uMEC for Akraino Hackathon

November 12, 2019

Tapio Tallgren

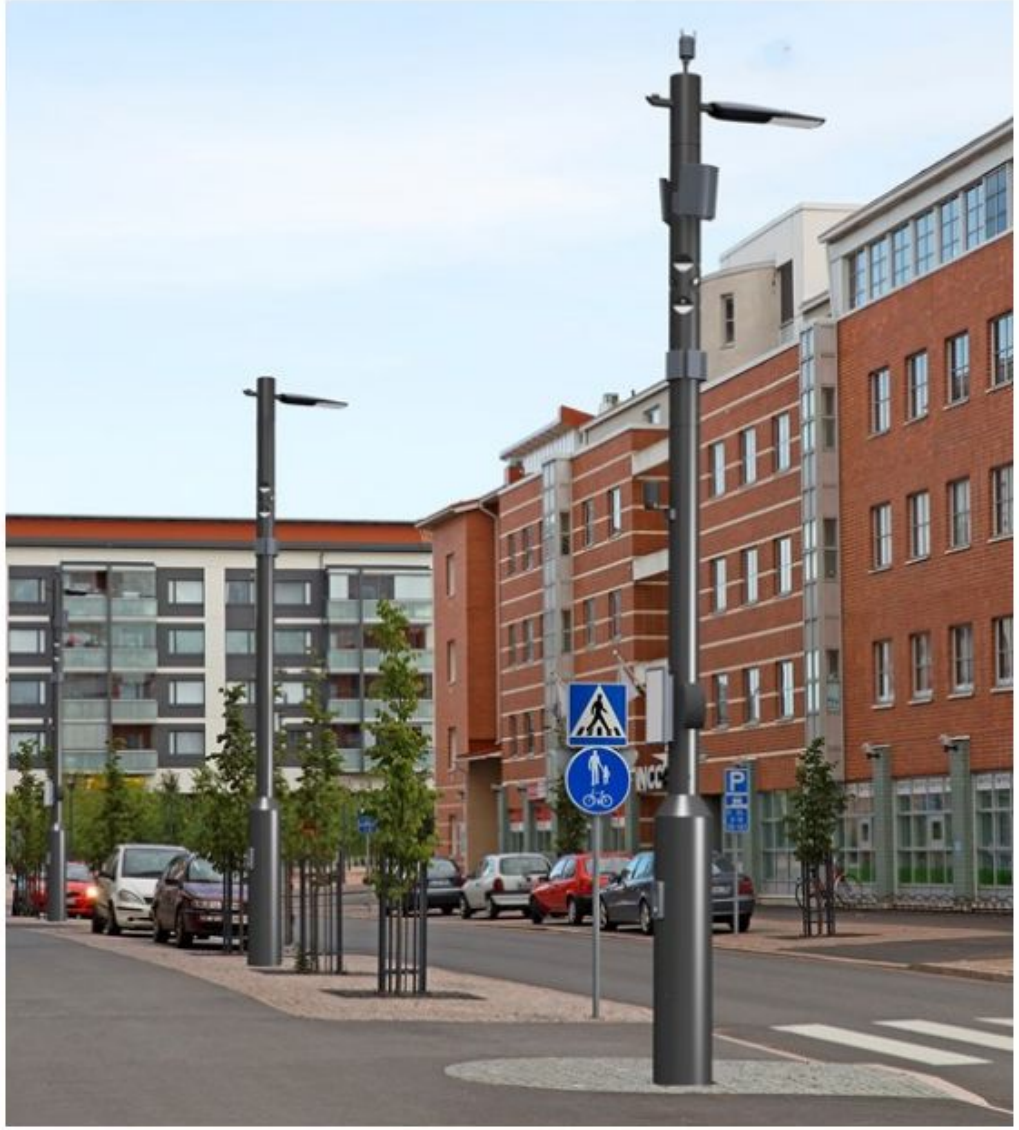
Ferenc Székely

AKRAINO EDGE STACK

What is a “Smart City”?

AKRAINO
EDGE STACK

LUXTURRIM5G SMART CITY ECOSYSTEM EXTENDS



ICT & Digitalization | Digitalization | Connected Intelligent Industries Finland

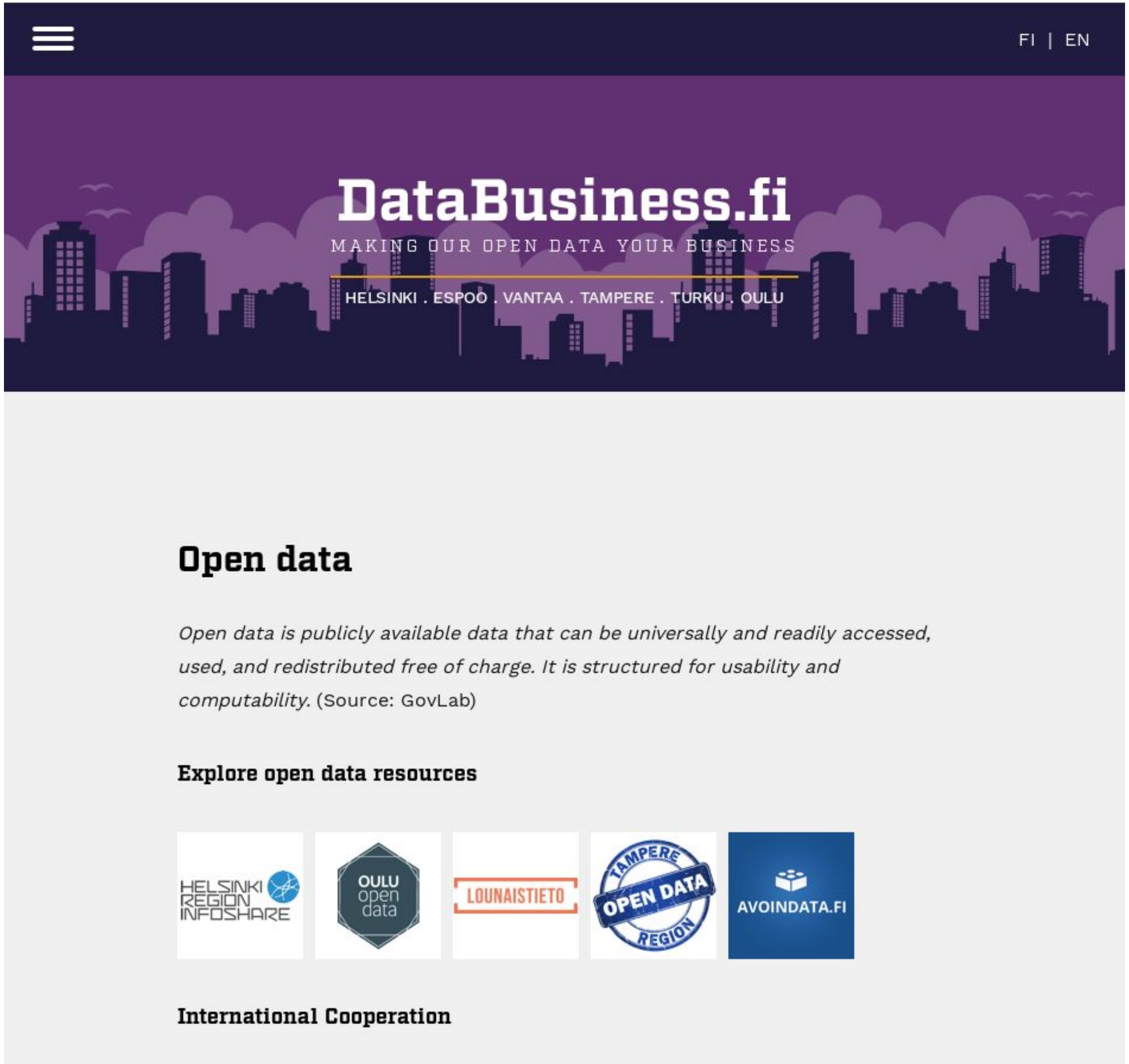
Smart Mobility Finland | Digital Trust Finland

The recently developed 5G smart pole concept in the LuxTurrim5G ecosystem is moving towards productization and practical piloting. Another goal is to create a platform utilizing a wide variety of data in a reliable and secured way and develop new digital services to meet real needs of cities.

Through a two-year and EUR 26 million funding, the group of 26 partners target the global smart city markets worth tens of billions euros in close collaboration. Business Finland provides innovation funding for the project.

The first phase of the LuxTurrim5G project, which ended in May, successfully developed the 5G smart pole concept, which integrates the 5G base station, weather and air quality sensors, video cameras, monitors electric vehicle charging unit and other active devices. The good results and the first pilots at the Nokia Campus in Espoo, Finland have attracted a lot of interest around the world and given the LuxTurrim5G ecosystem a boost for further expansion.

Leverage from the EU 2014–2020



FI | EN

DataBusiness.fi

MAKING OUR OPEN DATA YOUR BUSINESS

HELSINKI . ESPOO . VANTAA . TAMPERE . TURKU . OULU

Open data

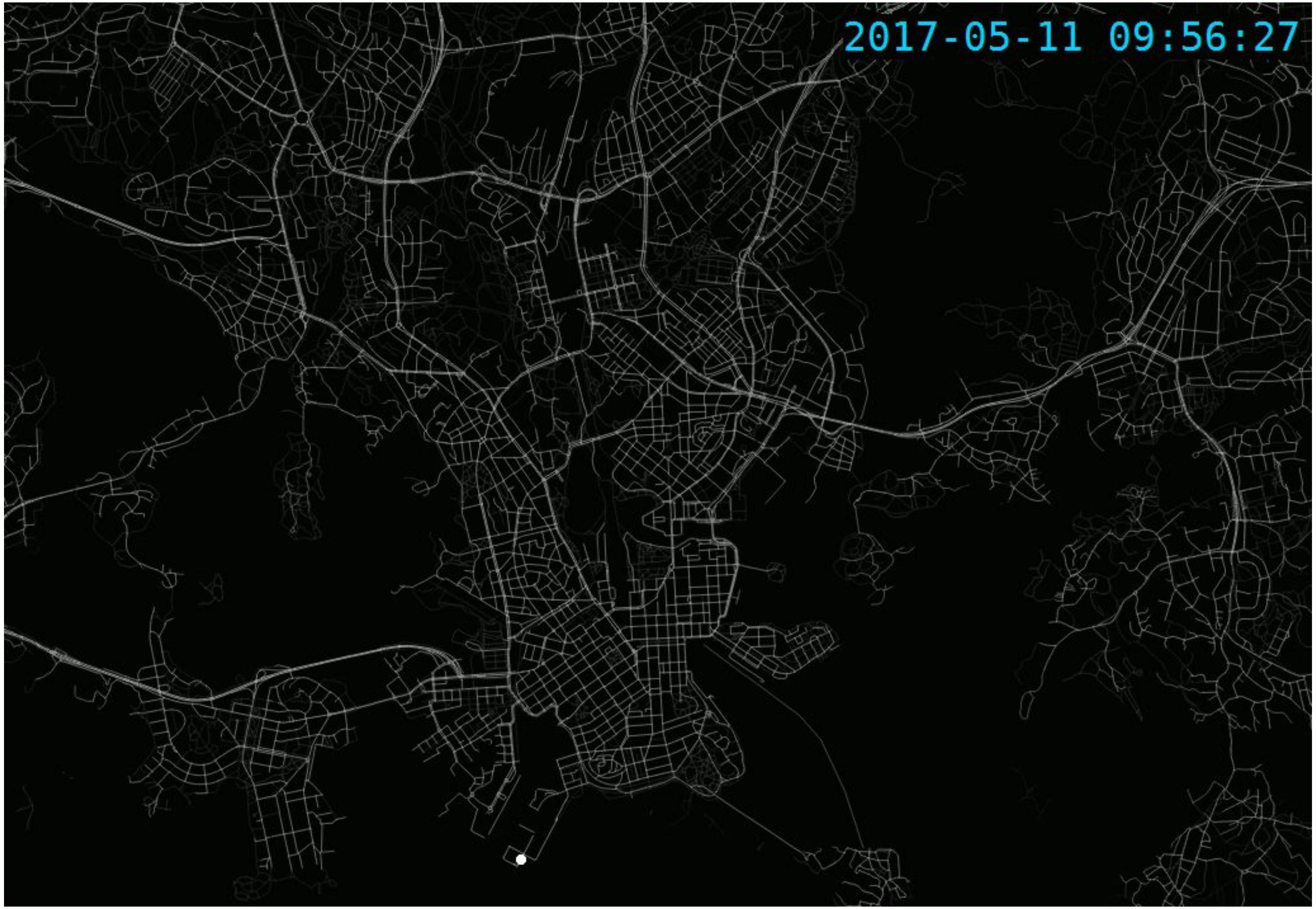
Open data is publicly available data that can be universally and readily accessed, used, and redistributed free of charge. It is structured for usability and computability. (Source: GovLab)

Explore open data resources

International Cooperation

AKRAINO EDGE STACK

Open Data

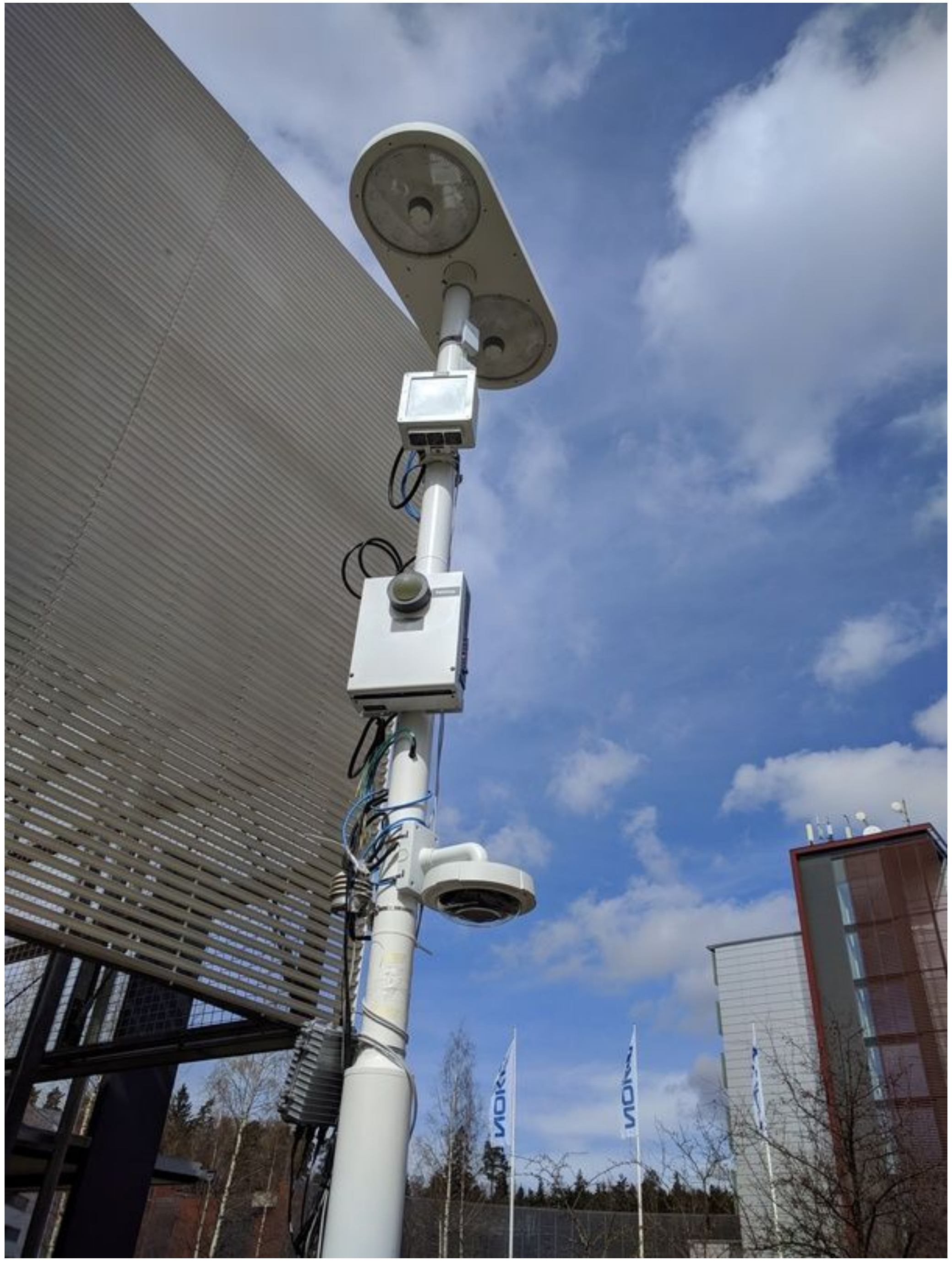


AKRAINO EDGE STACK

What is µMEC?

AKRAINO EDGE STACK

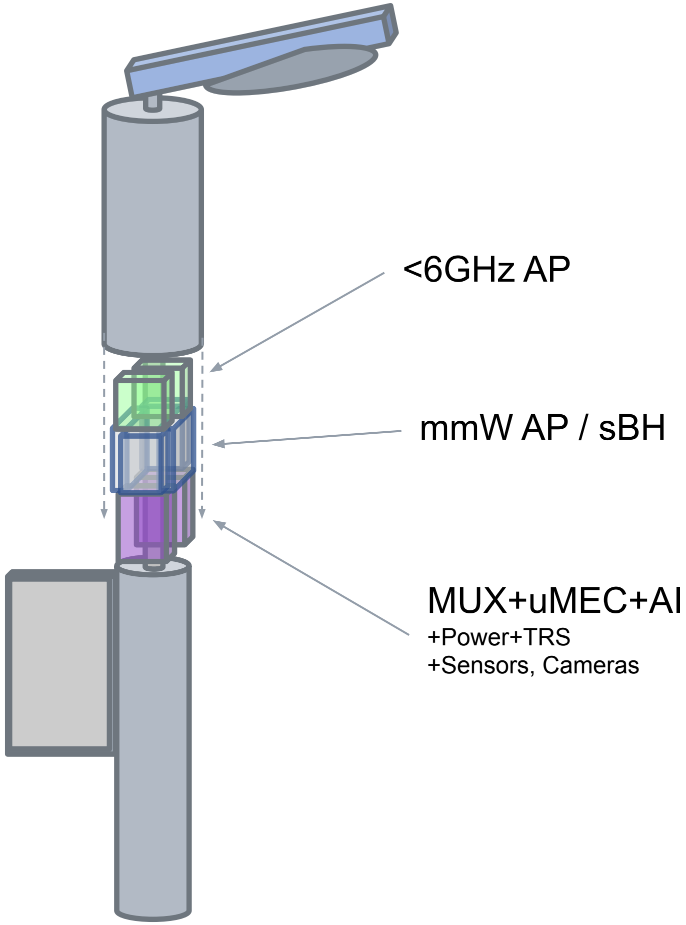
LuxTurrim5G light pole



AKRAINO EDGE STACK

μMEC concept

- μMEC complements the emerging 5G radio networks by enabling new applications
- μMEC is a small form factor HW+SW platform for especially the Smart City services on Ultra Far Edge
- It can use 5G, WLAN or fiber connection
- It can be installed on light poles, vehicles, etc
- The μMEC proof-of-concept is based on LuxTurrim5G and open source components



μMEC deployment example: LuxTurrim5G

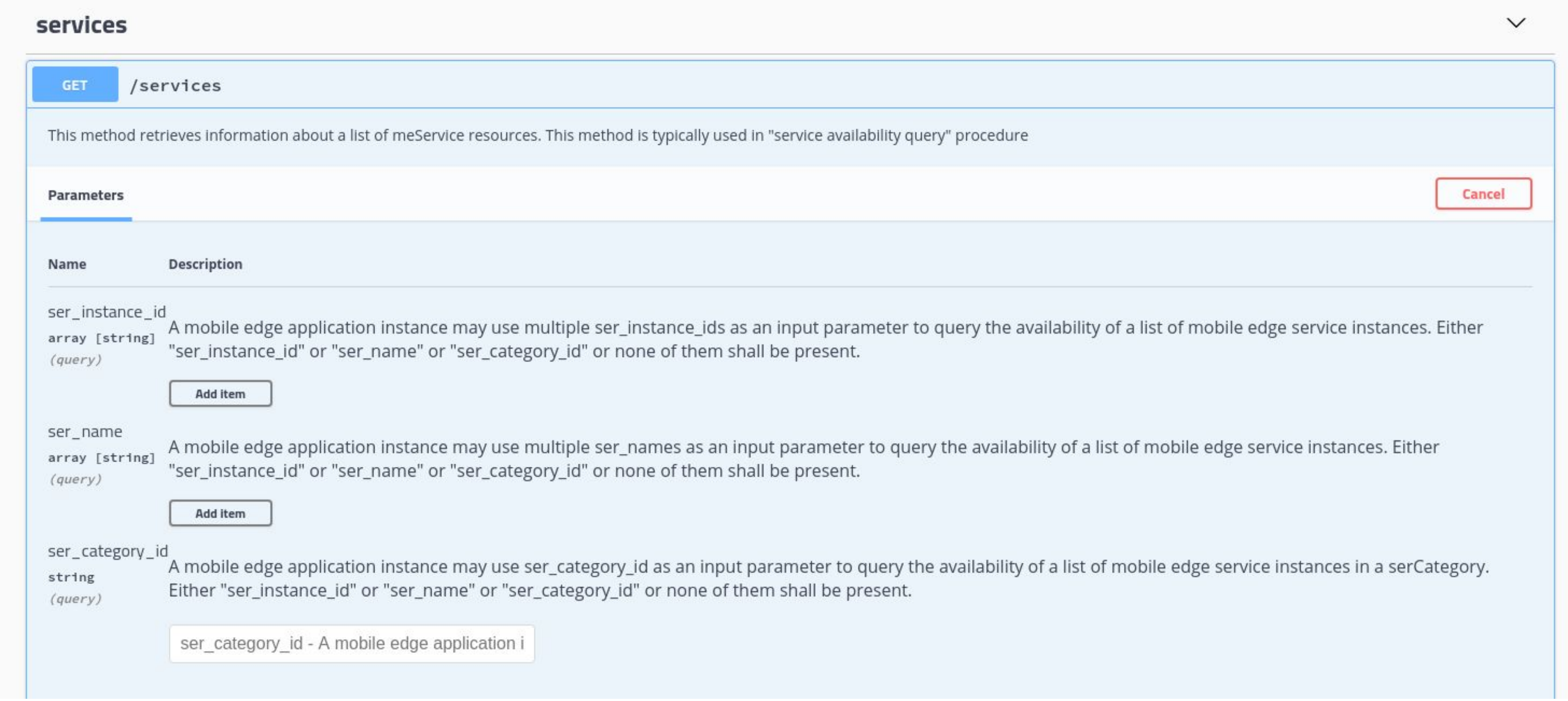
What is Multi-Access Edge Computing?

AKRAINO EDGE STACK

Multi-Access Edge Computing

- Standardized application development model for the Edge
- Interfaces are defined using OpenAPI that allows generating server and client stubs for tens of programming languages
- MEC-11 (Application Enablement) allows modifying traffic rules, DNS rules, and discovering new services
- Supports multiple transports, security with OAuth2.0 etc

ETSI MEC (Multi-Access Edge Computing)



/services

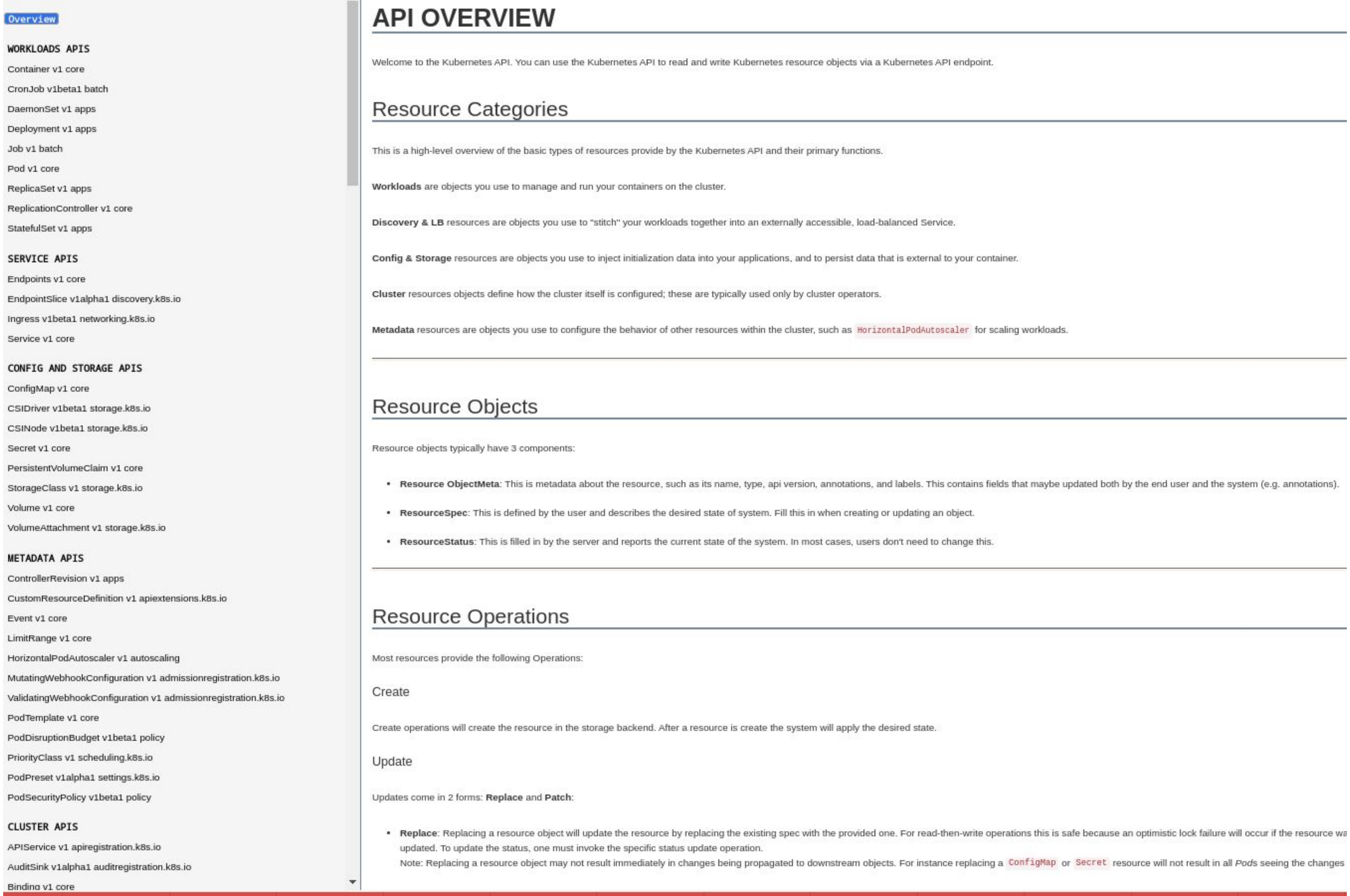
This method retrieves information about a list of meService resources. This method is typically used in "service availability query" procedure

Parameters

Name	Description
ser_instance_id array [string] *(query)*	A mobile edge application instance may use multiple ser_instance_ids as an input parameter to query the availability of a list of mobile edge service instances. Either "ser_instance_id" or "ser_name" or "ser_category_id" or none of them shall be present.
ser_name array [string] *(query)*	A mobile edge application instance may use multiple ser_names as an input parameter to query the availability of a list of mobile edge service instances. Either "ser_instance_id" or "ser_name" or "ser_category_id" or none of them shall be present.
ser_category_id string *(query)*	A mobile edge application instance may use ser_category_id as an input parameter to query the availability of a list of mobile edge service instances in a serCategory. Either "ser_instance_id" or "ser_name" or "ser_category_id" or none of them shall be present. ser_category_id - A mobile edge application i

What is Kubernetes?

AKRAINO EDGE STACK



Overview

WORKLOADS APIS

- Container v1 core
- CronJob v1beta1 batch
- DaemonSet v1 apps
- Deployment v1 apps
- Job v1 batch
- Pod v1 core
- ReplicaSet v1 apps
- ReplicationController v1 core
- StatefulSet v1 apps

SERVICE APIS

- Endpoints v1 core
- EndpointSlice v1alpha1 discovery.k8s.io
- Ingress v1beta1 networking.k8s.io
- Service v1 core

CONFIG AND STORAGE APIS

- ConfigMap v1 core
- CSIDriver v1beta1 storage.k8s.io
- CSINode v1beta1 storage.k8s.io
- Secret v1 core
- PersistentVolumeClaim v1 core
- StorageClass v1 storage.k8s.io
- Volume v1 core
- VolumeAttachment v1 storage.k8s.io

METADATA APIS

- ControllerRevision v1 apps
- CustomResourceDefinition v1 apiextensions.k8s.io
- Event v1 core
- LimitRange v1 core
- HorizontalPodAutoscaler v1 autoscaling
- MutatingWebhookConfiguration v1 admissionregistration.k8s.io
- ValidatingWebhookConfiguration v1 admissionregistration.k8s.io
- PodTemplate v1 core
- PodDisruptionBudget v1beta1 policy
- PriorityClass v1 scheduling.k8s.io
- PodPreset v1alpha1 settings.k8s.io
- PodSecurityPolicy v1beta1 policy

CLUSTER APIS

- APIService v1 apiregistration.k8s.io
- AuditSink v1alpha1 auditregistration.k8s.io
- Binding v1 core

API OVERVIEW

Welcome to the Kubernetes API. You can use the Kubernetes API to read and write Kubernetes resource objects via a Kubernetes API endpoint.

Resource Categories

This is a high-level overview of the basic types of resources provide by the Kubernetes API and their primary functions.

Workloads are objects you use to manage and run your containers on the cluster.

Discovery & LB resources are objects you use to "stitch" your workloads together into an externally accessible, load-balanced Service.

Config & Storage resources are objects you use to inject initialization data into your applications, and to persist data that is external to your container.

Cluster resources objects define how the cluster itself is configured; these are typically used only by cluster operators.

Metadata resources are objects you use to configure the behavior of other resources within the cluster, such as `HorizontalPodAutoscaler` for scaling workloads.

Resource Objects

Resource objects typically have 3 components:

- **Resource ObjectMeta**: This is metadata about the resource, such as its name, type, api version, annotations, and labels. This contains fields that maybe updated both by the end user and the system (e.g. annotations).
- **ResourceSpec**: This is defined by the user and describes the desired state of system. Fill this in when creating or updating an object.
- **ResourceStatus**: This is filled in by the server and reports the current state of the system. In most cases, users don't need to change this.

Resource Operations

Most resources provide the following Operations:

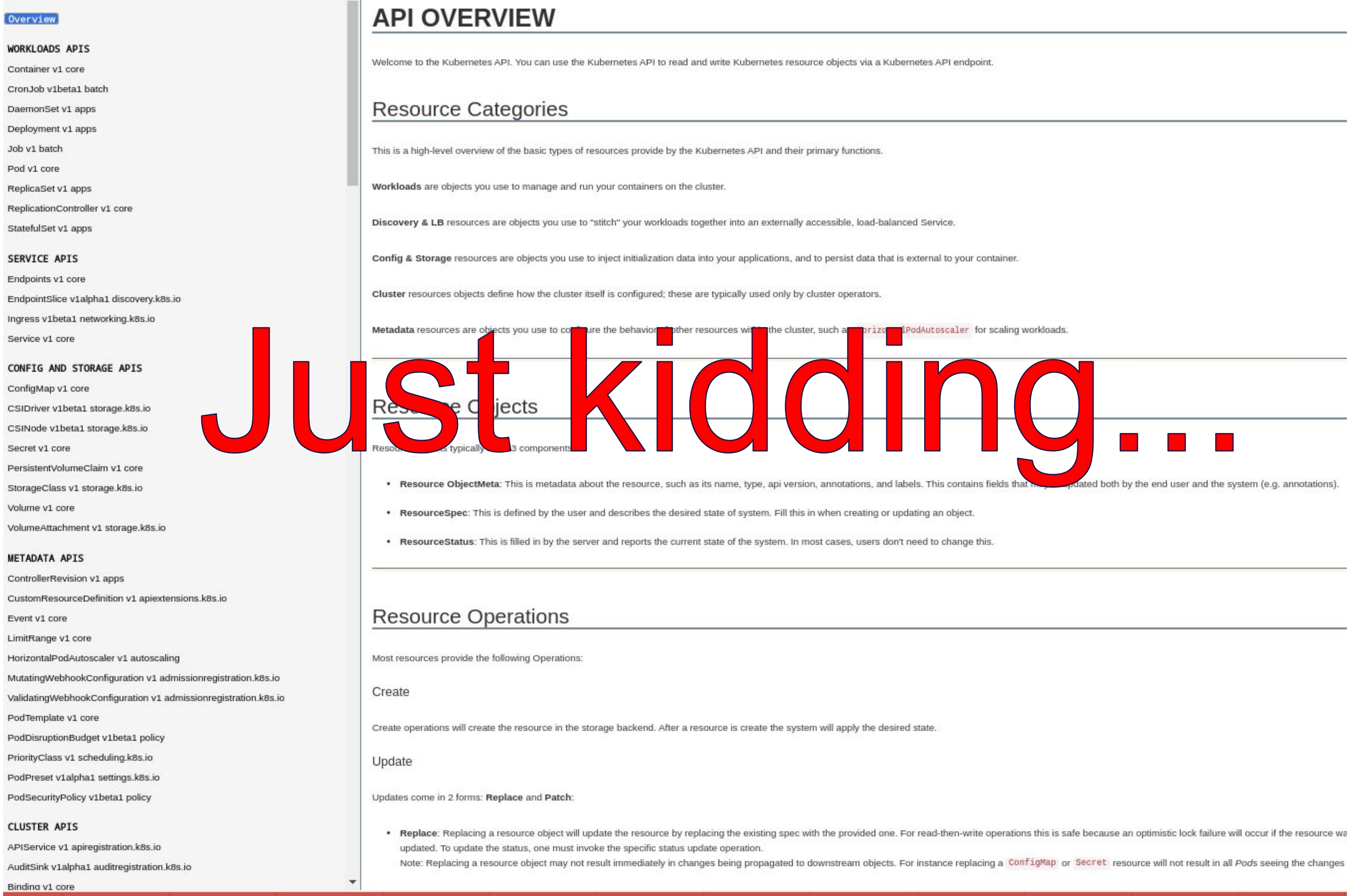
Create

Create operations will create the resource in the storage backend. After a resource is create the system will apply the desired state.

Update

Updates come in 2 forms: **Replace** and **Patch**:

- **Replace**: Replacing a resource object will update the resource by replacing the existing spec with the provided one. For read-then-write operations this is safe because an optimistic lock failure will occur if the resource wa updated. To update the status, one must invoke the specific status update operation.
  Note: Replacing a resource object may not result immediately in changes being propagated to downstream objects. For instance replacing a `ConfigMap` or `Secret` resource will not result in all *Pods* seeing the changes



Just kidding...

Overview

WORKLOADS APIS

- Container v1 core
- CronJob v1beta1 batch
- DaemonSet v1 apps
- Deployment v1 apps
- Job v1 batch
- Pod v1 core
- ReplicaSet v1 apps
- ReplicationController v1 core
- StatefulSet v1 apps

SERVICE APIS

- Endpoints v1 core
- EndpointSlice v1alpha1 discovery.k8s.io
- Ingress v1beta1 networking.k8s.io
- Service v1 core

CONFIG AND STORAGE APIS

- ConfigMap v1 core
- CSIDriver v1beta1 storage.k8s.io
- CSINode v1beta1 storage.k8s.io
- Secret v1 core
- PersistentVolumeClaim v1 core
- StorageClass v1 storage.k8s.io
- Volume v1 core
- VolumeAttachment v1 storage.k8s.io

METADATA APIS

- ControllerRevision v1 apps
- CustomResourceDefinition v1 apiextensions.k8s.io
- Event v1 core
- LimitRange v1 core
- HorizontalPodAutoscaler v1 autoscaling
- MutatingWebhookConfiguration v1 admissionregistration.k8s.io
- ValidatingWebhookConfiguration v1 admissionregistration.k8s.io
- PodTemplate v1 core
- PodDisruptionBudget v1beta1 policy
- PriorityClass v1 scheduling.k8s.io
- PodPreset v1alpha1 settings.k8s.io
- PodSecurityPolicy v1beta1 policy

CLUSTER APIS

- APIService v1 apiregistration.k8s.io
- AuditSink v1alpha1 auditregistration.k8s.io
- Binding v1 core

API OVERVIEW

Welcome to the Kubernetes API. You can use the Kubernetes API to read and write Kubernetes resource objects via a Kubernetes API endpoint.

Resource Categories

This is a high-level overview of the basic types of resources provide by the Kubernetes API and their primary functions.

Workloads are objects you use to manage and run your containers on the cluster.

Discovery & LB resources are objects you use to "stitch" your workloads together into an externally accessible, load-balanced Service.

Config & Storage resources are objects you use to inject initialization data into your applications, and to persist data that is external to your container.

Cluster resources objects define how the cluster itself is configured; these are typically used only by cluster operators.

Metadata resources are objects you use to co[illegible]ure the behavior [illegible] other resources wi[illegible] the cluster, such a[illegible] `[illegible]rizo[illegible]lPodAutoscaler` for scaling workloads.

Res[illegible]e O[illegible]jects

Resou[illegible] typically [illegible] 3 components:

- **Resource ObjectMeta**: This is metadata about the resource, such as its name, type, api version, annotations, and labels. This contains fields that [illegible] updated both by the end user and the system (e.g. annotations).
- **ResourceSpec**: This is defined by the user and describes the desired state of system. Fill this in when creating or updating an object.
- **ResourceStatus**: This is filled in by the server and reports the current state of the system. In most cases, users don't need to change this.

Resource Operations

Most resources provide the following Operations:

Create

Create operations will create the resource in the storage backend. After a resource is create the system will apply the desired state.

Update

Updates come in 2 forms: **Replace** and **Patch**:

- **Replace**: Replacing a resource object will update the resource by replacing the existing spec with the provided one. For read-then-write operations this is safe because an optimistic lock failure will occur if the resource wa
  updated. To update the status, one must invoke the specific status update operation.
  Note: Replacing a resource object may not result immediately in changes being propagated to downstream objects. For instance replacing a `ConfigMap` or `Secret` resource will not result in all *Pods* seeing the changes

Putting it all together: your task is to create a Smart City service that uses a µMEC that is deployed in the city

For simplicity, we have a ready-made application as a starting point.

Your solution can include elements outside this app

AKRAINO EDGE STACK

Your city...

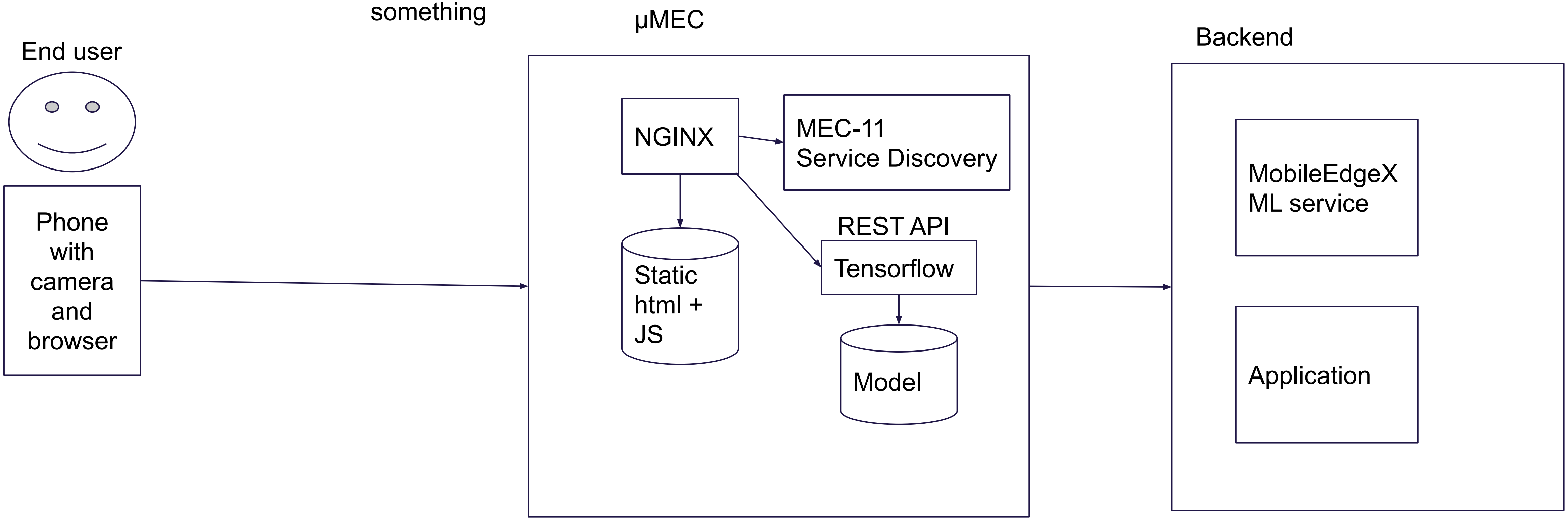
1. End user will use a browser to access the service

2. Web server will return a JavaScript page

3. JavaScript will use the camera to take a picture

4. The web server will use MEC-11 to find a Tensorflow service to analyze the image

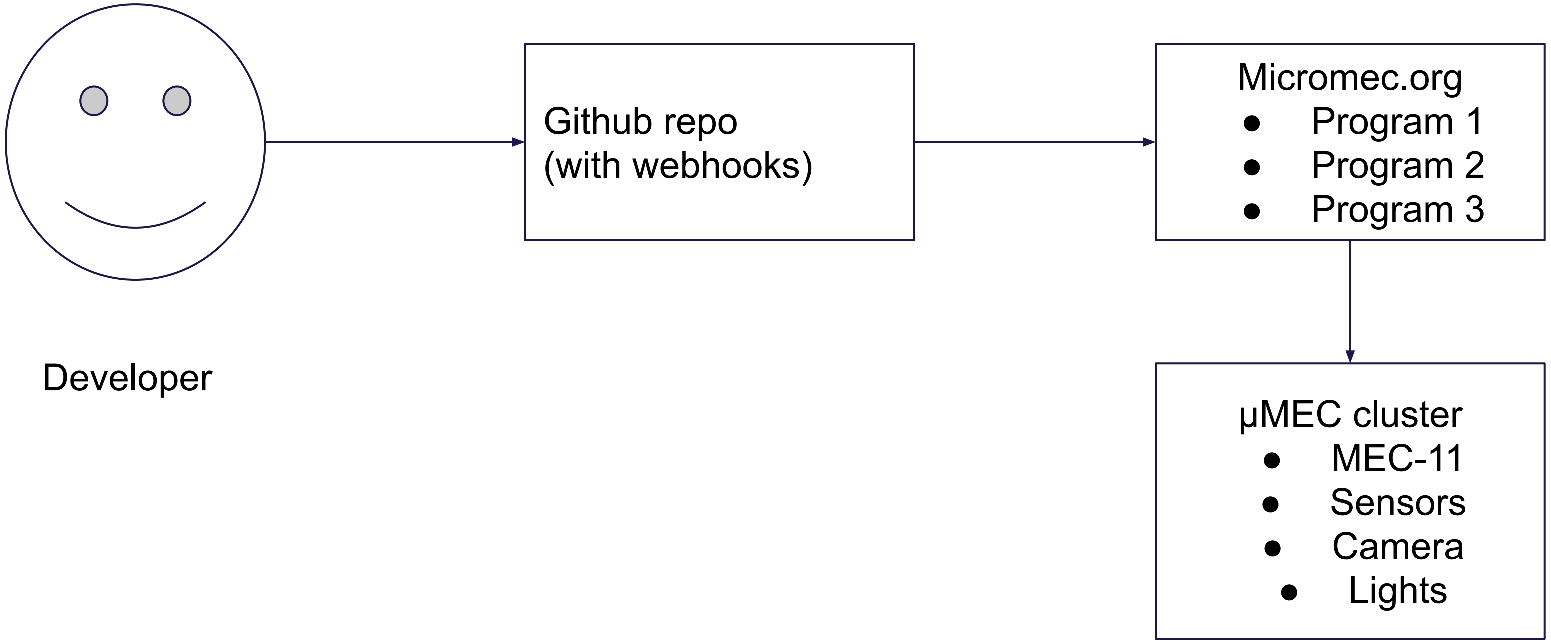
5. The web server will use the results of the analysis and do something



How it works

- You need to bring in your own laptop and smart phone
- You will be able to download the sample code from Akraino gerrit and see how it runs on micromec.org/hack/selfie
- To modify and run the modified code,
  - clone it to your own Github repo
  - enable webhooks on the repo
- Your app will be served on a web server with a public URL
- You can also modify the Tensorflow model that the sample app uses. Please talk to us, if you want to do that.

Development workflow



Developer

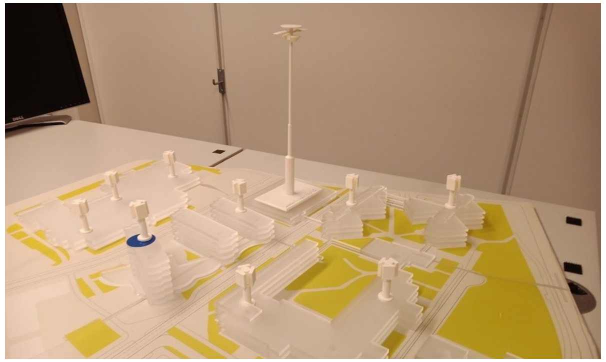
Github repo
(with webhooks)

Micromec.org

- Program 1
- Program 2
- Program 3

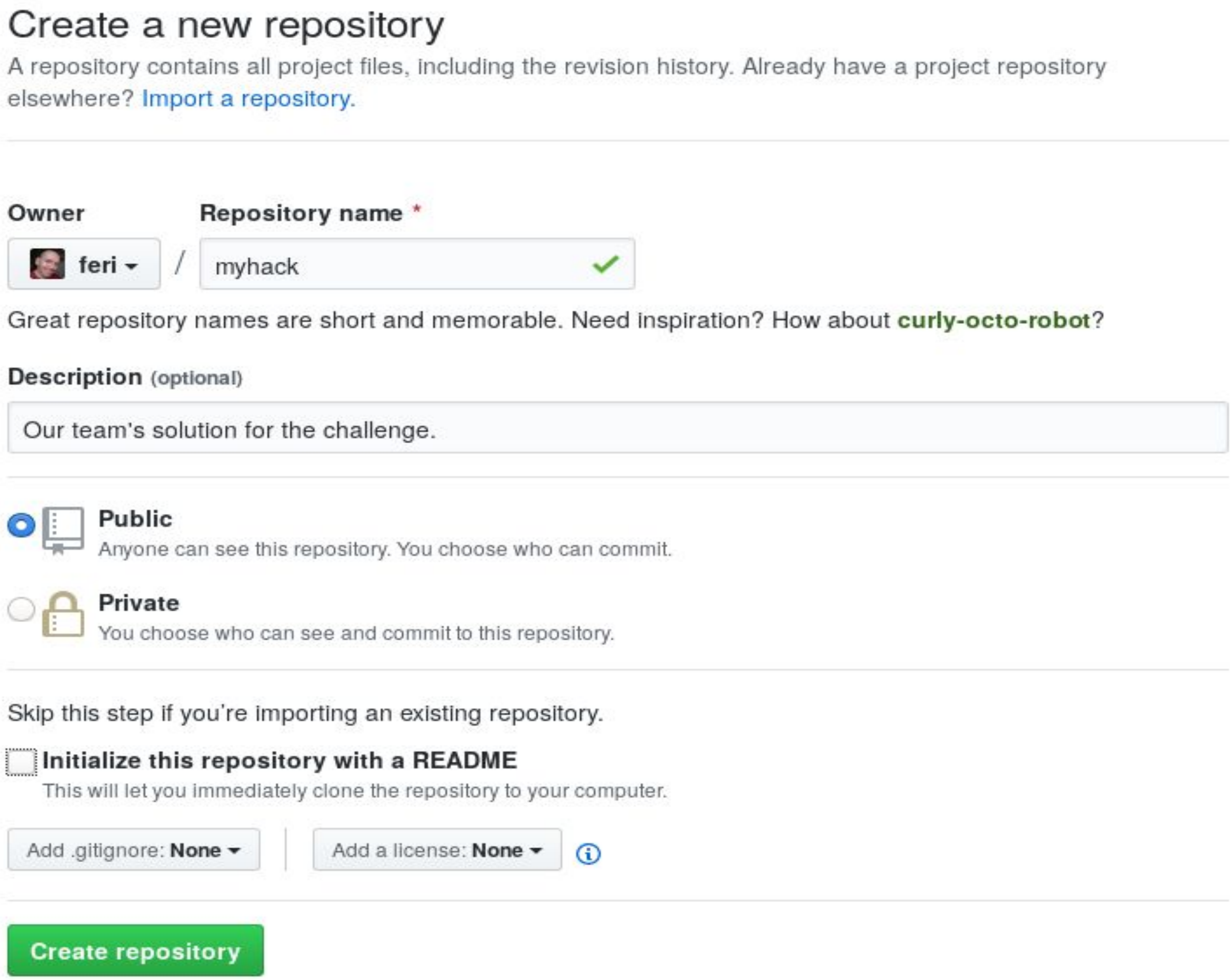
µMEC cluster

- MEC-11
- Sensors
- Camera
- Lights



Create a new github repo

Click: https://github.com/new



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: feri / Repository name *: myhack ✓

Great repository names are short and memorable. Need inspiration? How about **curly-octo-robot**?

Description (optional)

Our team's solution for the challenge.

● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

□ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None**

Add a webhook to the repository

Payload URL:

https://micromec.org/hack

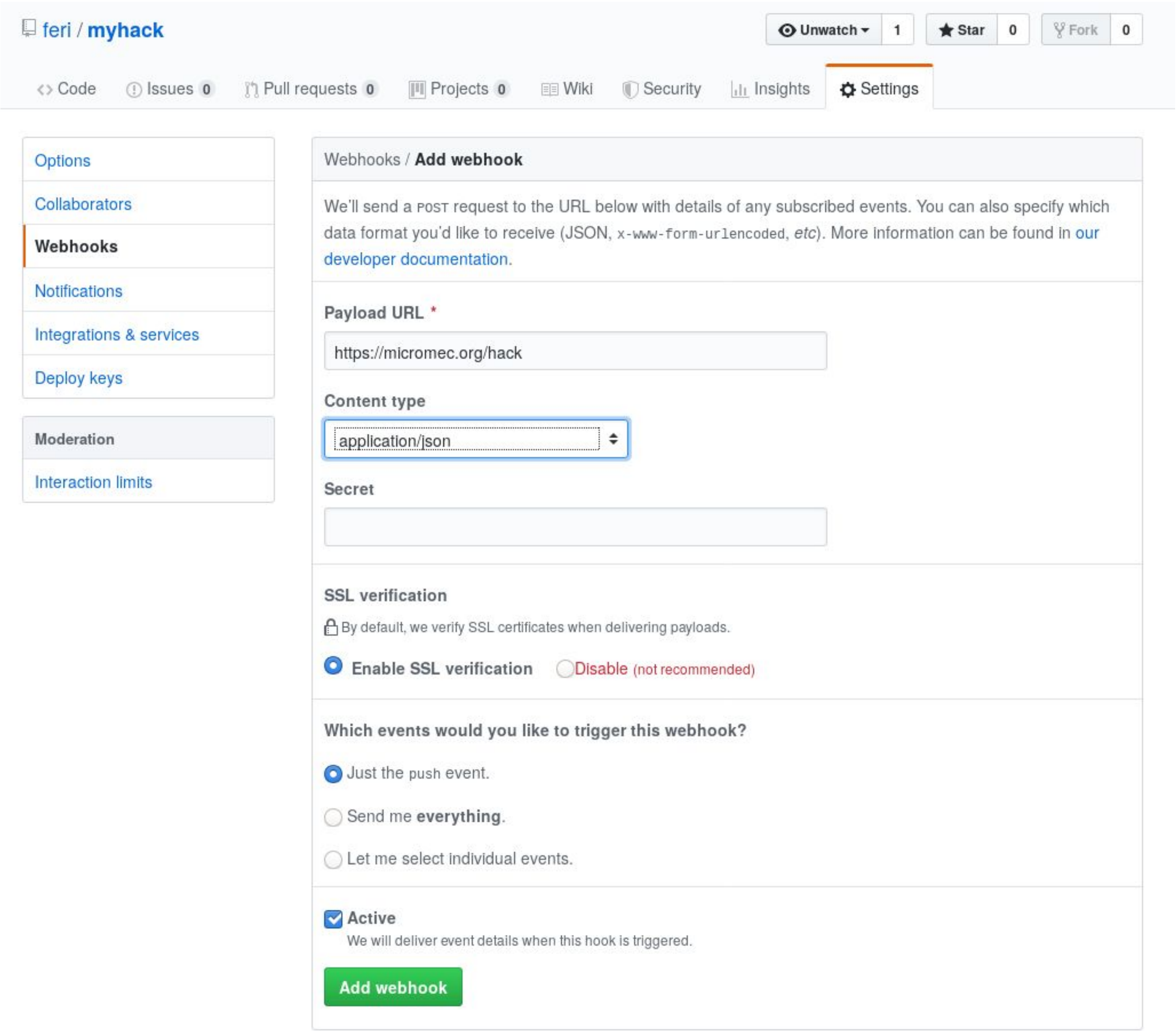
Content type:

application/json

Secret: leave it **blank**

SSL verification: **Enable**

Events: just the **push** event



Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://micromec.org/hack

Content type

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

● **Enable SSL verification** ○ Disable (not recommended)

Which events would you like to trigger this webhook?

● Just the push event.

○ Send me **everything**.

○ Let me select individual events.

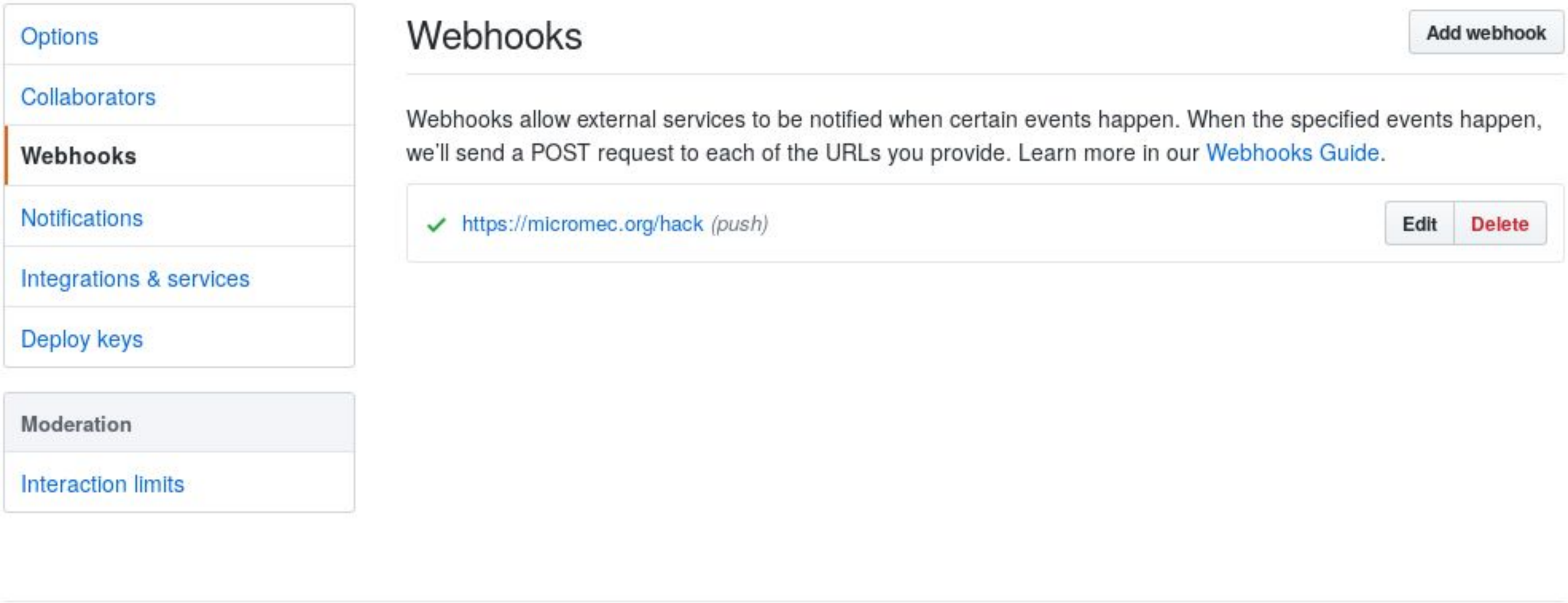
☑ **Active**

We will deliver event details when this hook is triggered.

Add webhook

AKRAINO EDGE STACK

Webhook is now active



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

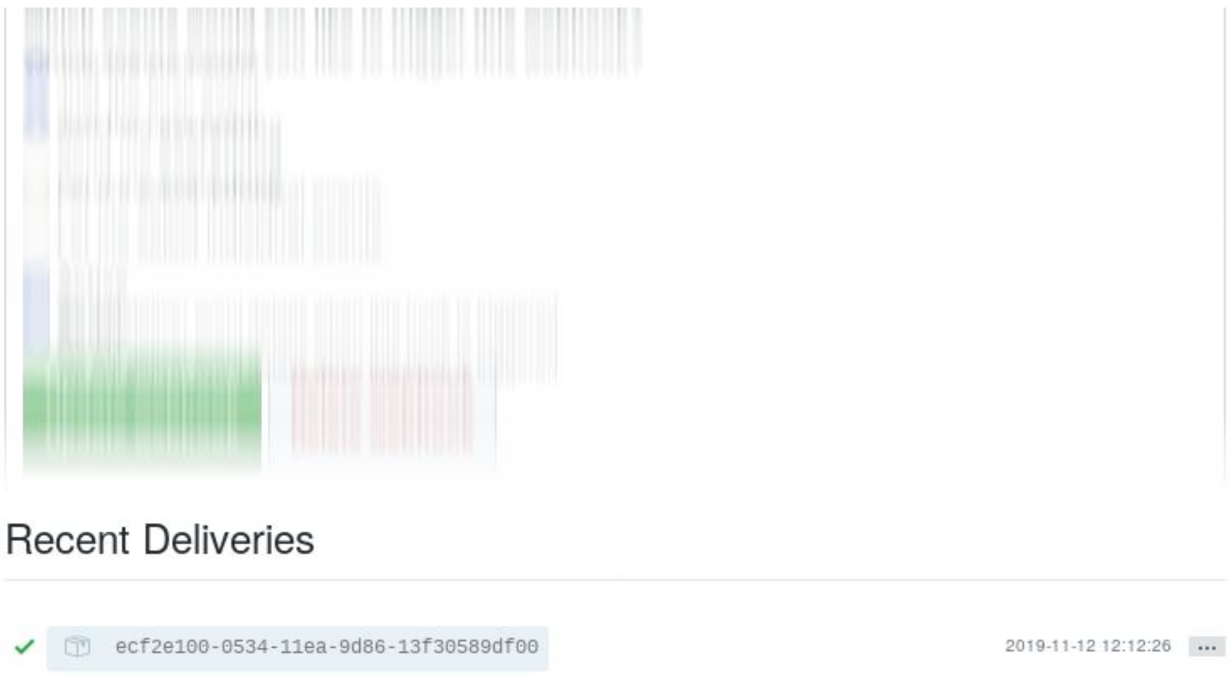
✓ https://micromec.org/hack *(push)*

Back to webhook

Check **Recent Deliveries** on the webhook’s settings page.

The delivery from the webhook is marked with a **green** tick mark.

If the mark is red then please **contact** the hackathon **organizers** online.



Recent Deliveries

✓ ecf2e100-0534-11ea-9d86-13f30589df00 2019-11-12 12:12:26 ...

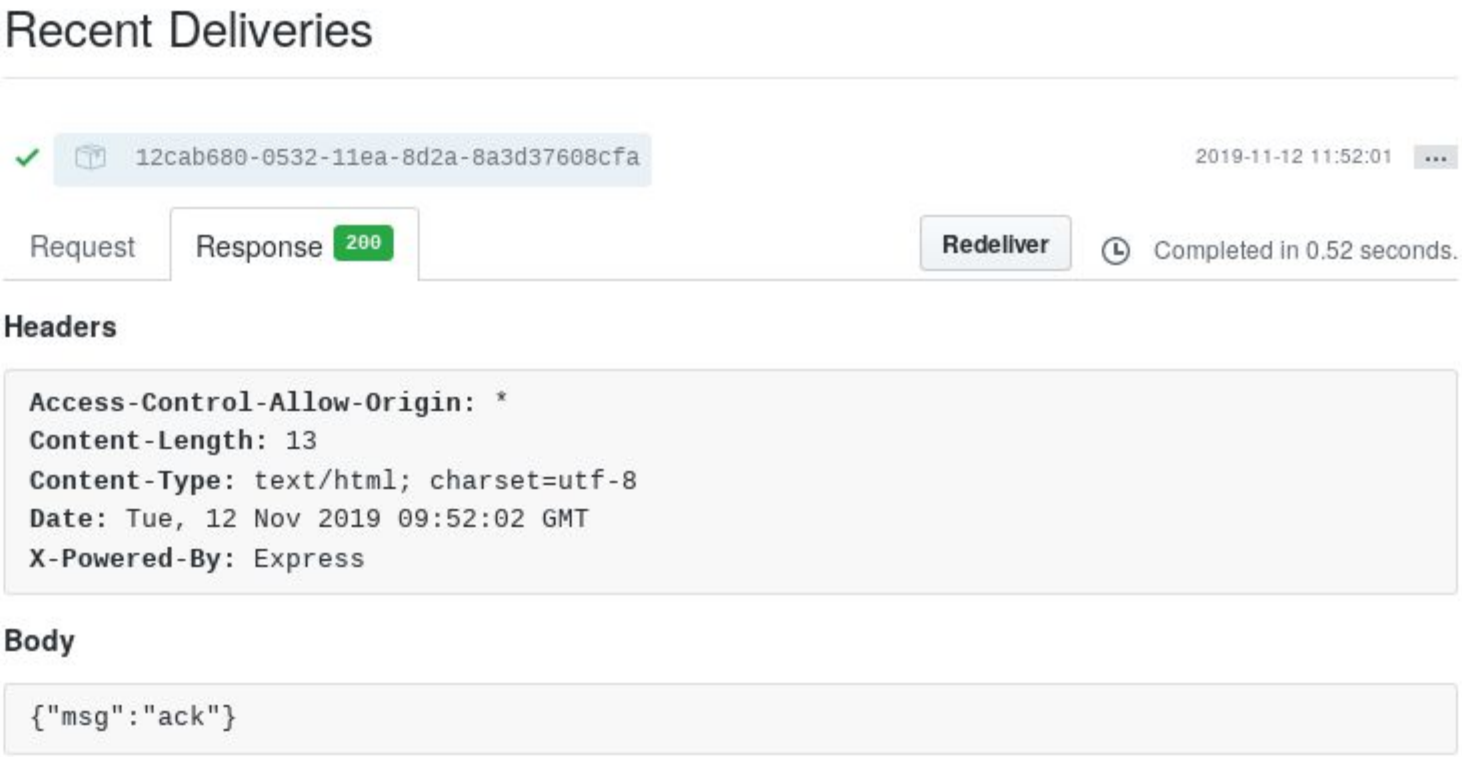
Recent deliveries

Response should be

HTTP **200.**

Click **Redeliver**

to test the hook.



Recent Deliveries

✓ 12cab680-0532-11ea-8d2a-8a3d37608cfa 2019-11-12 11:52:01 ...

Completed in 0.52 seconds.

Headers

```
Access-Control-Allow-Origin: *
Content-Length: 13
Content-Type: text/html; charset=utf-8
Date: Tue, 12 Nov 2019 09:52:02 GMT
X-Powered-By: Express
```

Body

```
{"msg":"ack"}
```

AKRAINO EDGE STACK

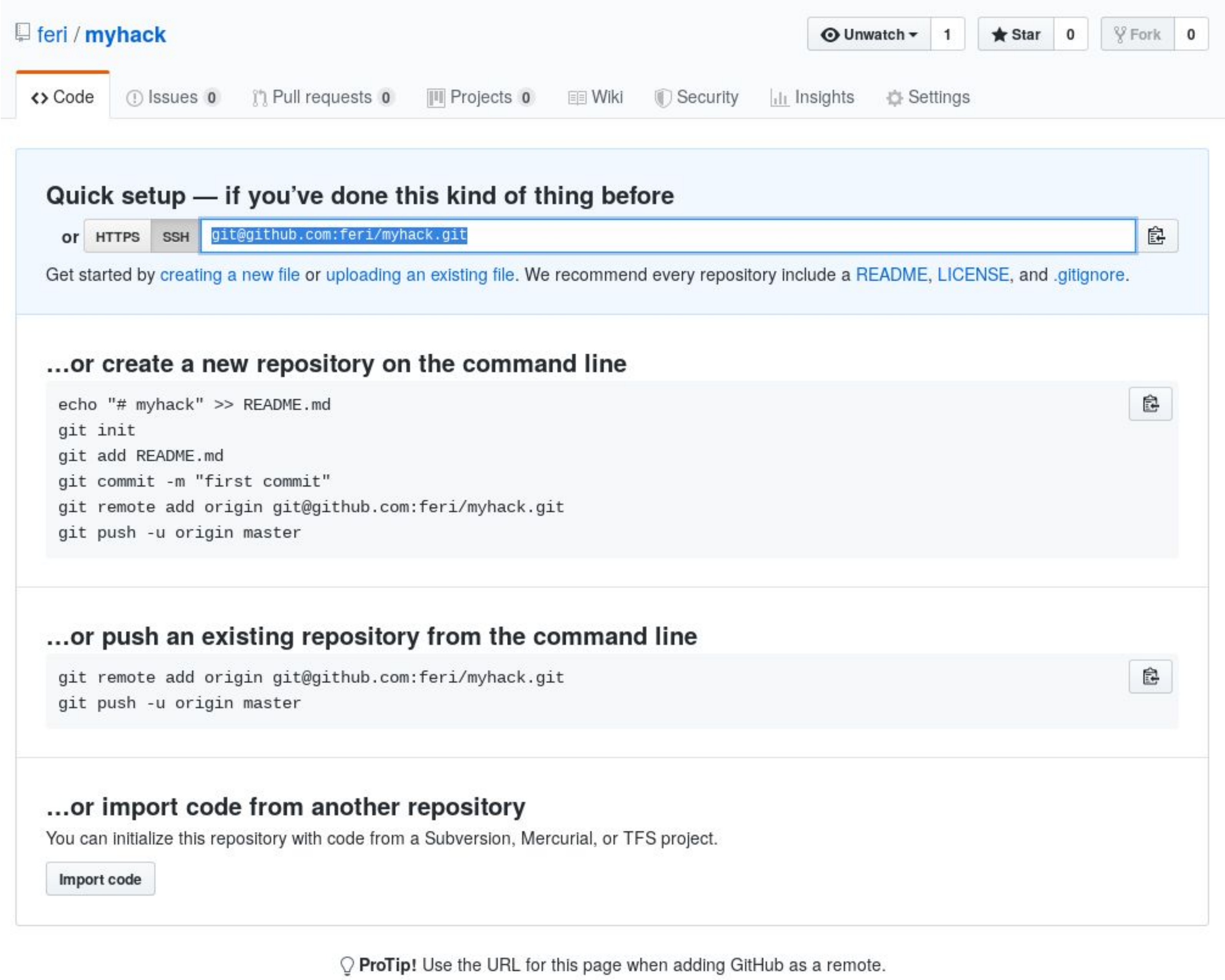
Local setup

Follow instructions from Github.

“Business as usual.”

Initiate your local copy of the repository and make the first **commit** and **push**.

AKRAINO EDGE STACK



Quick setup — if you’ve done this kind of thing before

or HTTPS SSH `git@github.com:feri/myhack.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# myhack" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:feri/myhack.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:feri/myhack.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

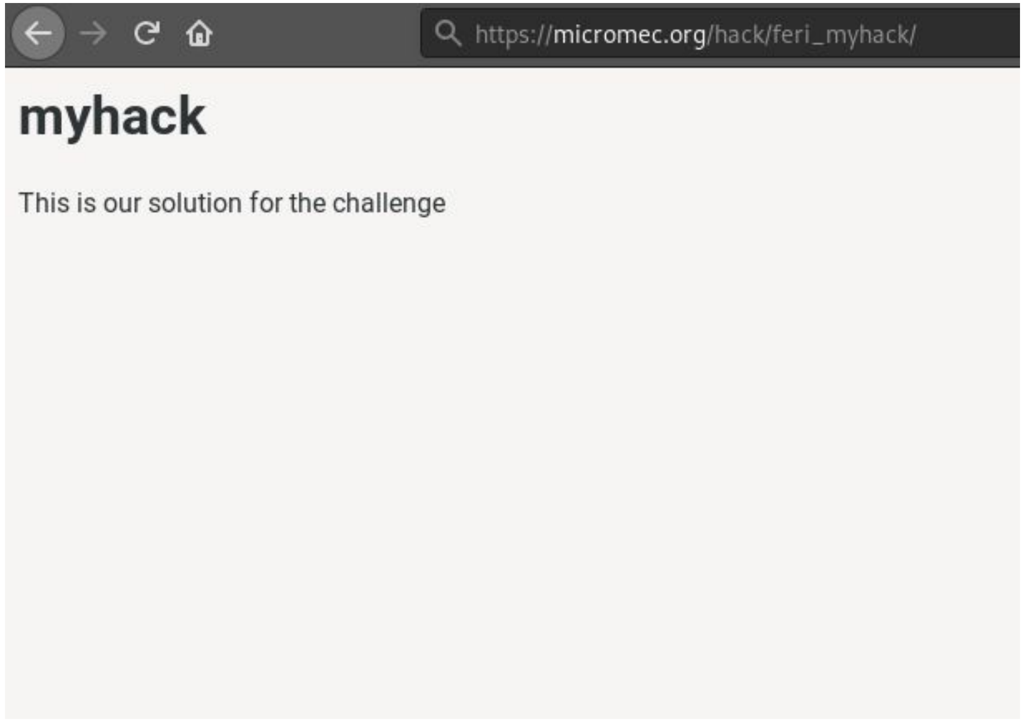
Check the result

We pull and deploy your code to micromec.org.

See the list of projects: https://micromec.org/hack

Our server will host standard **HTML**, **JS** and **CSS**. We also allow small images (max. 1M).

Please do not push inappropriate content!



https://micromec.org/hack/feri_myhack/

myhack

This is our solution for the challenge

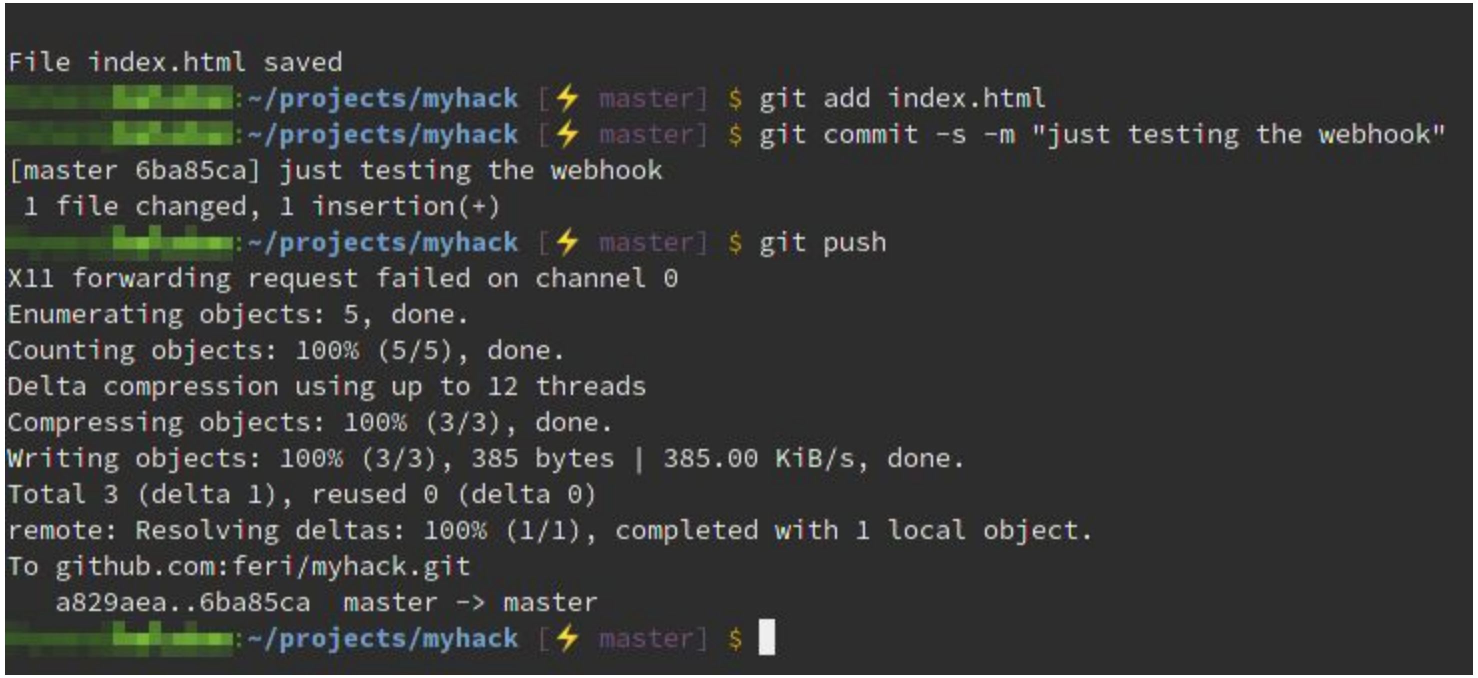
AKRAINO EDGE STACK

Develop your app

“Commit early, commit often!”

Work on your project and push changes from the command line (for instance).

Github will trigger a new deployment on micromec.org.



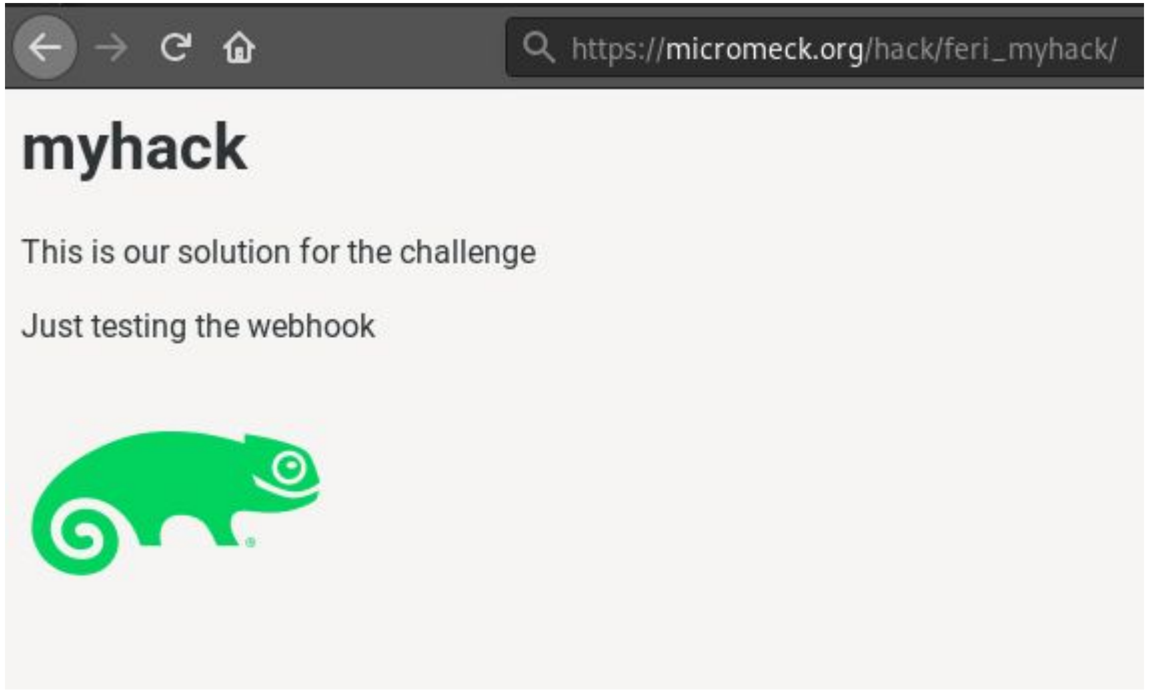
```
File index.html saved
[redacted]:~/projects/myhack [⚡ master] $ git add index.html
[redacted]:~/projects/myhack [⚡ master] $ git commit -s -m "just testing the webhook"
[master 6ba85ca] just testing the webhook
 1 file changed, 1 insertion(+)
[redacted]:~/projects/myhack [⚡ master] $ git push
X11 forwarding request failed on channel 0
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 385 bytes | 385.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:feri/myhack.git
   a829aea..6ba85ca  master -> master
[redacted]:~/projects/myhack [⚡ master] $ 
```

Check results again

Your new code is deployed and available at micromec.org.

Have a lot of fun...

Happy hacking!



myhack

This is our solution for the challenge

Just testing the webhook

AKRAINO EDGE STACK