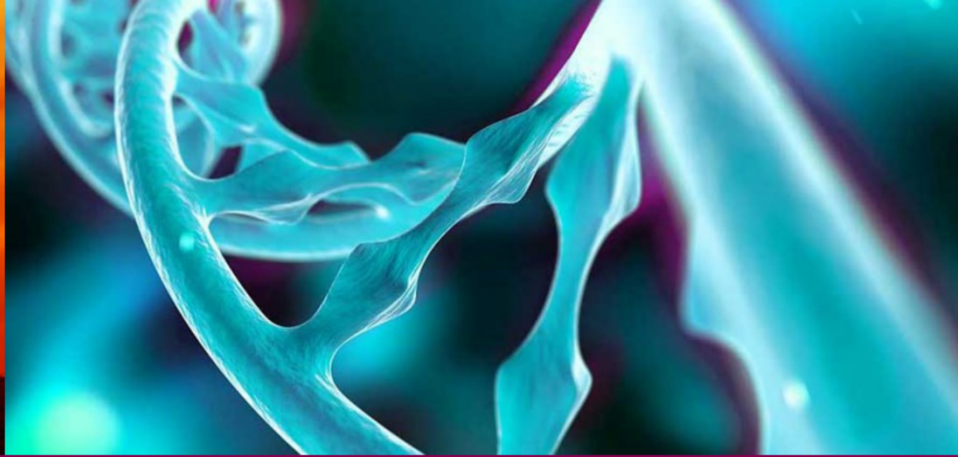


Container clouds fundamentals

(with Rahti OpenShift OKD)

Tristan Perard, Cloud System Specialist

Jemal Tahir, Cloud System Specialist

Joona Tolonen, Cloud System Specialist

Álvaro González, Cloud System Specialist

CSC – Finnish expertise in ICT for research, education and public administration

Notice

We will record this presentation

- This is to explore the idea of publishing an online video of this course
- We will cut out from the recording the Q&A sections (for GDPR and privacy reasons).
- So feel free to ask questions any time

If something **makes no sense**, you want to make a **question** or **correction**, **Please interrupt** and make your comment

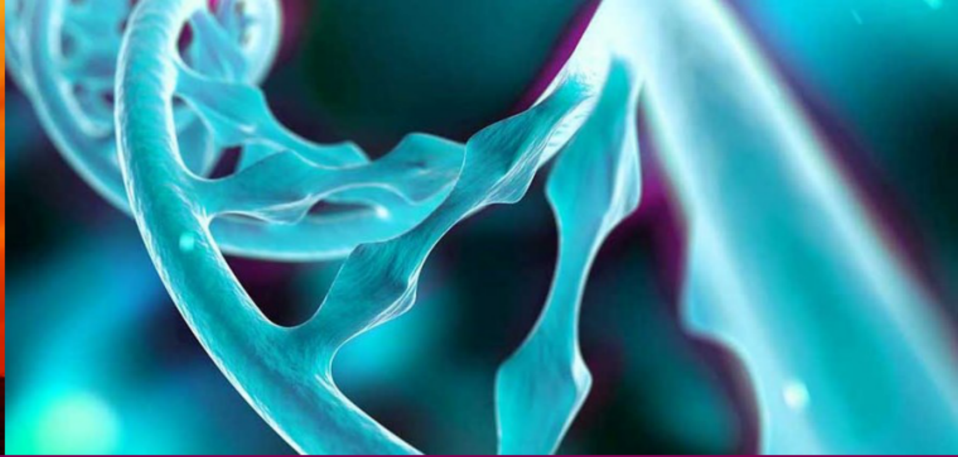
Expectations

- How familiar are you with Containers and Rahti Cloud?
- What are you expecting to learn from this course?

https://www.menti.com/alt4w9vdjifn

Schedule

When	What	
9:00 - 10:30	Lecture	What is Rahti? Introduction to containers
		Application templates
		Web interface Howtos
10:30 - 10:45	Coffee break	☕
10:45 - 12:00	Exercises	A
12:00 - 13:00	Lunch break	🍽️
13:00 - 14:30	Lecture	Storage
		High level Kubernetes architecture
		Command line tool
		Command line interface Howtos
14:30 - 14:45	Coffee break	☕
14:45 - 15:00	Exercises	B and C
15:00 - 15:15	Closing	Documentation and contact info
15:15 - 16:00	Exercises	Extra time



What is Rahti?

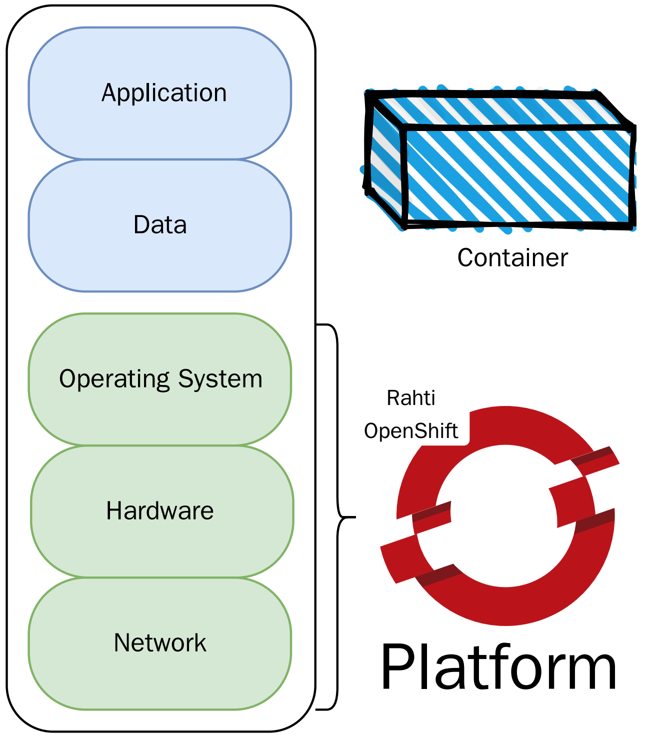
PaaS cloud

Rahti PaaS cloud

- Rahti (https://rahti.csc.fi) is a platform as a service (**PaaS**) container cloud orchestration service.

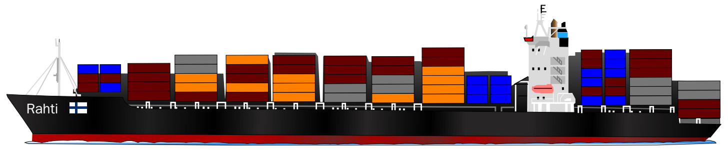
"The infrastructure (network, hardware, Operating System, ...) is offered as a **platform** to you, the user, so you can just worry about running the Software and nothing else".

- No worries about: Hardware issues, Operation systems patches, etc.
- Security: Containers allow software from independent teams of people to run isolated, even though they run in the same hardware.
- QoS: Orchestration services provide assured resources
- Based in OpenShift OKD (by RedHat)
  - Extends the functionality of Kubernetes.



Rahti advantages

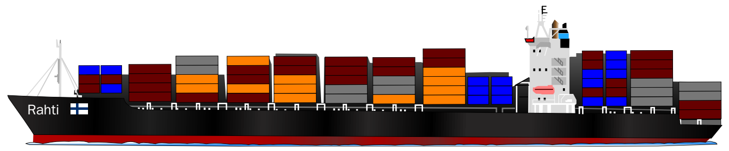
- Out of the box:
  - **health monitoring**, resource consumption, and liveness and readiness probes.
  - **scaling**, resources can be configured to scale up or down responding to load. (faster than VMs)
  - **failover**, in case of any failure, like hardware failure, the software will be restarted.
  - **rolling updates**, a new version of an application will be deployed with no downtime.
  - **load balancing**, automatically distributes load among resources.
  - **DNS**, no need to make any support request or wait*.
  - **certificates**, always valid, automatically renewed*.



*For a given pattern of URLs. `something.rahtiapp.fi`

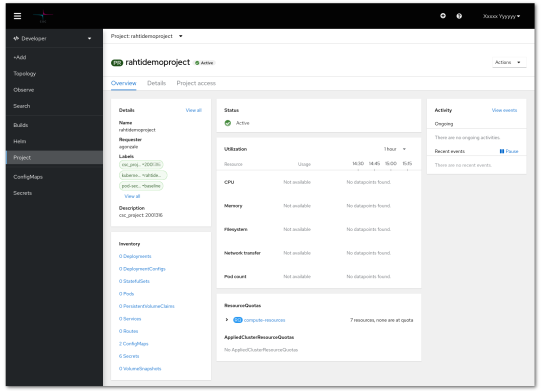
More Rahti advantages

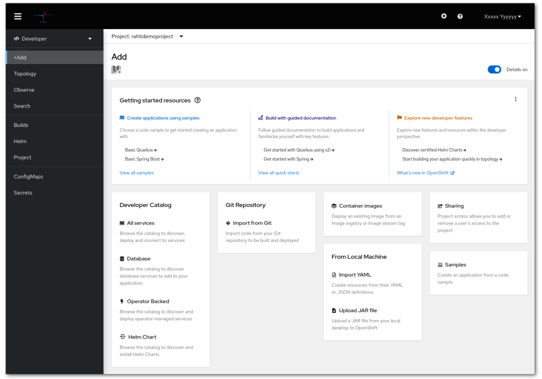
- Simple code deploying:
    - Source code. Rahti provides tools to build and deploy code automatically. **Source2Image (S2I)**.
    - Internal Rahti **template catalog** and **Helm** charts.
- Support in the **web interface**:
    - Launch applications
    - Tune application parameters
    - Request storage
    - Debug and monitor applications
    - Check logs
- Also powerful **CLI** and **library** interfaces.



Interacting with Rahti control plane

Web console





Command line

```
oc create -f pod.yaml
oc replace --force pod.yaml
oc apply -f *.yaml
oc patch ...
oc expose ...
```

Using client library

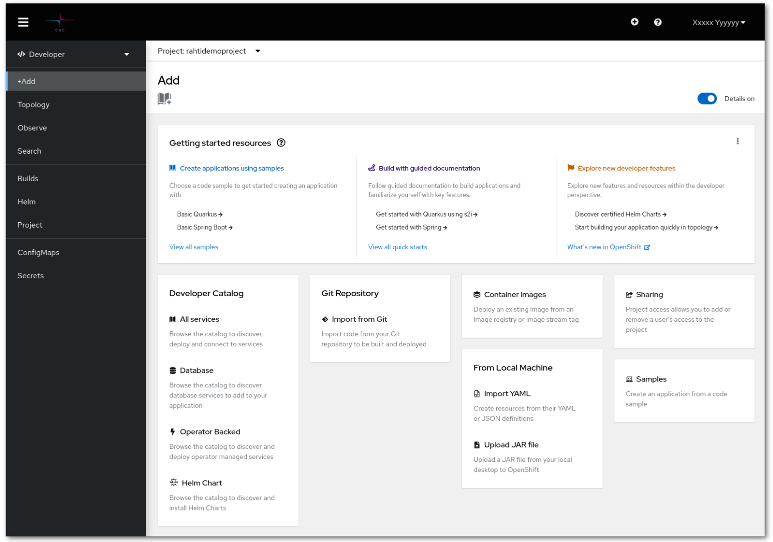
```
from kubernetes import config

c = config.new_client_from_config()
# etc...
```

- Official: Go, Python, Java, dotnet, JavaScript
- Community maintained: Clojure, Go, Java, Lisp, Node.js, Perl, PHP, Python, Ruby, Rust, Scala, dotNet, Elixir, Haskell

Web console:

Service catalog   Developer console   Administrator console

- The default opening viewport
- Create projects
- Launch applications from templates
- Deploy from images



Choose a code sample to get started creating an application with.

Follow guided documentation to build applications and familiarize yourself with key features.

Explore new features and resources within the developer perspective.

Browse the catalog to discover, deploy and connect to services

Browse the catalog to discover database services to add to your application

Browse the catalog to discover and deploy operator managed services

Browse the catalog to discover and install Helm Charts

Import code from your Git repository to be built and deployed

Deploy an existing Image from an Image registry or Image stream tag

Create resources from their YAML or JSON definitions

Upload a JAR file from your local desktop to OpenShift

Project access allows you to add or remove a user's access to the project

Create an application from a code sample

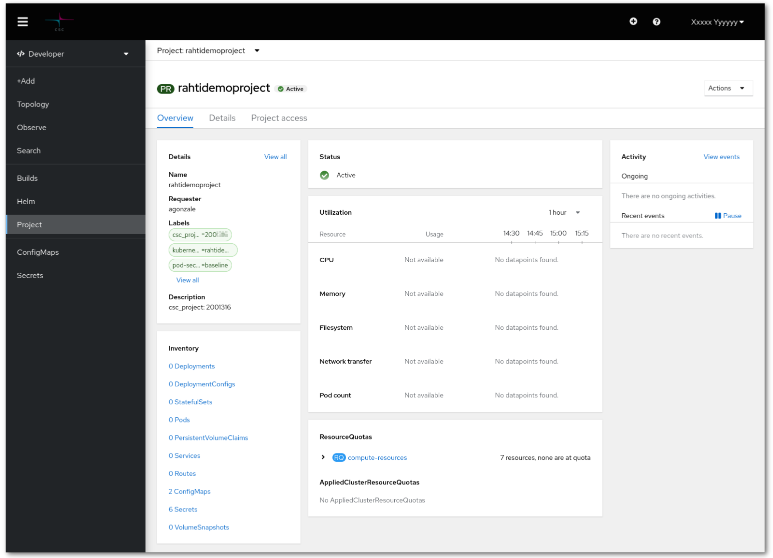
Web console:

Service catalog **Developer console** Administrator console

- Create some API objects
- Deploy images
- Claim storage

- View and modify workloads and API objects

- Monitoring



Project: rahtidemoproject

PR rahtidemoproject Active

Details

Name
rahtidemoproject

Requester
agonzale

Labels
csc_proj... =200[redacted]
kuberne... =rahtide...
pod-sec... =baseline

Description
csc_project: 2001316

Inventory

0 Deployments
0 DeploymentConfigs
0 StatefulSets
0 Pods
0 PersistentVolumeClaims
0 Services
0 Routes
2 ConfigMaps
6 Secrets
0 VolumeSnapshots

Status

Active

Utilization

Resource	Usage	14:30 14:45 15:00 15:15
CPU	Not available	No datapoints found.
Memory	Not available	No datapoints found.
Filesystem	Not available	No datapoints found.
Network transfer	Not available	No datapoints found.
Pod count	Not available	No datapoints found.

ResourceQuotas

RQ compute-resources 7 resources, none are at quota

AppliedClusterResourceQuotas

No AppliedClusterResourceQuotas

Activity

Ongoing

There are no ongoing activities.

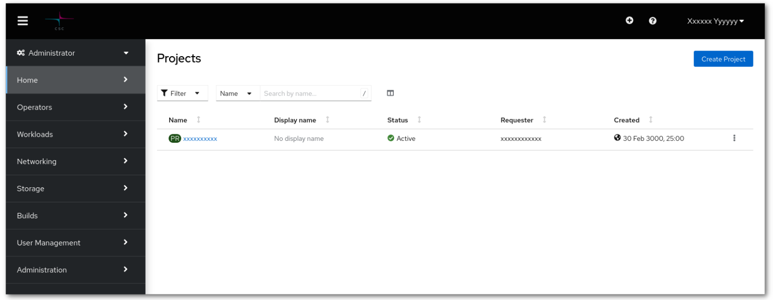
Recent events

There are no recent events.

Web console:

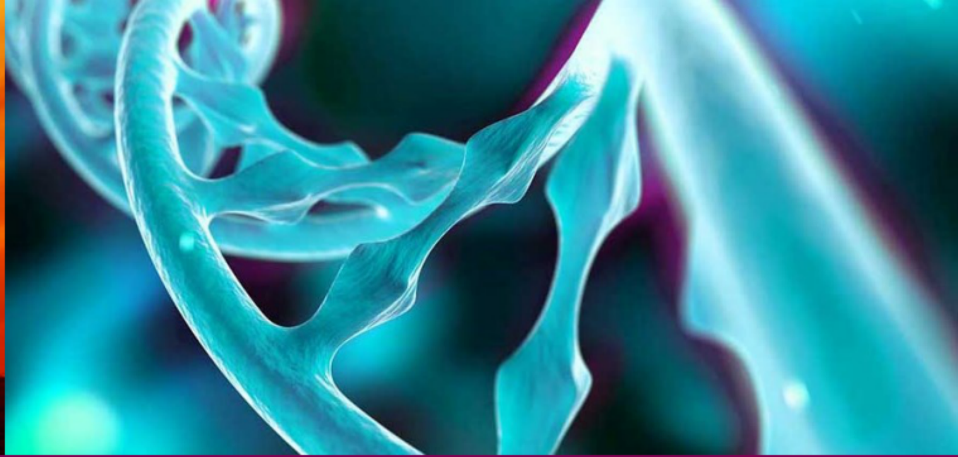
Service catalog   Developer console   **Administrator console**

- Administrator tasks



Projects

Name	Display name	Status	Requester	Created
PR xxxxxxxxxx	No display name	Active	xxxxxxxxxxxx	30 Feb 3000, 25:00



Introduction to containers

CSC – Finnish expertise in ICT for research, education and public administration

Containers (Software vs real life)

Before

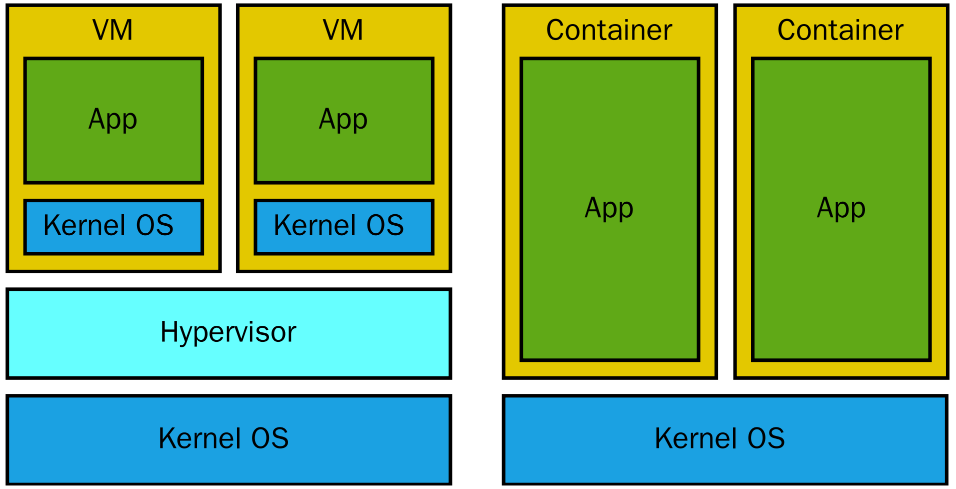
- **Different installation methods**: compile from source, installation wizard, rpm/deb package, etc
- **Libraries dependency** problems: untested, hard to find, outdated, etc
- **No security** isolation
- **No assured** resources

After

- **Standard image registry**, fast and standard deployment.
- **Uniform resource identifier**
  - `name:version`
- **Included library dependencies** in the container
- **Isolated** from the rest of the system
- **Assured resources**

Containers

- All containers running in the same hardware are run by a single operating system kernel and therefore use **fewer resources than virtual machines**.



- Containers are stateless.
  - Any change to a file, done inside a container image will be **lost**.
  - Necessary to use **external volumes** to save data or configuration
- Container images are stored in "container registries"
  - Docker hub is the default registry.
    - https://hub.docker.com
  - Rahti provides a private registry per project.
    - http://image-registry.apps.2.rahti.csc.fi/$PROJECT/$IMAGE

Container Runtimes

Container runtimes are a set of PaaS products that use **OS-level virtualization** to deliver **software in packages** called containers[2], in a user friendly manner.

There are few OCI [3] compatible container runtimes, Docker is currently the most famous, but others also exist:

- CRI-O, "Lightweight Container Runtime for Kubernetes".
- Podman, daemonless container engine that can be run in rootless mode.

Singularity is a non OCI container runtime, mainly used in the HPC world. It is out of scope for this copurse.

[2]: https://en.wikipedia.org/wiki/OS-level_virtualization, [3]: OCI Containers

Container Runtimes II

With a container runtime you usually can:

	Docker	**Podman**
Run	`sudo docker run <image>`	`podman run <image>`
Build	`sudo docker build . --tag <image>`	`podman build . --tag <image>`
Pull (from registry)	`sudo docker pull <image>`	`podman pull <image>`
Push (to registry)	`sudo docker push <image>`	`podman push <image>`
History	`sudo docker history <image>`	`podman history <image>`

They use Linux Kernel features like `cgroups` and `namespaces`.

More info on [how to run containers in Linux]()

Demo I

Docker (cinema)

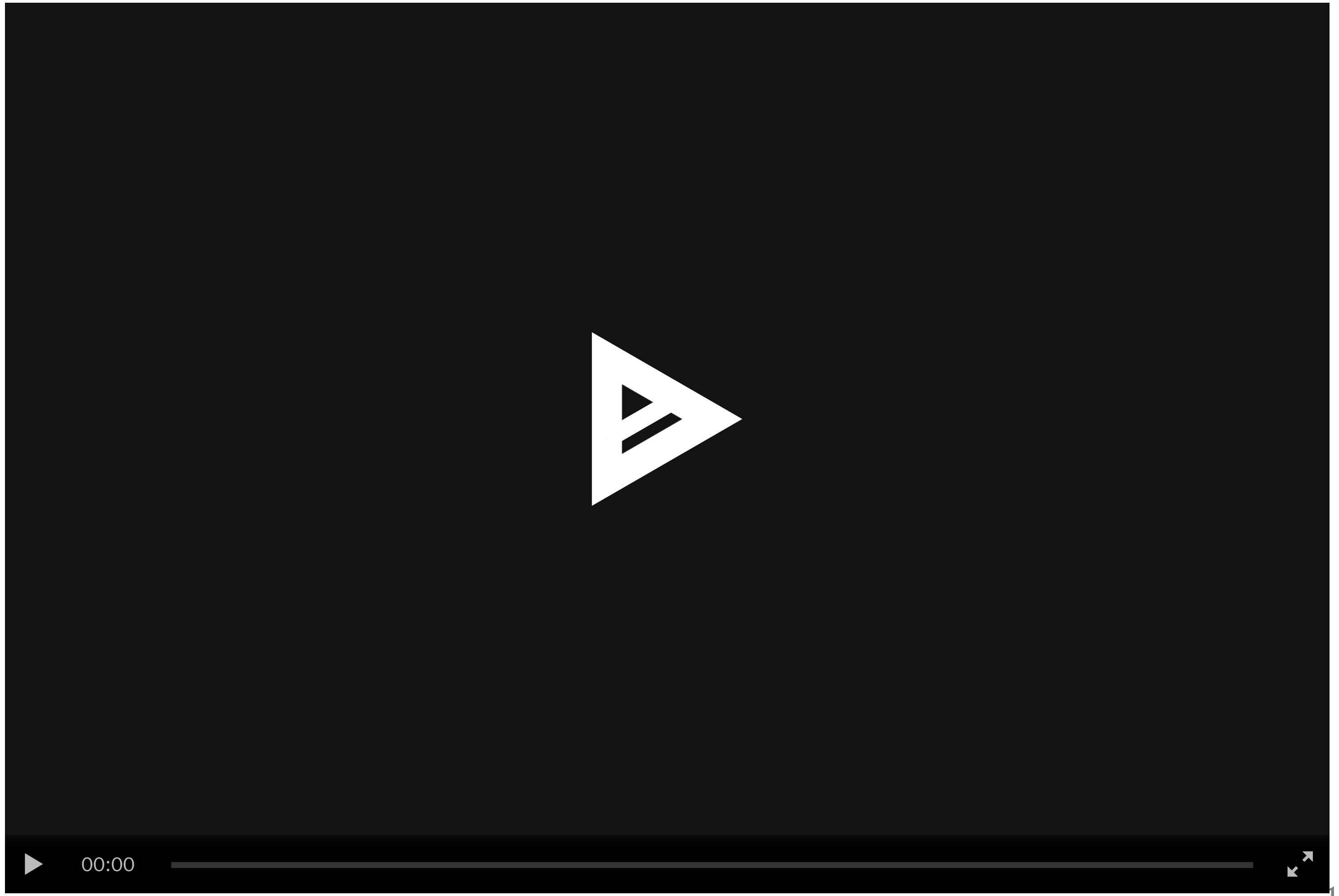
1. Run few command before
2. Run the container `alpine`
3. Repeat the commands inside the container

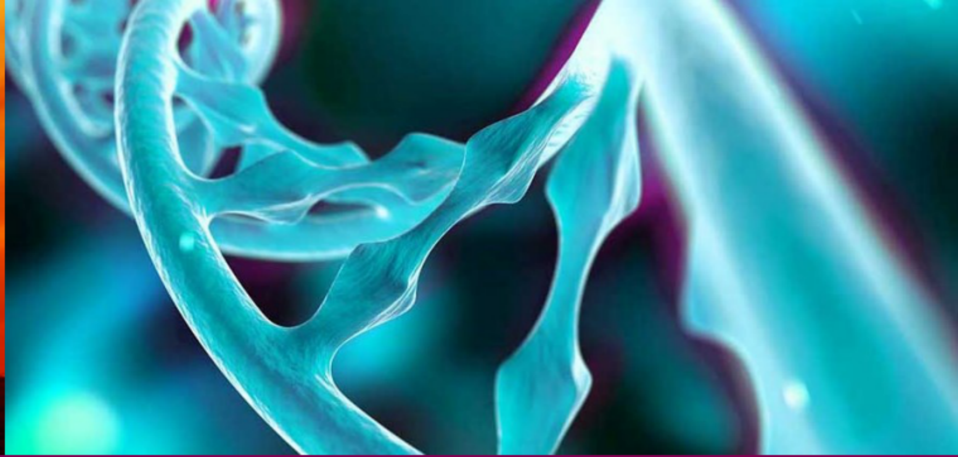
4. Install `python`'s package:

```
python # Not found
apk add python
python
```

5. Exit the container the container, and run it again, python is no longer there.

* player





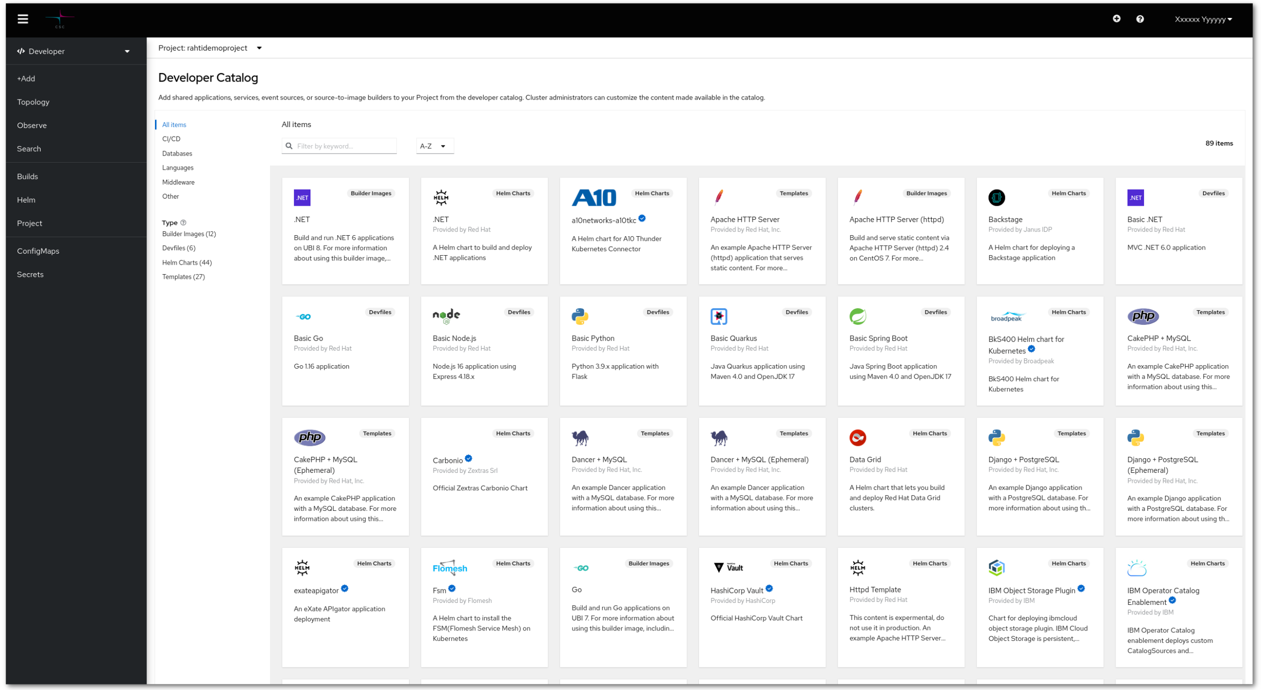
Application Charts

CSC – Finnish expertise in ICT for research, education and public administration

Charts

- Ready to go applications
  - or components of applications (ex: Databases).
- Easy to deploy from the graphical interface:
  - Languages (S2I): Java, Ruby, Python...
  - Databases: MongoDB, MySQL, MariaDB, PostgreSQL...
  - Others: Jenkins

Catalog



Project: rahtidemoproject

Developer Catalog

Add shared applications, services, event sources, or source-to-image builders to your Project from the developer catalog. Cluster administrators can customize the content made available in the catalog.

All items
CI/CD
Databases
Languages
Middleware
Other

Type
Builder Images (12)
Devfiles (6)
Helm Charts (44)
Templates (27)

All items

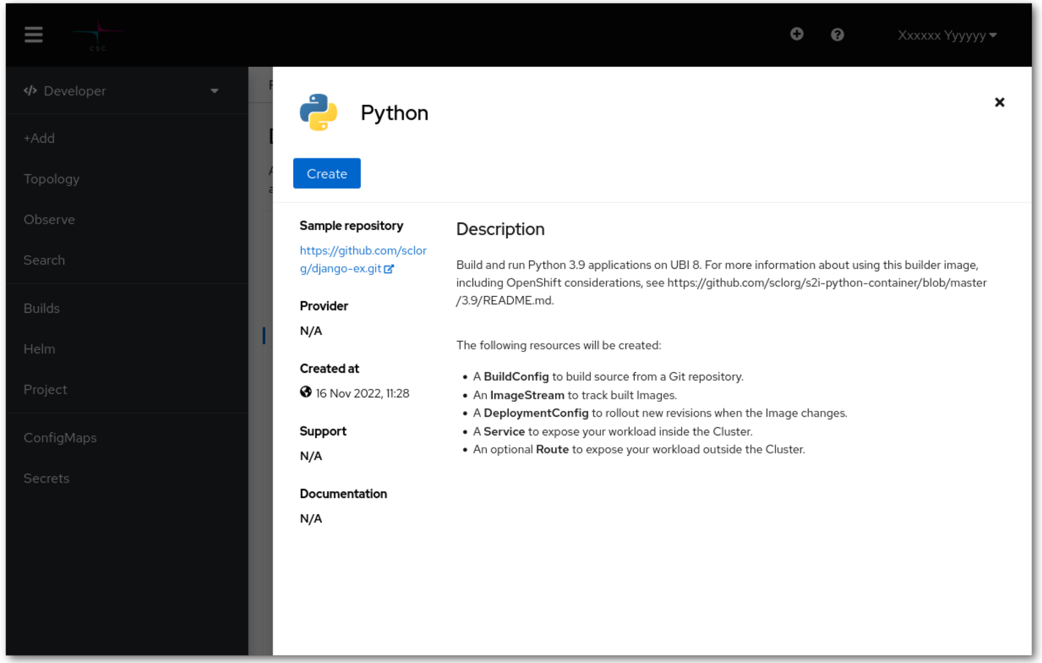
Filter by keyword...

A-Z

89 items

.NET	Builder Images		Build and run .NET 6 applications on UBI 8. For more information about using this builder image,...
.NET	Helm Charts	Provided by Red Hat	A Helm chart to build and deploy .NET applications
a10networks-a10tkc	Helm Charts		A Helm chart for A10 Thunder Kubernetes Connector
Apache HTTP Server	Templates	Provided by Red Hat, Inc.	An example Apache HTTP Server (httpd) application that serves static content. For more...
Apache HTTP Server (httpd)	Builder Images		Build and serve static content via Apache HTTP Server (httpd) 2.4 on CentOS 7. For more...
Backstage	Helm Charts	Provided by Janus IDP	A Helm chart for deploying a Backstage application
Basic .NET	Devfiles	Provided by Red Hat	MVC .NET 6.0 application
Basic Go	Devfiles	Provided by Red Hat	Go 1.16 application
Basic Node.js	Devfiles	Provided by Red Hat	Node.js 16 application using Express 4.18.x
Basic Python	Devfiles	Provided by Red Hat	Python 3.9.x application with Flask
Basic Quarkus	Devfiles	Provided by Red Hat	Java Quarkus application using Maven 4.0 and OpenJDK 17
Basic Spring Boot	Devfiles	Provided by Red Hat	Java Spring Boot application using Maven 4.0 and OpenJDK 17
BkS400 Helm chart for Kubernetes	Helm Charts	Provided by Broadpeak	BkS400 Helm chart for Kubernetes
CakePHP + MySQL	Templates	Provided by Red Hat, Inc.	An example CakePHP application with a MySQL database. For more information about using this...
CakePHP + MySQL (Ephemeral)	Templates	Provided by Red Hat, Inc.	An example CakePHP application with a MySQL database. For more information about using this...
Carbonio	Helm Charts	Provided by Zextras Srl	Official Zextras Carbonio Chart
Dancer + MySQL	Templates	Provided by Red Hat, Inc.	An example Dancer application with a MySQL database. For more information about using this...
Dancer + MySQL (Ephemeral)	Templates	Provided by Red Hat, Inc.	An example Dancer application with a MySQL database. For more information about using this...
Data Grid	Helm Charts	Provided by Red Hat	A Helm chart that lets you build and deploy Red Hat Data Grid clusters.
Django + PostgreSQL	Templates	Provided by Red Hat, Inc.	An example Django application with a PostgreSQL database. For more information about using th...
Django + PostgreSQL (Ephemeral)	Templates	Provided by Red Hat, Inc.	An example Django application with a PostgreSQL database. For more information about using th...
exateapigator	Helm Charts		An eXate APIgator application deployment
Fsm	Helm Charts	Provided by Flomesh	A Helm chart to install the FSM(Flomesh Service Mesh) on Kubernetes
Go	Builder Images		Build and run Go applications on UBI 7. For more information about using this builder image, includin...
HashiCorp Vault	Helm Charts	Provided by HashiCorp	Official HashiCorp Vault Chart
Httpd Template	Helm Charts	Provided by Red Hat	This content is expermental, do not use it in production. An example Apache HTTP Server...
IBM Object Storage Plugin	Helm Charts	Provided by IBM	Chart for deploying ibmcloud object storage plugin. IBM Cloud Object Storage is persistent,...
IBM Operator Catalog Enablement	Helm Charts	Provided by IBM	IBM Operator Catalog enablement deploys custom CatalogSources and...

Source to image Python I



Sample repository

https://github.com/sclorg/django-ex.git

Provider

N/A

Created at

16 Nov 2022, 11:28

Support

N/A

Documentation

N/A

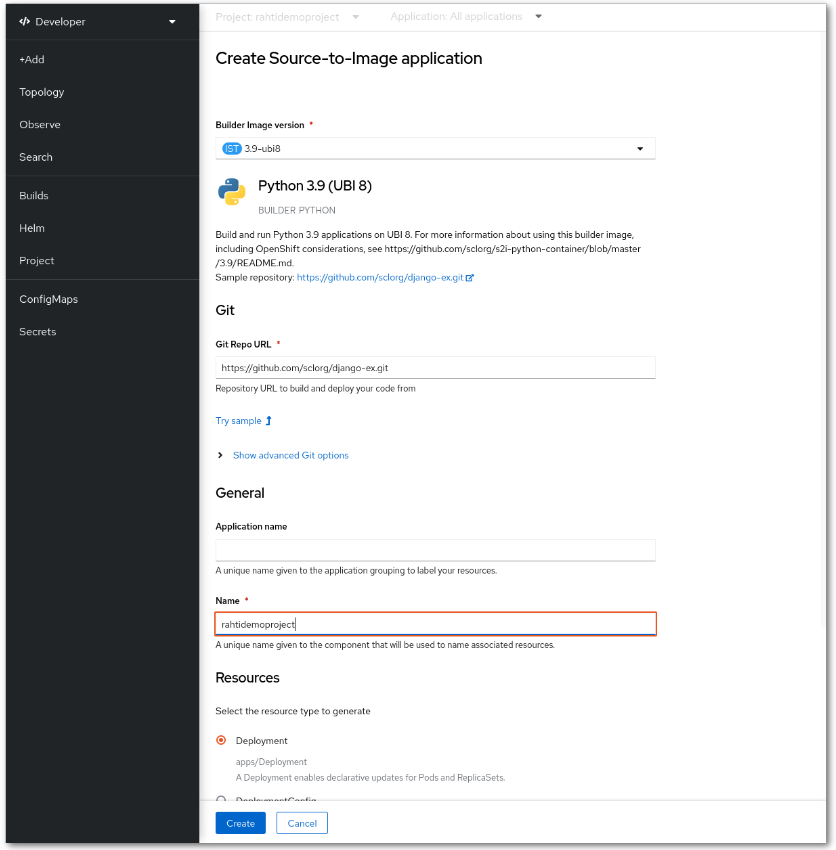
Description

Build and run Python 3.9 applications on UBI 8. For more information about using this builder image, including OpenShift considerations, see https://github.com/sclorg/s2i-python-container/blob/master/3.9/README.md.

The following resources will be created:

- A **BuildConfig** to build source from a Git repository.
- An **ImageStream** to track built Images.
- A **DeploymentConfig** to rollout new revisions when the Image changes.
- A **Service** to expose your workload inside the Cluster.
- An optional **Route** to expose your workload outside the Cluster.

Source to image Python II



Create Source-to-Image application

Builder Image version *

IST 3.9-ubi8

Python 3.9 (UBI 8)

BUILDER PYTHON

Build and run Python 3.9 applications on UBI 8. For more information about using this builder image, including OpenShift considerations, see https://github.com/sclorg/s2i-python-container/blob/master/3.9/README.md.
Sample repository: https://github.com/sclorg/django-ex.git

Git

Git Repo URL *

https://github.com/sclorg/django-ex.git

Repository URL to build and deploy your code from

Try sample

Show advanced Git options

General

Application name

A unique name given to the application grouping to label your resources.

Name *

rahtidemoproject

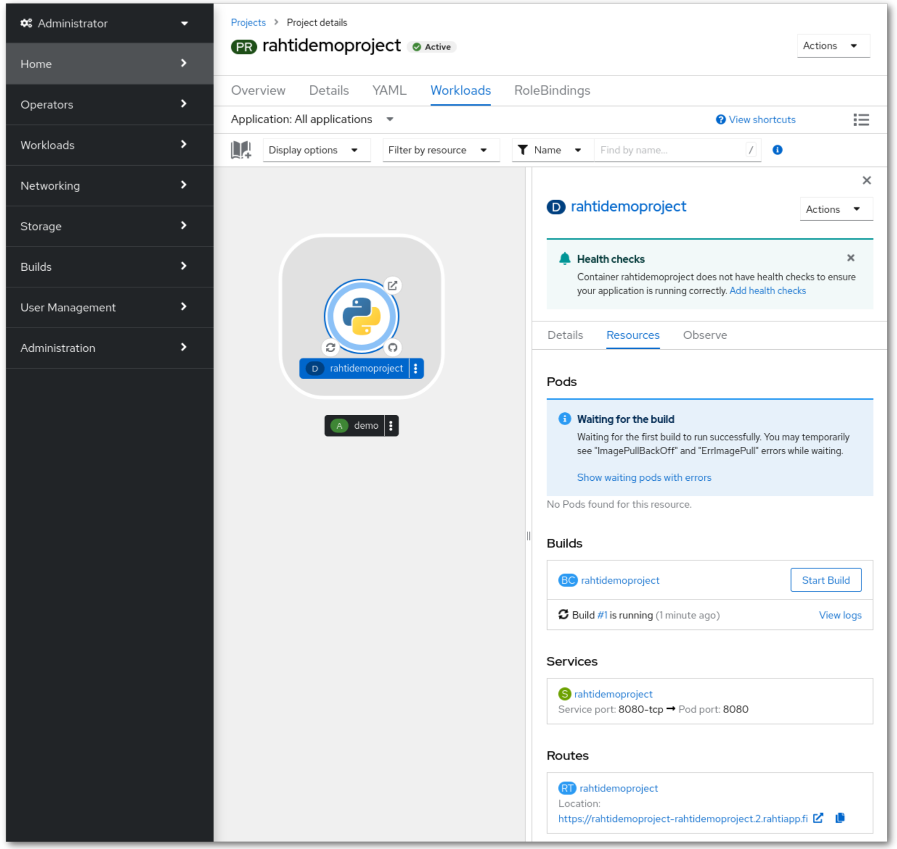
A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

[✓ Deployment]
apps/Deployment
A Deployment enables declarative updates for Pods and ReplicaSets.

Source to image Python III



Health checks

Container rahtidemoproject does not have health checks to ensure your application is running correctly. Add health checks

Pods

Waiting for the build

Waiting for the first build to run successfully. You may temporarily see "ImagePullBackOff" and "ErrImagePull" errors while waiting.

Show waiting pods with errors

No Pods found for this resource.

Builds

BC rahtidemoproject

Build #1 is running (1 minute ago)

Services

S rahtidemoproject

Service port: 8080-tcp → Pod port: 8080

Routes

RT rahtidemoproject

Location:

https://rahtidemoproject-rahtidemoproject.2.rahtiapp.fi

Rahti will automatically:

- Fetch the code
- Analyze it
- Build a new image
- Deploy it
- Make it available to the Internet

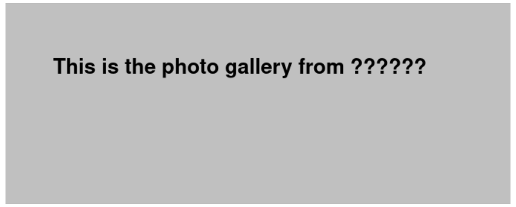
Demo II

Flask hello-world in Rahti

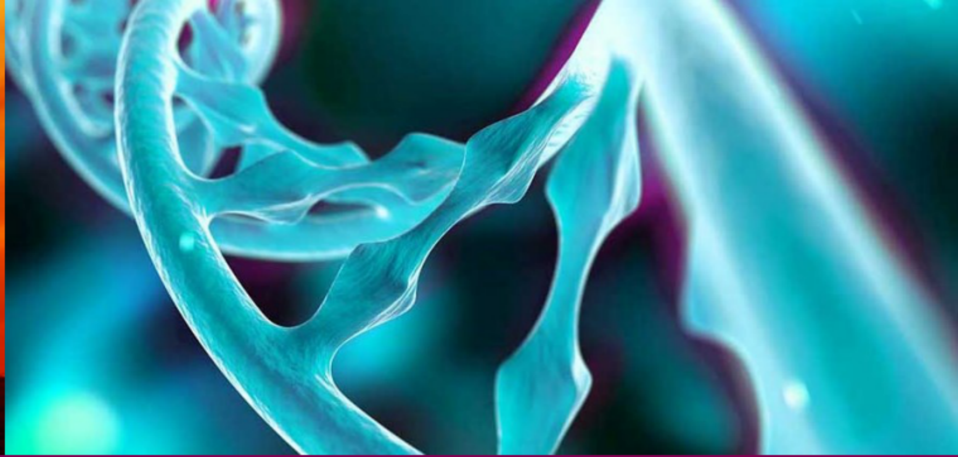
Using the web interface deploy:

```
https://github.com/cscfi/rahti-flask-hello
```

- Use the project flask-demo
- Rahti automatically builds a **container image** given application sources.
- then the system *orchestrates* all the components so the application becomes available



This is the photo gallery from ??????

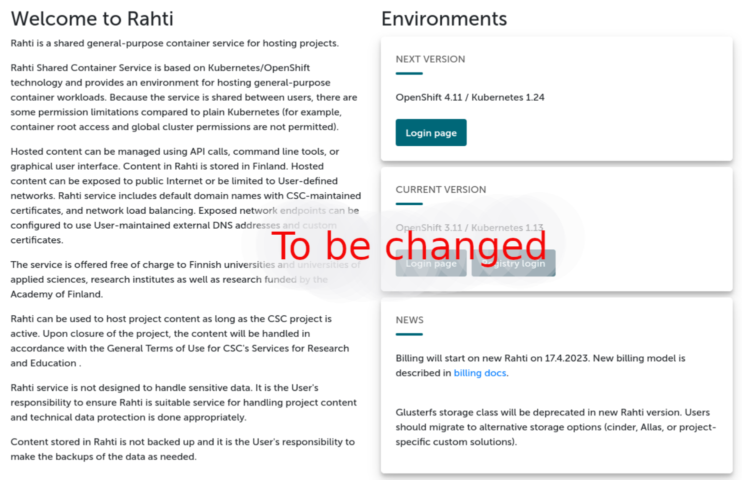


Web interface Howtos

short howtos for the exercises

Logging in on web console

- Navigate to https://rahti.csc.fi.
- Click in "Login page"
- Select CSC or Haka. Use your own account.



Welcome to Rahti

Rahti is a shared general-purpose container service for hosting projects.

Rahti Shared Container Service is based on Kubernetes/OpenShift technology and provides an environment for hosting general-purpose container workloads. Because the service is shared between users, there are some permission limitations compared to plain Kubernetes (for example, container root access and global cluster permissions are not permitted).

Hosted content can be managed using API calls, command line tools, or graphical user interface. Content in Rahti is stored in Finland. Hosted content can be exposed to public Internet or be limited to User-defined networks. Rahti service includes default domain names with CSC-maintained certificates, and network load balancing. Exposed network endpoints can be configured to use User-maintained external DNS addresses and custom certificates.

The service is offered free of charge to Finnish universities and universities of applied sciences, research institutes as well as research funded by the Academy of Finland.

Rahti can be used to host project content as long as the CSC project is active. Upon closure of the project, the content will be handled in accordance with the General Terms of Use for CSC's Services for Research and Education .

Rahti service is not designed to handle sensitive data. It is the User's responsibility to ensure Rahti is suitable service for handling project content and technical data protection is done appropriately.

Content stored in Rahti is not backed up and it is the User's responsibility to make the backups of the data as needed.

Environments

NEXT VERSION

OpenShift 4.11 / Kubernetes 1.24

Login page

CURRENT VERSION

OpenShift 3.11 / Kubernetes 1.13

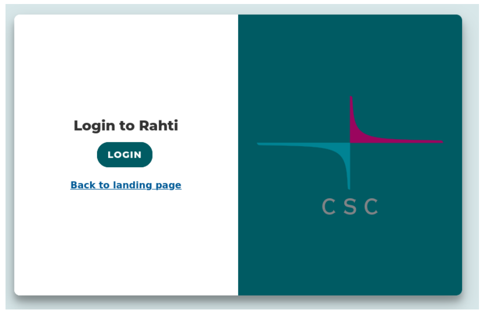
Login page Registry login

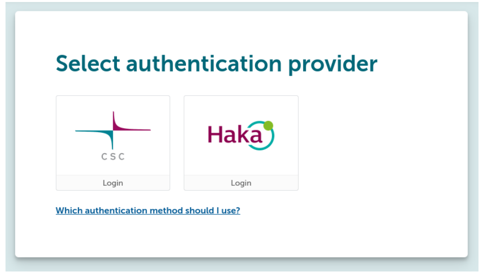
To be changed

NEWS

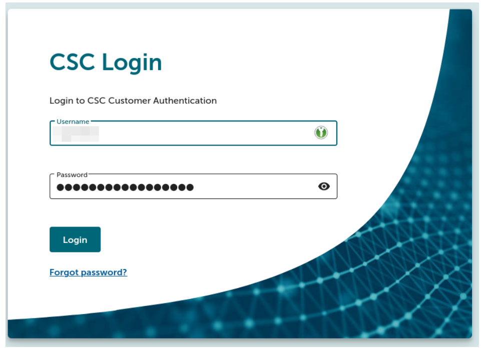
Billing will start on new Rahti on 17.4.2023. New billing model is described in billing docs.

Glusterfs storage class will be deprecated in new Rahti version. Users should migrate to alternative storage options (cinder, Allas, or project-specific custom solutions).



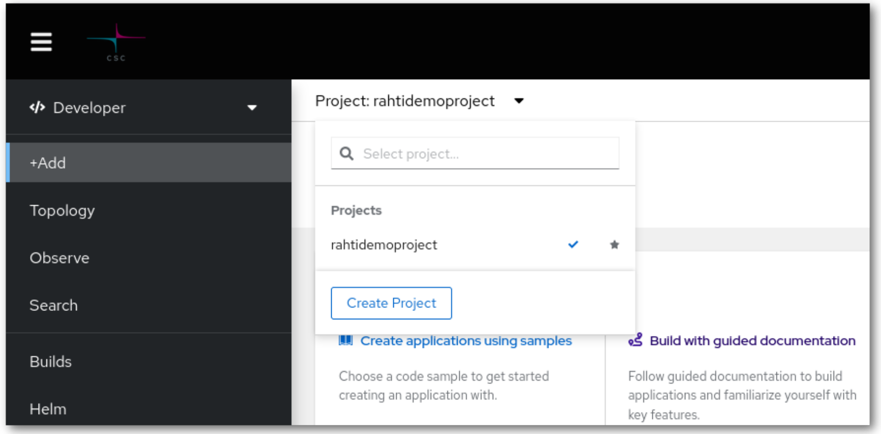


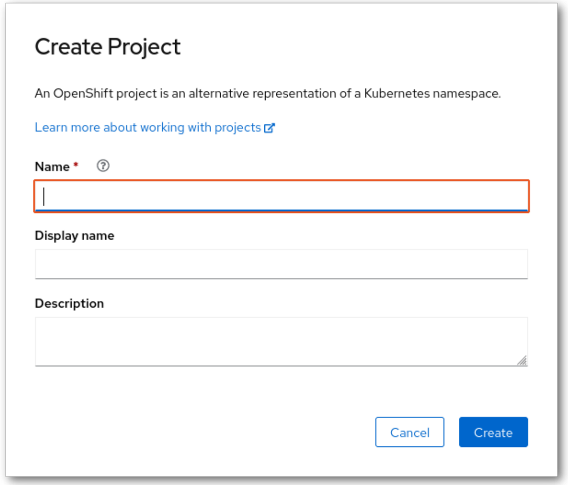
Select authentication provider
CSC
Login
Haka
Login
Which authentication method should I use?



Creating a project

- Click in "Create Project"
  - **Name**: Short name that will be used to reference the project
  - **Display Name**: Descriptive name that should make clear what the project is
  - **Description**: It **must** be: "csc_project: ?????????". It must be associated to a CSC project for billing purposes.
- Initial quota of 5 projects per user


Developer
+Add
Topology
Observe
Search
Builds
Helm
Project: rahtidemoproject
Select project...
Projects
rahtidemoproject
Create Project
Create applications using samples
Choose a code sample to get started creating an application with.
Build with guided documentation
Follow guided documentation to build applications and familiarize yourself with key features.


Create Project

An OpenShift project is an alternative representation of a Kubernetes namespace.

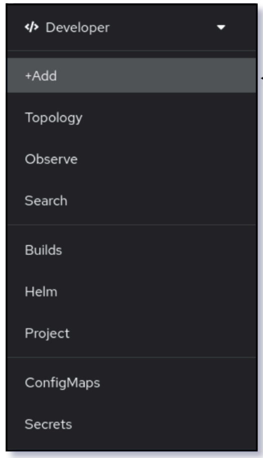
Learn more about working with projects

Name *

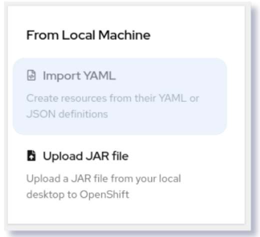
Display name

Description

Creating *API* objects (WEB)

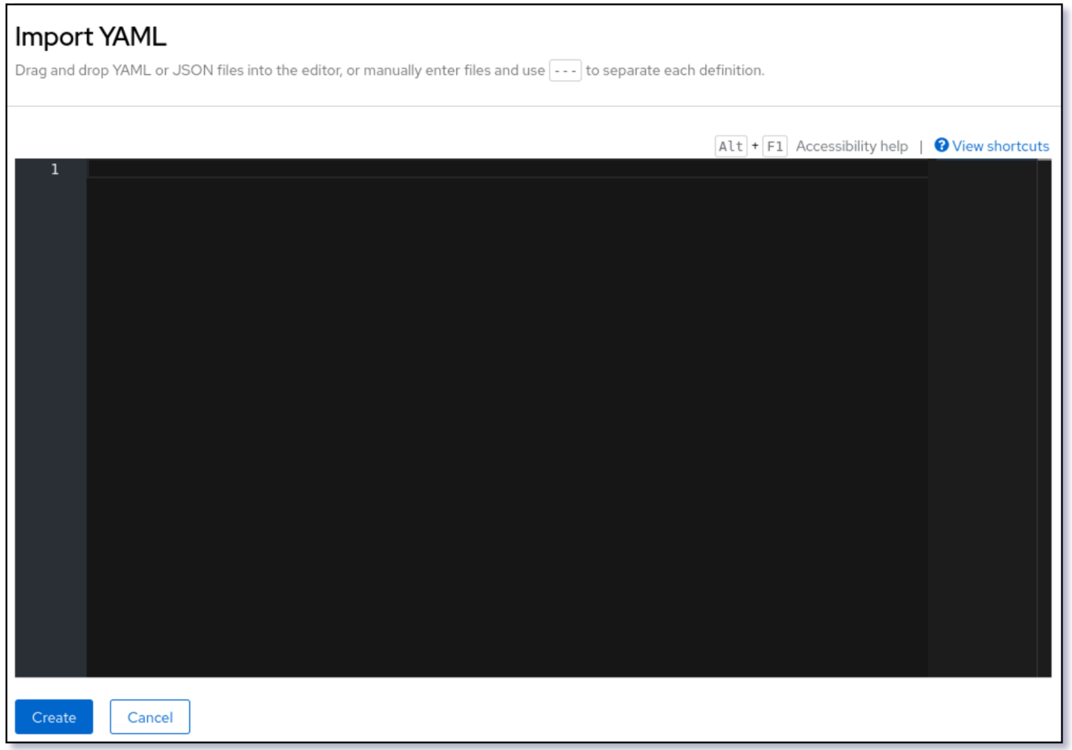


Click in "+Add" in the "Developer console"


From Local Machine
Import YAML
Create resources from their YAML or JSON definitions
Upload JAR file
Upload a JAR file from your local desktop to OpenShift

Click in "Import YAML"

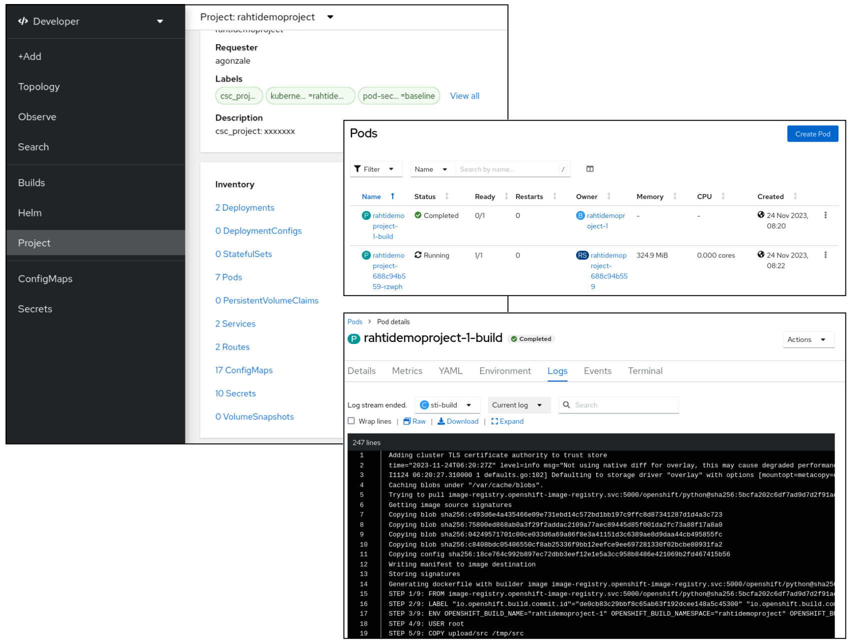
Paste the YAML (or JSON) code of the object


Import YAML

Drag and drop YAML or JSON files into the editor, or manually enter files and use `---` to separate each definition.

Click Create

How to see application information?



Requester
agonzale

Labels
csc_proj... kuberne... =rahtide... pod-sec... =baseline

Description
csc_project: xxxxxxx

Inventory
2 Deployments
0 DeploymentConfigs
0 StatefulSets
7 Pods
0 PersistentVolumeClaims
2 Services
2 Routes
17 ConfigMaps
10 Secrets
0 VolumeSnapshots

Pods

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
rahtidemoproject-1-build	Completed	0/1	0	rahtidemoproject-1	-	-	24 Nov 2023, 08:20
rahtidemoproject-688c94b559-rzwph	Running	1/1	0	rahtidemoproject-688c94b559	324.9 MiB	0.000 cores	24 Nov 2023, 08:22

rahtidemoproject-1-build Completed

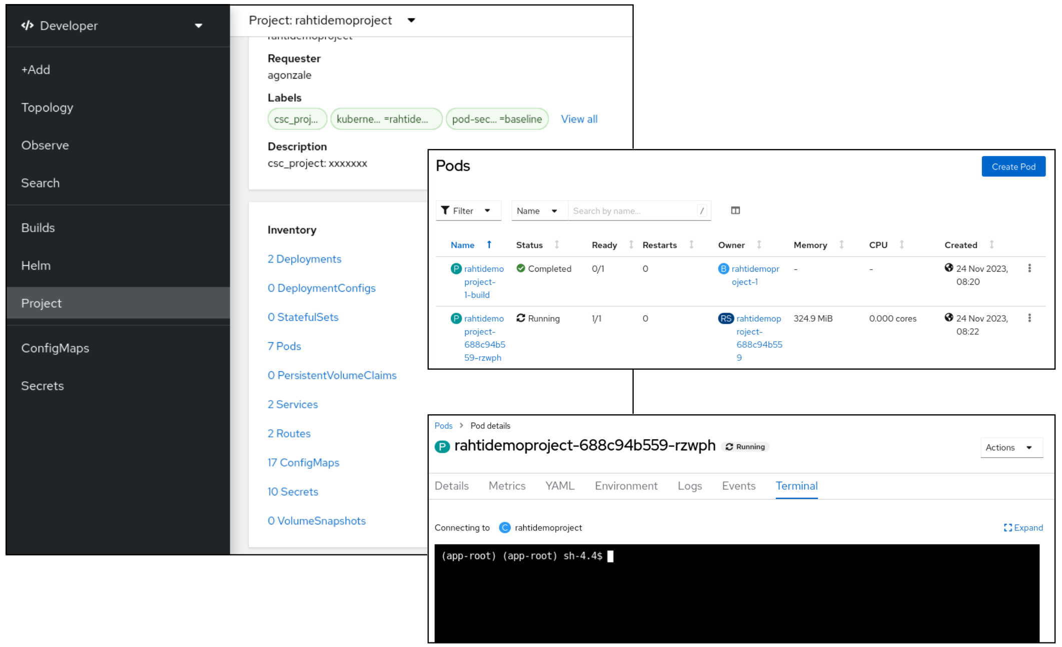
Log stream ended.

247 lines

```
Adding cluster TLS certificate authority to trust store
time="2023-11-24T06:20:27Z" level=info msg="Not using native diff for overlay, this may cause degraded performanc
I1124 06:20:27.310000 1 defaults.go:102] Defaulting to storage driver "overlay" with options [mountopt=metacopy=
Caching blobs under "/var/cache/blobs".
Trying to pull image-registry.openshift-image-registry.svc:5000/openshift/python@sha256:5bcfa202c6df7ad9d7d2f91a
Getting image source signatures
Copying blob sha256:c493d6e4a435466e09e731ebd14c572bd1bb197c9ffc8d87341287d1d4a3c723
Copying blob sha256:75800ed868ab0a3f29f2addac2109a77aec89445d85f001da2fc73a88f17a8a0
Copying blob sha256:04249571701c00ce033d6a69a86f8e3a41151d3c6389ae8d9daa44cb495855fc
Copying blob sha256:c8408bdc05406550cf8ab25336f9bb12eefce9ee697281330f02bcbe80931fa2
Copying config sha256:18ce764c992b897ec72dbb3eef12e1e5a3cc958b8486e421069b2fd467415b56
Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-registry.svc:5000/openshift/python@sha25
STEP 1/9: FROM image-registry.openshift-image-registry.svc:5000/openshift/python@sha256:5bcfa202c6df7ad9d7d2f91a
STEP 2/9: LABEL "io.openshift.build.commit.id"="de0cb83c29bbf8c65ab63f192dcee148a5c45300" "io.openshift.build.co
STEP 3/9: ENV OPENSHIFT_BUILD_NAME="rahtidemoproject-1" OPENSHIFT_BUILD_NAMESPACE="rahtidemoproject" OPENSHIFT_B
STEP 4/9: USER root
STEP 5/9: COPY upload/src /tmp/src
```

- In the "Developer console", click in "Project", scroll down until "Inventory" and click in "Pods".
- Click in **any** Pod that you want to see more information about
- You can see:
  - General **Details** of the Pod.
  - Read **Metrics** like CPU, Memory, Filesystem usage and Network.
  - The **YAML** representation can be seen and edited.
  - The **Environment** variables configured and theirs values.
  - The **Logs** can be seen in real time.
  - **Events** like image pull errors.

How to open a terminal session



Requester
agonzale

Labels
csc_proj... kuberne... =rahtide... pod-sec... =baseline View all

Description
csc_project: xxxxxxx

Inventory
2 Deployments
0 DeploymentConfigs
0 StatefulSets
7 Pods
0 PersistentVolumeClaims
2 Services
2 Routes
17 ConfigMaps
10 Secrets
0 VolumeSnapshots

Pods

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
rahtidemoproject-1-build	Completed	0/1	0	rahtidemoproject-1	-	-	24 Nov 2023, 08:20
rahtidemoproject-688c94b559-rzwph	Running	1/1	0	rahtidemoproject-688c94b559	324.9 MiB	0.000 cores	24 Nov 2023, 08:22

Pods > Pod details

rahtidemoproject-688c94b559-rzwph Running

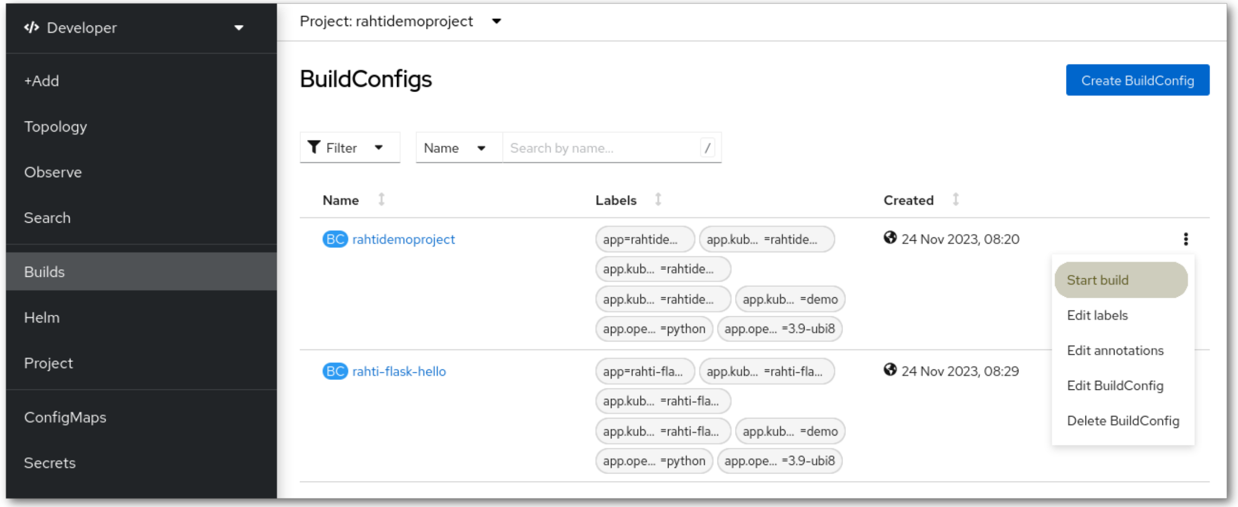
Connecting to rahtidemoproject

```
(app-root) (app-root) sh-4.4$
```

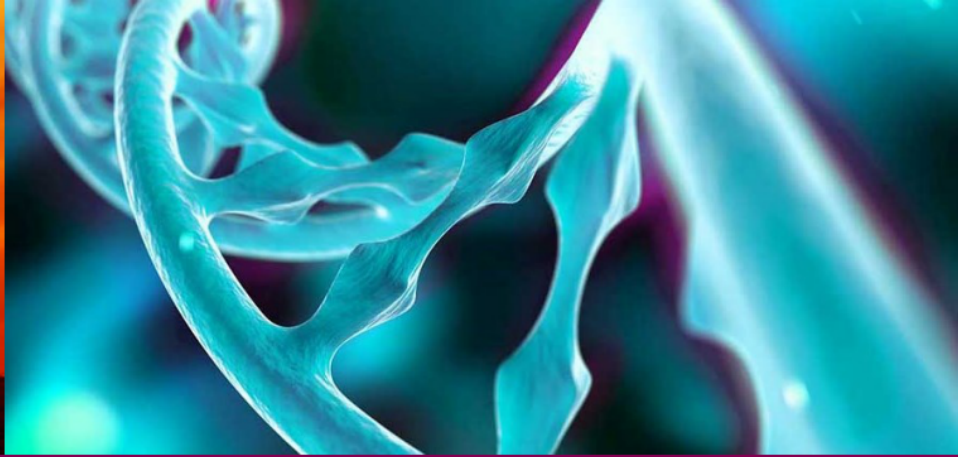
- In the "Developer console", click in "Project", scroll down until "Inventory" and click in "Pods".
- It is only available for ☒ **Running** Pods.
- It allows an interactive session.

Launching a build

- Go to the **BuildConfigs** page
- By clicking in the *3 dots icon* of the build you want to start, a drop down menu will appear.
- Click in "Start build"



Name	Labels	Created
BC rahtidemoproject	app=rahtide... app.kub... =rahtide... app.kub... =rahtide... app.kub... =rahtide... app.kub... =demo app.ope... =python app.ope... =3.9-ubi8	24 Nov 2023, 08:20
BC rahti-flask-hello	app=rahti-fla... app.kub... =rahti-fla... app.kub... =rahti-fla... app.kub... =rahti-fla... app.kub... =demo app.ope... =python app.ope... =3.9-ubi8	24 Nov 2023, 08:29



Coffee break

🥐 ☕

15 min

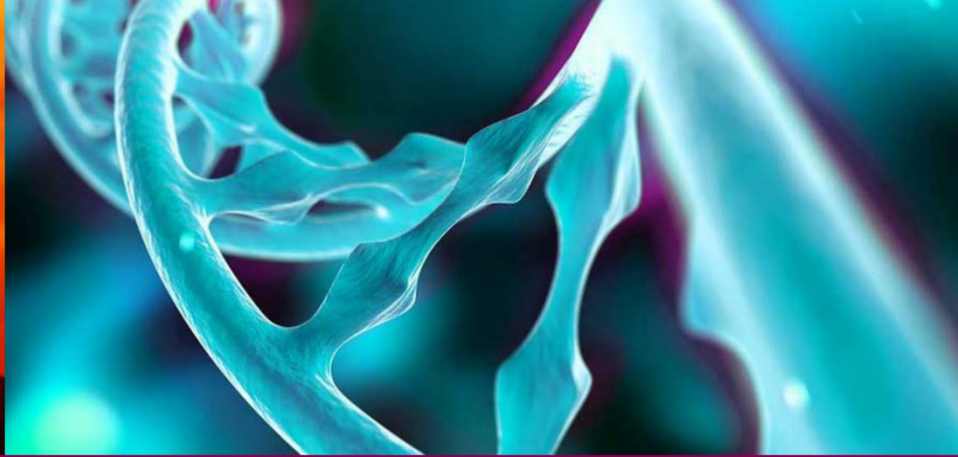
CSC – Finnish expertise in ICT for research, education and public administration

Exercises A

Go to the exercises page.

1. Authorizing client session and creating a project
2. Create python application in Rahti
3. Explore python application
4. Modify python application

Note: It is possible to do these exercises using only the web interface

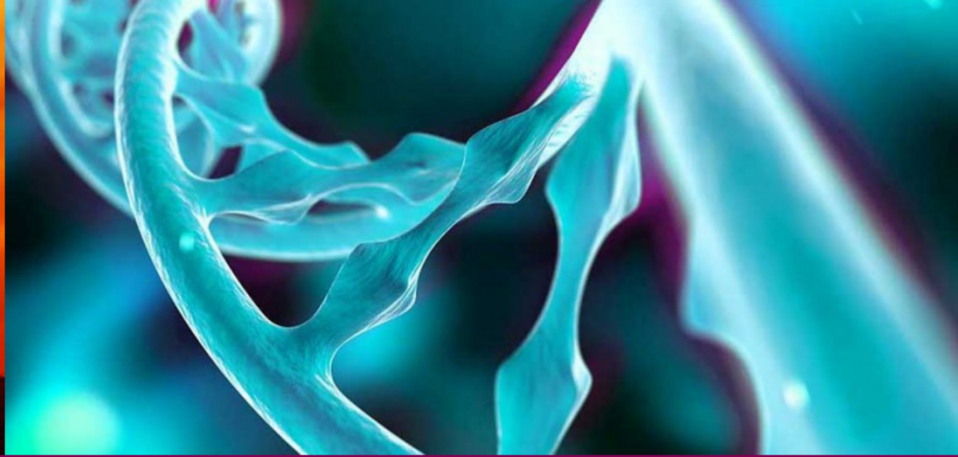


Lunch break

🍲 🍜

60 minutes

CSC – Finnish expertise in ICT for research, education and public administration



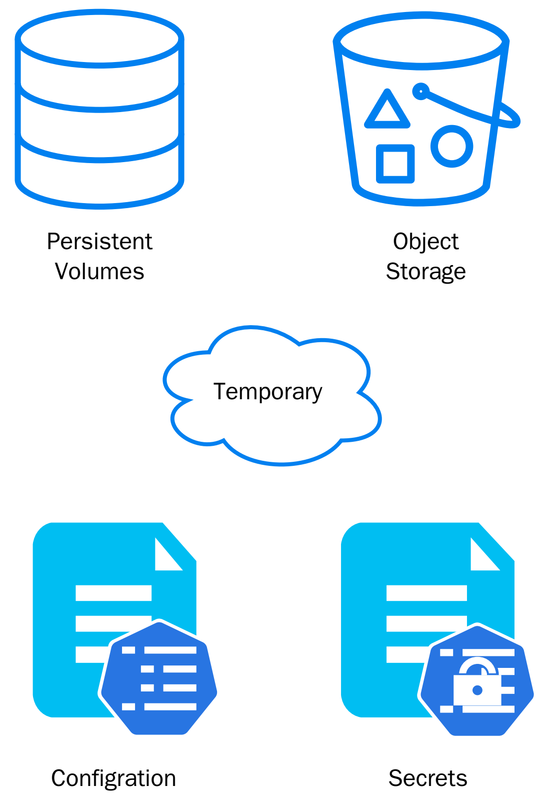
Storage

Storage

Containers are ephemeral, this means any **change** done to a container image will be **lost** upon restart. Due to the nature of container orchestration, container restarts are part of the life cycle of a cloud application. When a new version is deployed, a configuration change, or of course uncheduled failures.

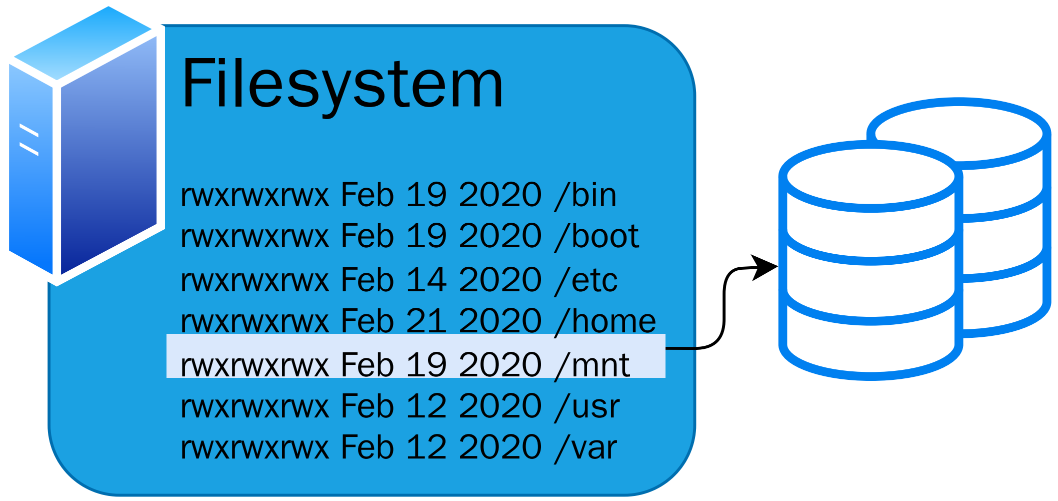
For these reasons we need to have storage solutions, Rahti provides several.

Storage types



1. Persistent Volumes:
   - Traditional filesystem approach.
   - When the application expects a traditional filesystem.
2. Temporary storage:
   - Traditional filesystem approach.
   - When read and write speeds are the most important.
3. Object storage, Allas. S3/Swift:
   - HTTP interface
   - Highly scalable
   - Useful for large volumes of data
4. Configuration: ConfigMaps (and Secrets):
   - Specific API object to store configuration

Persistent Volumes

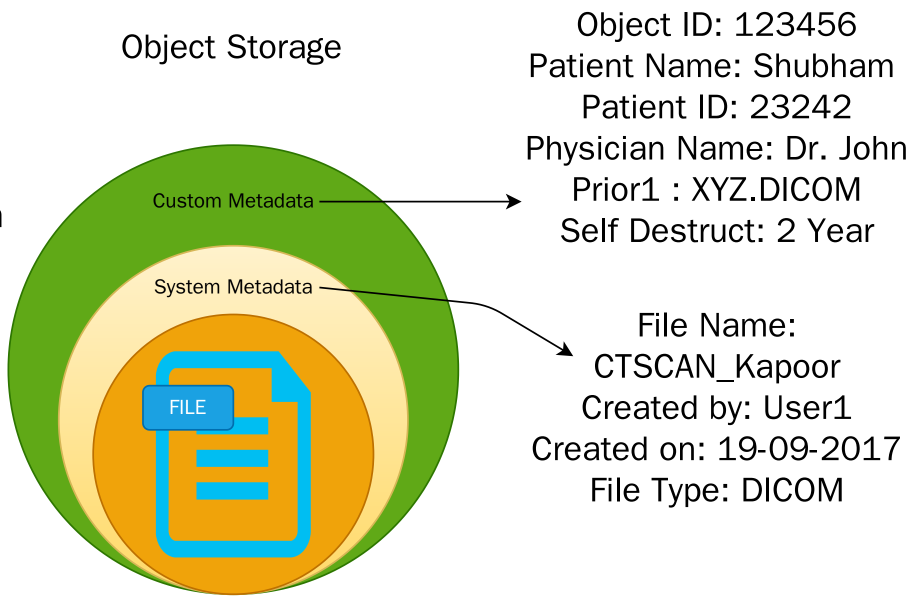

Filesystem
rwxrwxrwx Feb 19 2020 /bin
rwxrwxrwx Feb 19 2020 /boot
rwxrwxrwx Feb 14 2020 /etc
rwxrwxrwx Feb 21 2020 /home
rwxrwxrwx Feb 19 2020 /mnt
rwxrwxrwx Feb 12 2020 /usr
rwxrwxrwx Feb 12 2020 /var

- Traditional filesystem approach:
  - Folder mounted in file hierarchy
- Technology used is **Cinder**.

Allas, Object storage

Object storage is a computer data storage architecture that manages data as objects.

- Different from **Persistent Volumes**:
  - Data within a bucket, not as a file hierarchy.
  - It can be read and write only as a whole.
- Accessed via APIs/HTTP at application-level, rather than via OS at system level.
- Scalable and Self healing storage, thanks to replicas.



Configuration (and secrets)

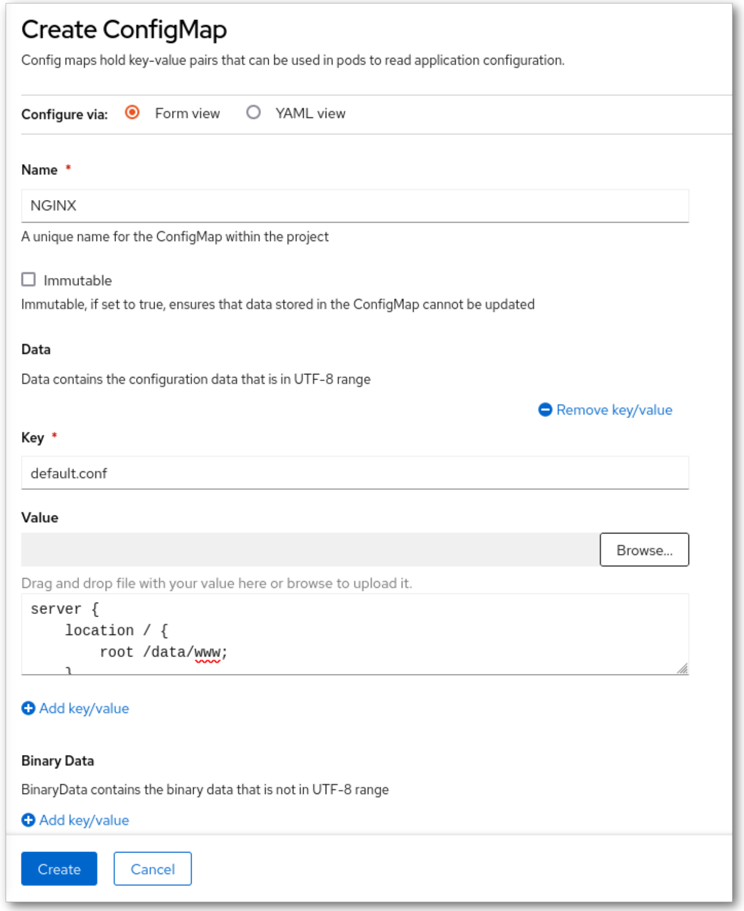
- Stored as internal API objects.
- Configuration files:
  - Can be edited directly in the Web interface,
  - or as YAML or JSON objects.
- Could be *mounted as files*:

```
Filesystem   Size    Used Available Use% Mounted on
/dev/device  3.9T    177.4M      3.9T   0% /etc/config
```

or as *environment variables*.

```
USER=admin
PASSWORD=7h15_15_n07_4_p422W0rD
```

- Secrets have an extra layer of security.



Create ConfigMap

Config maps hold key-value pairs that can be used in pods to read application configuration.

Configure via: [✓ Form view] YAML view

Name *

NGINX

A unique name for the ConfigMap within the project

☐ Immutable

Immutable, if set to true, ensures that data stored in the ConfigMap cannot be updated

Data

Data contains the configuration data that is in UTF-8 range

Key *

default.conf

Value

Drag and drop file with your value here or browse to upload it.

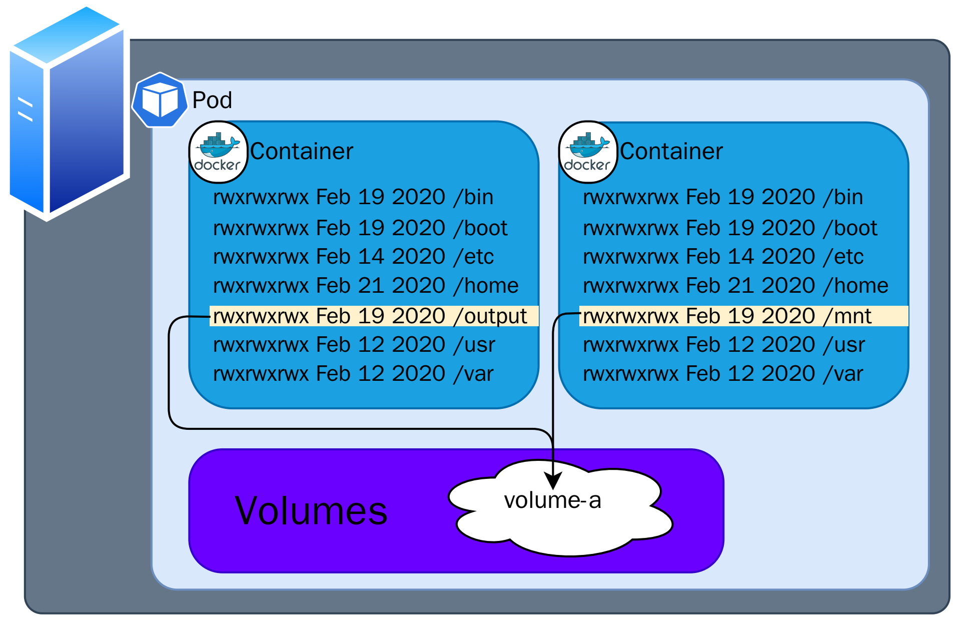
```
server {
    location / {
        root /data/www;
```

Binary Data

BinaryData contains the binary data that is not in UTF-8 range

Temporary storage

- Traditional filesystem approach, **emptyDir**:
  - Folder mounted in file hierarchy.
- **Local** temporary storage:
  - It is the fastest volume type available.
  - **Data is deleted when the application is restarted.**

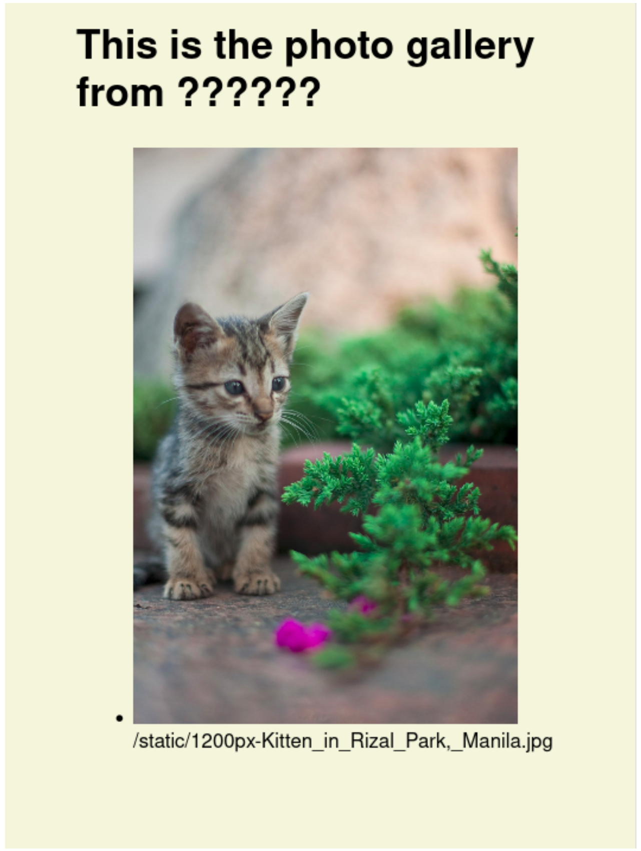

Pod
Container
docker
rwxrwxrwx Feb 19 2020 /bin
rwxrwxrwx Feb 19 2020 /boot
rwxrwxrwx Feb 14 2020 /etc
rwxrwxrwx Feb 21 2020 /home
rwxrwxrwx Feb 19 2020 /output
rwxrwxrwx Feb 12 2020 /usr
rwxrwxrwx Feb 12 2020 /var
Container
docker
rwxrwxrwx Feb 19 2020 /bin
rwxrwxrwx Feb 19 2020 /boot
rwxrwxrwx Feb 14 2020 /etc
rwxrwxrwx Feb 21 2020 /home
rwxrwxrwx Feb 19 2020 /mnt
rwxrwxrwx Feb 12 2020 /usr
rwxrwxrwx Feb 12 2020 /var
Volumes
volume-a

Demo III

Add storage to previous demo

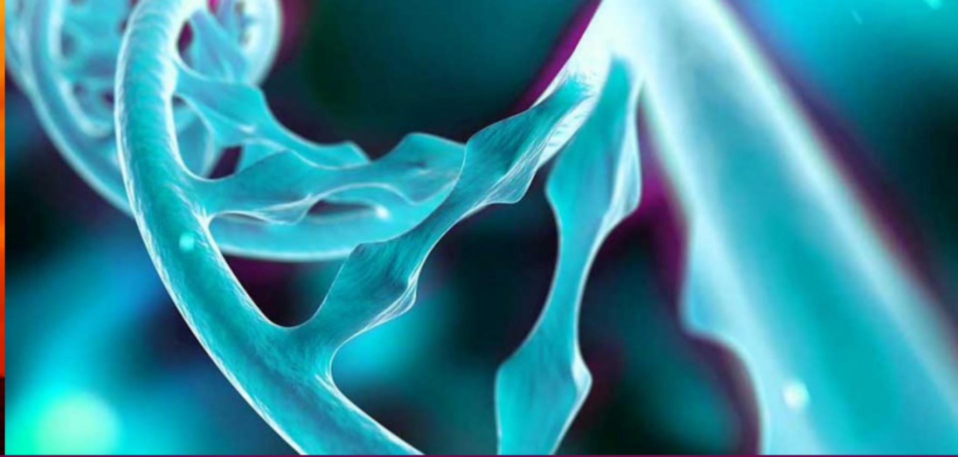
Using the web interface

- Use the same project used in Demo II, flask-demo
- Add a *cinder* volume and mount it to `/static/`.
- Add this kitten photo



This is the photo gallery from ??????

- /static/1200px-Kitten_in_Rizal_Park,_Manila.jpg

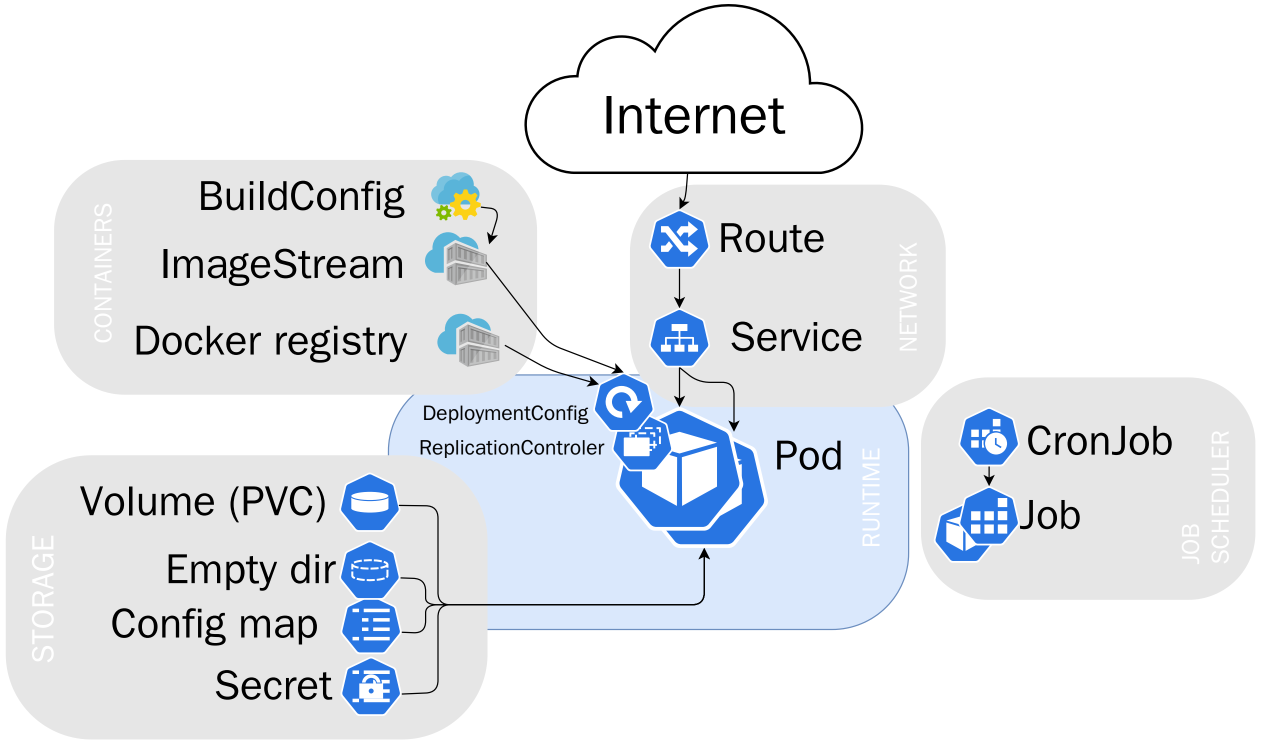


High level Kubernetes architecture

API Objects

In Kubernetes everything is an API object

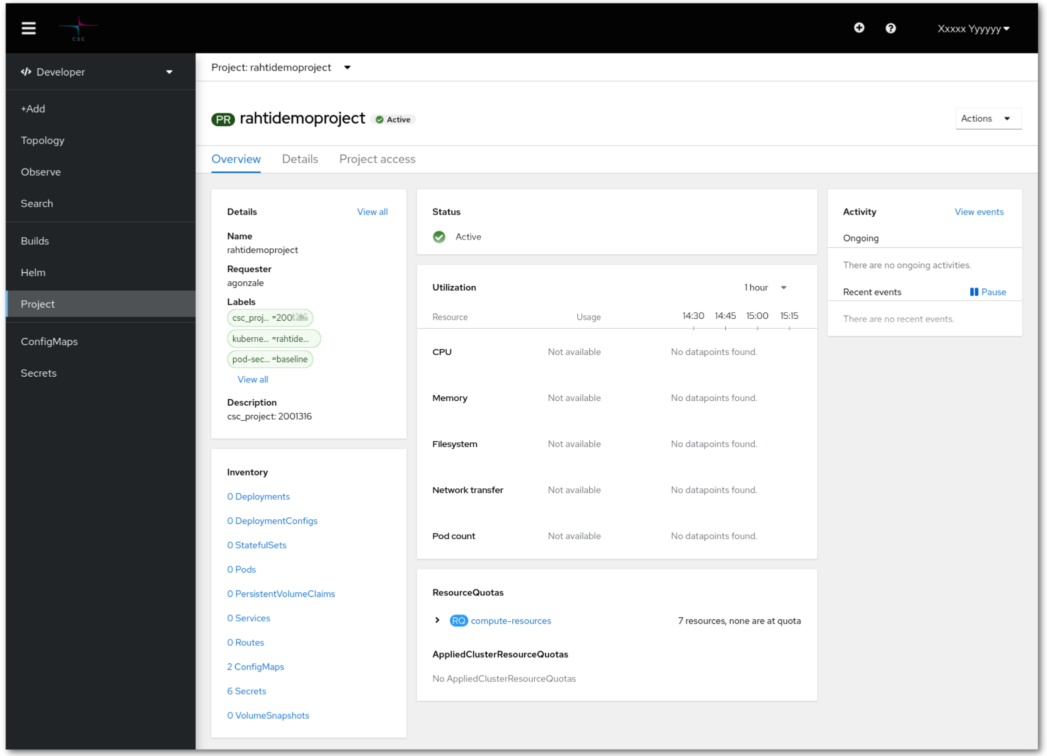
- Complex set of API objects:
  - Network
  - Container, management and creation
  - Job scheduling
  - Runtime of containers
  - Storage



Project

A project sandboxes API objects (Pods and others) in a common namespace.

- Local isolated network
- For security reasons, projects can not access other projects by default.
- Similar to **Namespace**
  - (with extra features)
- A project has:
  - **Name**: Should be short and descriptive
  - **Display Name**: Should be understandable
  - **Description**: Must be `csc_project: 9999999`
    - where `9999999` is the project number



Details

Name
rahtidemoproject

Requester
agonzale

Labels
csc_proj... =200
kuberne... =rahtide...
pod-sec... =baseline

Description
csc_project: 2001316

Inventory

0 Deployments
0 DeploymentConfigs
0 StatefulSets
0 Pods
0 PersistentVolumeClaims
0 Services
0 Routes
2 ConfigMaps
6 Secrets
0 VolumeSnapshots

Status

Active

Utilization

Resource	Usage	14:30 14:45 15:00 15:15
CPU	Not available	No datapoints found.
Memory	Not available	No datapoints found.
Filesystem	Not available	No datapoints found.
Network transfer	Not available	No datapoints found.
Pod count	Not available	No datapoints found.

ResourceQuotas

RQ compute-resources — 7 resources, none are at quota

AppliedClusterResourceQuotas

No AppliedClusterResourceQuotas

Activity

Ongoing

There are no ongoing activities.

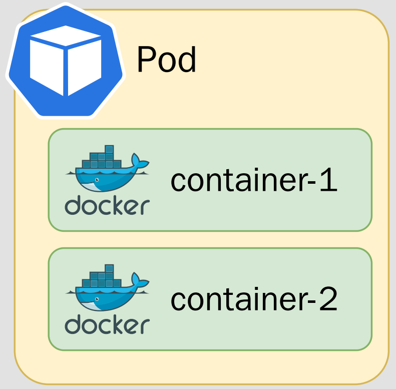
Recent events

There are no recent events.

Pod

- A pod is a collection of **containers** sharing a network and Inter-process communication namespace
  - Containers live in one pod
- There is no *container object* in Kubernetes
- Nearly always one container per pod

```yaml
# my-pod.yaml
kind: Pod
apiVersion: v1
metadata:
 name: my-pod
spec:
 containers:
 - name: container-1
   image: image-1
 - name: container-2
   image: image-2
```



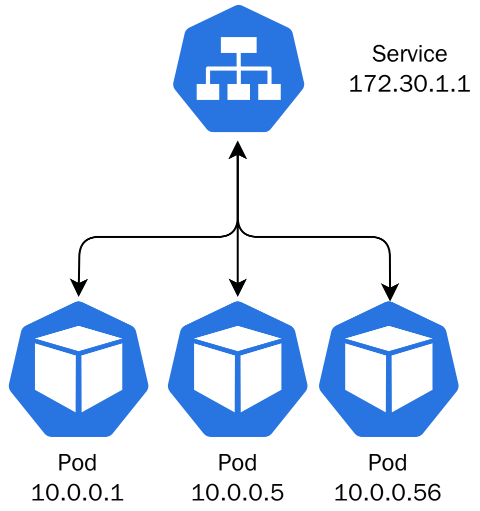
Communicate via

- `localhost` (network)
- memory (Inter-process communication)

Service

An API object that provides pods a load balanced stable network identity.

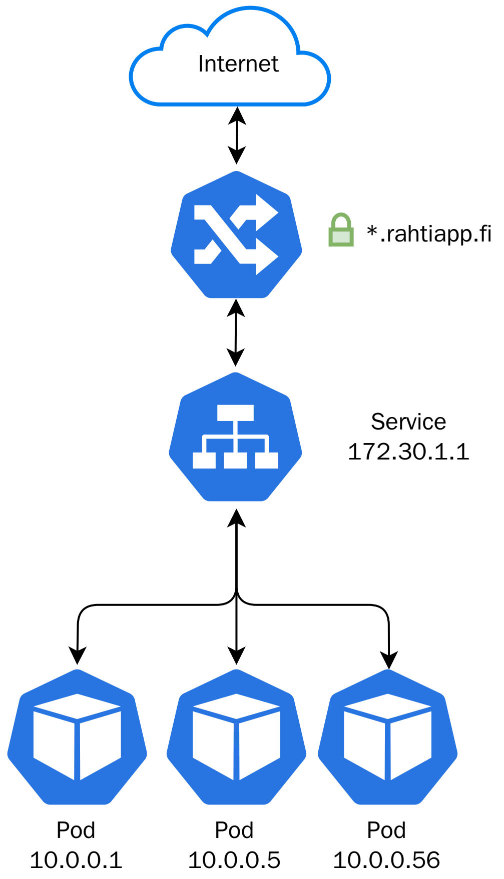
- The IP of a Pod may change, the **IP** of a Service **will not change**.
- Under one Service, there may be several pods.
- Tips:
  - Several ports can be exposed in the same service.
  - The one exposed port to the incoming traffic, may be different than the port in the container.

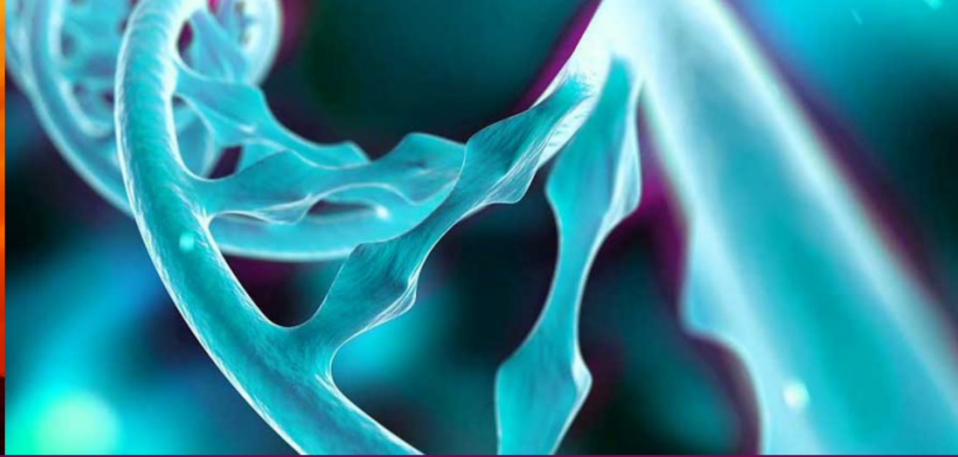


Route

An API object that exposes a Service to the internet via HTTP/HTTPS.

- Every host with the pattern `*.rahtiapp.fi` will point **automatically** to Rahti:
  - `my-hello-openshift.rahtiapp.fi` is an alias for `rahtiapp.fi`.
  - If the host must be different to this pattern, a `DNS CNAME` entry must be configured by the user to point to `rahtiapp.fi`.
- Every host with the pattern `*.rahtiapp.fi` will have automatically a valid **TLS certificate**.





Command line tool *oc*

The oc command

- `oc` is the OpenShift command line client*.
- Some common commands:
  - **LOGIN**, `oc login`. Could take a TOKEN or a username/password.
  - **PROJECT MANAGEMENT**, `oc projects` and `oc new-project`. List, switch, and create projects.
  - **INFORMATION**, `oc get` and `oc describe`. Describe is more detailed and more human friendly, and get is more machine friendly (JSON and YAML outputs).
  - **CREATE**, `oc create`.
  - **MODIFY**, `oc edit` and `oc replace`. Edit is interactive.
  - **DELETE**, `oc delete`.

*`kubectl` is equivalent for Kubernetes. `oc` features are a superset of `kubectl`.

Installation

The oc tool is a single binary that only needs to be included in your path. Installation:

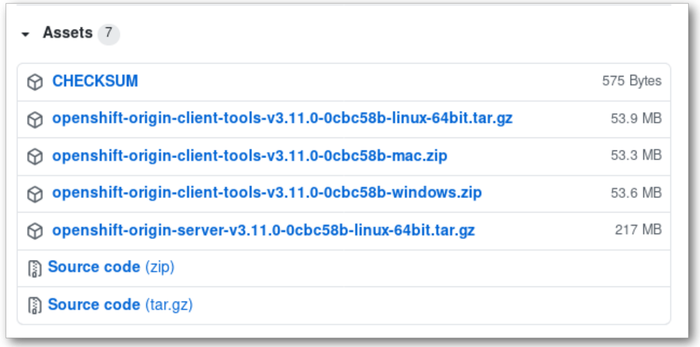
1. Go to the release page https://github.com/openshift/origin/releases/latest.
2. In the bottom you will see the list of clients. Download the "OpenShift origin client" corresponding to your OS (Windows, Mac or Linux).
3. Once downloaded, extract the oc binary file.
4. Copy the file to a folder in your $PATH and make it executable. You can see what is your $PATH by:

   - (Linux/MacOS) Open a terminal and run:

   ```
   $ echo $PATH
   ```

   - (Windows) Open the Command Prompt and run:

   ```
   C:\> path
   ```



Assets 7

CHECKSUM	575 Bytes
openshift-origin-client-tools-v3.11.0-0cbc58b-linux-64bit.tar.gz	53.9 MB
openshift-origin-client-tools-v3.11.0-0cbc58b-mac.zip	53.3 MB
openshift-origin-client-tools-v3.11.0-0cbc58b-windows.zip	53.6 MB
openshift-origin-server-v3.11.0-0cbc58b-linux-64bit.tar.gz	217 MB
Source code (zip)	
Source code (tar.gz)	

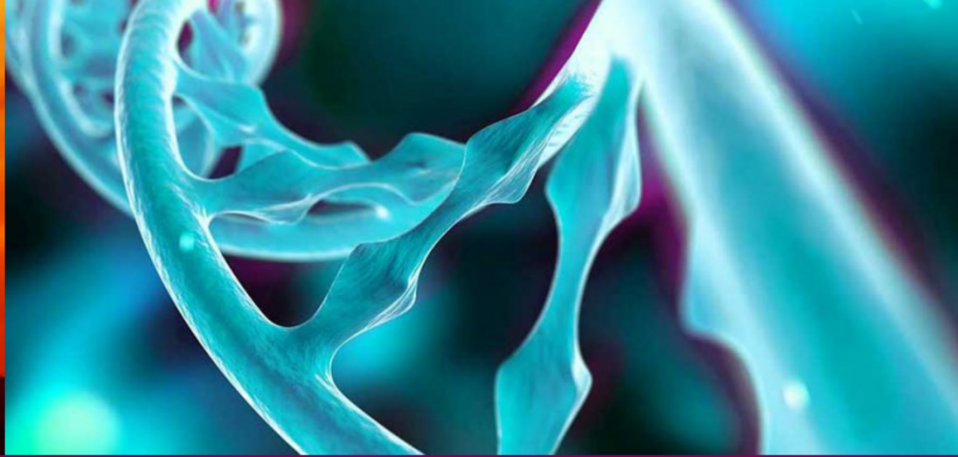
YAML and JSON

Data serialization formats used to represent API objects.

- "YAML Ain't Markup Language" (**YAML**).
- "JavaScript Object Notation" (**JSON**).

```yaml
# hello-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: hello-pod
  namespace: my-unique-project-name
spec:
  containers:
  - name: hello-container
    image: hello-world
```

```json
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "hello-pod",
    "namespace": "my-unique-project-name"
  },
  "spec": {
    "containers": [
      {
        "name": "hello-container",
        "image": "hello-world"
      }
    ]
  }
}
```

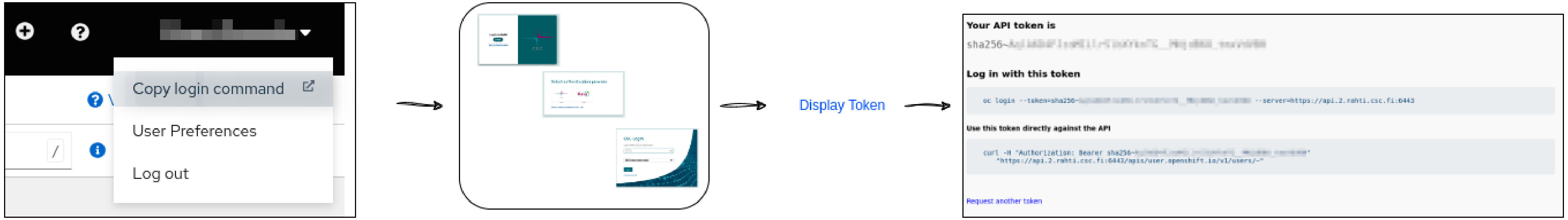


Command line interface Howtos

Logging in on command line interface

Following https://rahti.csc.fi/usage/cli/

- Click in the upper right corner on any Rahti page to reveal the menu option "Copy Login Command":
  - It will copy the login command to the clipboard.
- Paste the command in any Terminal:
  - Places the token in `$HOME/.kube/config`.
  - It will be available in every terminal for the duration of the session.



Your API token is

sha256~[redacted]

Log in with this token

```
oc login --token=sha256~[redacted] --server=https://api.2.rahti.csc.fi:6443
```

Use this token directly against the API

```
curl -H "Authorization: Bearer sha256~[redacted]"
   "https://api.2.rahti.csc.fi:6443/apis/user.openshift.io/v1/users/~"
```

Request another token

Note: Do not share the *TOKEN*, this will be the same as sharing a password.

Creating a project

Same information as in the web interface:

- **Name**: Short name that will be used to reference the project
- **Display Name**: Descriptive name that should make clear what the project is
- **Description**: It must be: "**csc_project: XXXXXXX**". It must be associated to a CSC project for billing purposes.

```
oc new-project nptest \
  --display-name='New project Test' \
  --description='csc_project: 2001316'
```

The output should be something like:

```
Now using project "nptest" on server "https://rahti.csc.fi:8443".

You can add applications to this project with the 'new-app' command. For example, try:

    oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

to build a new example application in Ruby.
```

Creating *API* objects (CLI)

1. Write the API object file. You may use *JSON* or *YAML*. It is recommended to use an existing API object as an initial point.

2. Create the object by calling the file created in the previous step.

   ```
   oc create -f Pod.yaml
   ```

3. Check if it has been created properly

   ```
   oc get pod/hello-pod -o yaml
   ```

```yaml
# Pod.yaml
kind: Pod
apiVersion: v1
metadata:
 name: hello-pod
spec:
 containers:
 - name: hello-container
   image: hello-world
 restartPolicy: OnFailure
```

How to connect to a running pod?

- First, get the name of the Pod to open the interactive session to:
  - and choose any Pod with Running STATUS.

```
$ oc get pods
NAME                 READY     STATUS      RESTARTS   AGE
django-ex-1-build    0/1       Completed   0          2h
django-ex-1-svwg2    1/1       Running     0          2h
```

```
$  oc rsh pod/django-ex-1-svwg2
(app-root) sh-4.2$
```

How to see application logs?

- Similar first step as previously, get the name of the Pod to get logs from:
    - and choose any Pod.

```
$ oc get pods
NAME                READY     STATUS      RESTARTS   AGE
django-ex-1-build   0/1       Completed   0          2h
django-ex-1-svwg2   1/1       Running     0          2h
```

```
$  oc logs pod/django-ex-1-svwg2
---> Migrating database ...
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, welcome
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
```

Edit API objects

It is possible to do this in a single command:

```
oc edit pod/hello-pod
```

It is also possible to get the API object into a file, edit the file with any editor, and replace the object:

1. Get the current object

   ```
   oc get pod/hello-pod -o yaml >hello-pod.yaml
   ```

   - `-o json` is also a possibility instead of `-o yaml`

2. Edit the *YAML* file.

3. Replace the object

   ```
   oc replace --force -f hello-pod.yaml
   ```

Run a container image interactively

It is sometimes useful to be able to run a random container image for debugging inside a project.

- This will run `bash` inside a new pod called `centos-test`, attach stdin to terminal (`--it`), remove it when exiting (`--rm` and `--restart=Never`), and use `centos:7` as container image.

```
$ oc run centos-test --rm -it --image=centos:7 --restart=Never -- /bin/bash
If you do not see a command prompt, try pressing enter.
bash-4.2$
```

Note: This is only possible to do using the command line

Source2Image: CLI

Create a new application automatically from source code. For example:

```
oc new-app https://github.com/openshift/django-ex.git
```

This will clone the GIT repository, analyze it, create a image with the code, and launch it. The only remaining step to make the application accessible to the whole Internet is to:

```
oc expose svc/django-ex
```

Demo IV

hello-world in Rahti

Using the command line

```
oc create -f hello-pod.yaml
```

```
# hello-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: hello-pod
  labels:
    app: hello-pod
spec:
  containers:
  - name: hello-container
    image: openshift/hello-openshift
  restartPolicy: Never
```

```
oc create -f hello-service.yaml
```

```
# hello-service.yaml
kind: Service
apiVersion: v1
metadata:
  name: hello-service
spec:
  ports:
    - name: 8888-8888
      port: 8888
      protocol: TCP
      targetPort: 8888
  selector:
    app: hello-pod
  type: LoadBalancer
status: {}
```

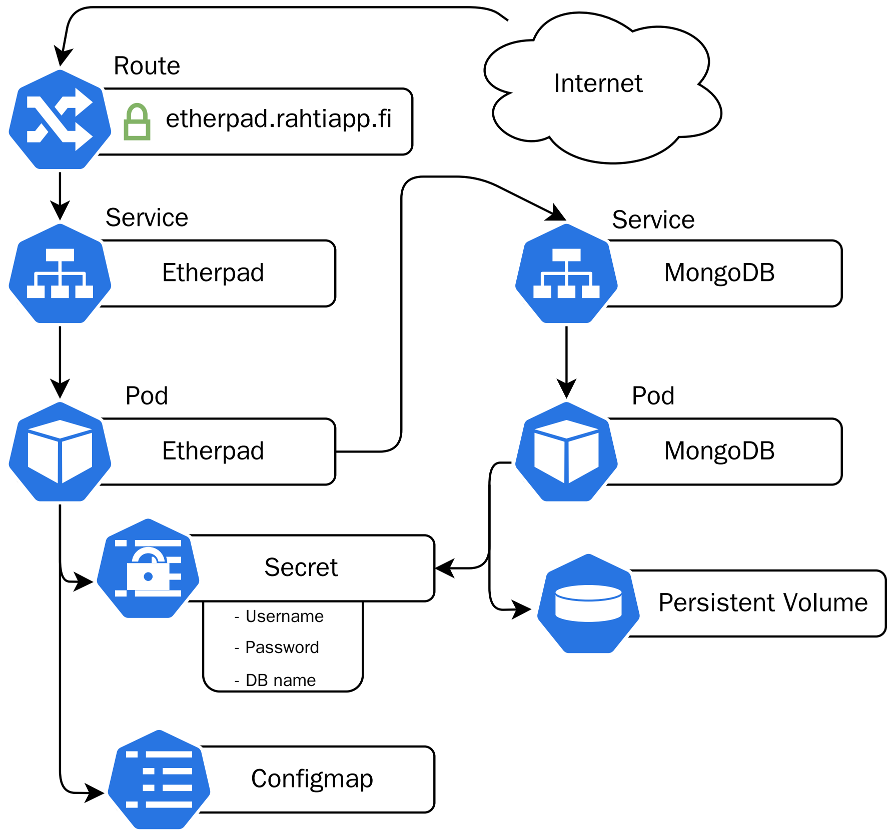
```
oc create -f hello-route.yaml
```

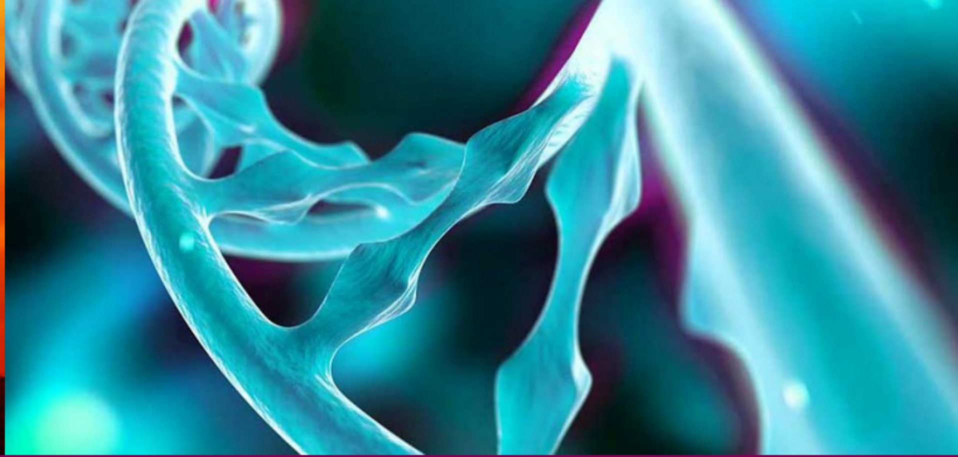
```
# hello-route.yaml
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  labels:
    app: hello-pod
  name: hello-route
spec:
  port:
    targetPort: 8888-8888
  to:
    kind: Service
    name: hello-service
    weight: 100
  wildcardPolicy: None
status: {}
```

Etherpad, A collaborative notepad application

This architecture uses everything we talked today about.

- MongoDB as database
- Persistence via Persistent Volumes
- Configuration of Etherpad with ConfigMap
- Database configuration via Secret object
  - Same Secret to configure the frontend (etherpad) and database
- Etherpad template





Coffee break II

🍩 🍵

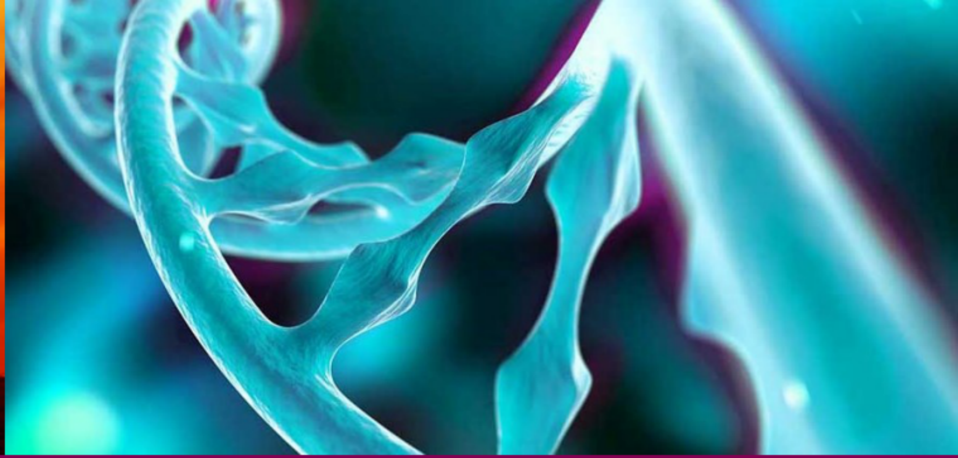
15 min

Exercises B

Go to the exercises page.

1. Add persistent storage python application
2. Add configuration python application
3. Execute a container in a pod
4. Create Service and Route

Advanced topics and exercises



CSC – Finnish expertise in ICT for research, education and public administration

EmptyDir

Temporary storage, how to set it up?

- Edit the API object, **Pod** or **Deployment**:
  - Under **spec** > **volumes**, add a new entry of type **emptyDir**.
  - Under **spec** > **containers** > **volumeMounts**, add an entry mounting the previously created volume into a path.

The first change tells Rahti to reserve a space in the node, the second says where to mount it in the container.

```
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

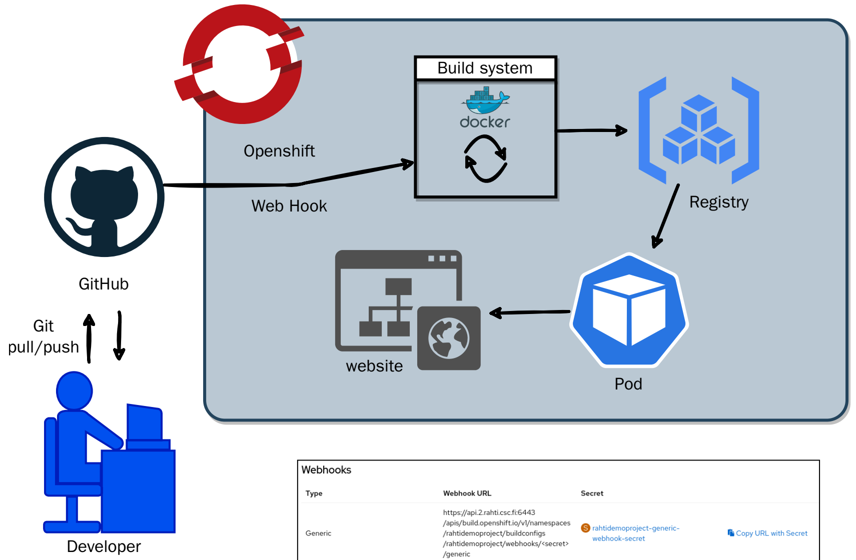
Webhooks

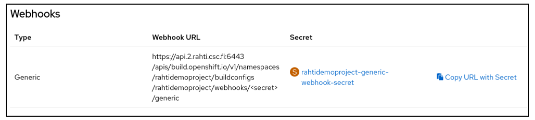
📜 A user-defined callback over HTTP. A mechanism in which an application (ex. GitHub) uses HTTP to notify another independent application (ex. Rahti).

- Go to the builds page
- Select the build you want to notify
- Scroll down until **Webhook URL**.

Click in 📋 **Copy URL with Secret**

Paste it in the repository's Webhook section.

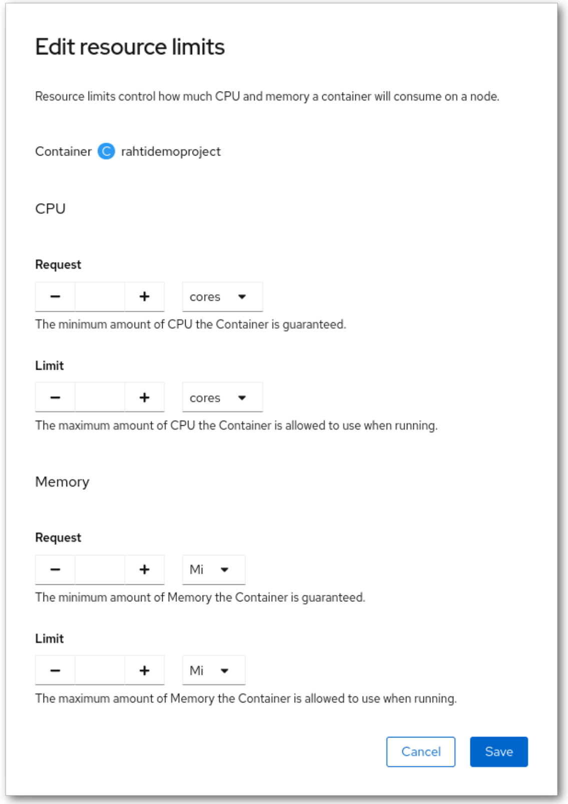




Webhooks

Type	Webhook URL	Secret	
Generic	https://api.2.rahti.csc.fi:6443 /apis/build.openshift.io/v1/namespaces /rahtidemoproject/buildconfigs /rahtidemoproject/webhooks/<secret> /generic	S rahtidemoproject-generic-webhook-secret	Copy URL with Secret

Resource Limits


Edit resource limits

Resource limits control how much CPU and memory a container will consume on a node.

Container rahtidemoproject

CPU

Request

cores

The minimum amount of CPU the Container is guaranteed.

Limit

cores

The maximum amount of CPU the Container is allowed to use when running.

Memory

Request

Mi

The minimum amount of Memory the Container is guaranteed.

Limit

Mi

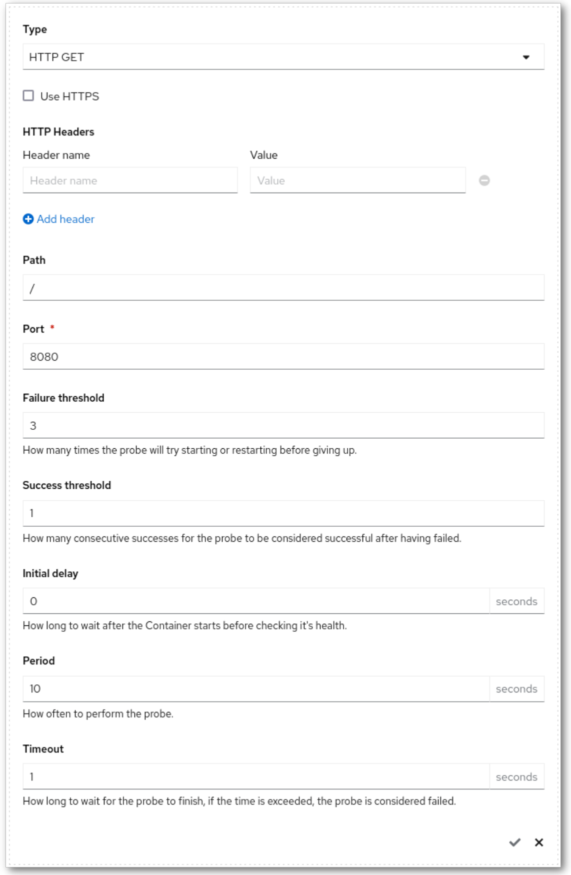
The maximum amount of Memory the Container is allowed to use when running.

In the `Deployment` page. `Actions > Edit Resource Limits`

Makes sure that the application will have, at a minimum, the requested CPU and memory.

- CPU, **prevents** the application to use more than the limit
- Memory, **kills** the application if it uses more than the limit

Health Checks (Probes)



Type
HTTP GET

☐ Use HTTPS

HTTP Headers

Header name	Value
Header name	Value

Add header

Path
/

Port *
8080

Failure threshold
3
How many times the probe will try starting or restarting before giving up.

Success threshold
1
How many consecutive successes for the probe to be considered successful after having failed.

Initial delay
0 seconds
How long to wait after the Container starts before checking it's health.

Period
10 seconds
How often to perform the probe.

Timeout
1 seconds
How long to wait for the probe to finish, if the time is exceeded, the probe is considered failed.

Health checks are highly recommended for all production loads. In the `Deployment` page. `Actions > Edit Health Checks`

- Kinds:
  - Readiness, has the application started yet?
  - Liveness, is the application alive?
- Types:
  - HTTP GET
  - Container Command
  - TCP socket
- Initial Delay
- Timeout

Exercises C (Extra)

Go to the exercises page.

1. Temporary storage
2. Webhook to trigger rebuild
3. Out of memory killer OOM
4. Probes

Note: You may as well repeat any exercise (from A or B), but using only the command line now.

Documentation Links

- The Rahti main page: rahti.csc.fi
- These slides: https://rahti-course.a3s.fi/basic.html
- These slides in PDF: https://rahti-course.a3s.fi/rahti-course-slides.pdf
- e-Lena Cloud computing fundamentals course
  - Enrolment key: `cloudcomputing`.
- Rahti documentation: docs.csc.fi
- Command line tools
- External documentation
  - Kubernetes documentation: kubernetes.io/docs/home
  - OpenShift documentation: docs.okd.io

Accounts:

- Create CSC account
- Rahti access

Contact Us

If you have any problem, request, or you just need more information:

servicedesk@csc.fi

C S C

Cloud solutions team:

- Alvaro Gonzalez, Alvaro.Gonzalez@csc.fi
- Tristan Perard, Tristan.Perard@csc.fi
- Jemal Tahir, Jemal.Tahir@csc.fi
- Joona Tolonen, Joona.Tolonen@csc.fi

https://facebook.com/CSCfi

https://twitter.com/CSCfi

https://www.youtube.com/c/CSCfi

https://www.linkedin.com/company/csc--it-center-for-science