**[Slide 1]:**

Buna ziua, numele meu este Mărgărit Darius, iar in minutele ce urmeaza va voi prezenta proiectul meu intitulat “Automated Workflow for Scalable AI using Kubernetes”.

**[Slide 2]:**

In aceasta schema am evidentiat principalele componente ale unui “pipeline” de “build” si “deploy” pentru o aplicatie.

Atunci cand programatorii adauga modificarile codului sursa al unei aplicatii intr-un “repository” (in acest caz Gitlab), este declansat automat un proces de “Continuous integration” definit de fisierul “.gitlab-ci.yaml” care foloseste fisiere de tip “dockerfile” pentru a genera imagini “docker”. Acestora li se atribuie “tag-uri” unice reprezentand versiunea imaginii. In continuare, imaginile ajung intr-un “Docker Registry”, iar procesul se termina cu modificarea unor valori intr-un “Manifest Repository”. “ArgoCD” vede aceste modificari si incepe procesul de “Continuous deployment / Continuous delivery ”, adica se reaplica fisierele de configurare ale aplicatiei in “Kubernetes cluster” cu noile imagini “docker”.

**[Slide 3]:**

Arhitectura acestui “cluster” are la baza un singur nod care contine un spatiu de nume pentru componentele esentiale functionarii lui numit “kube-system”, altul, “istio-system”, pentru agentul de gestionare a traficului si pentru uneltele de monitorizare, “knative-serving” pentru agentul care scaleaza aplicatia la nevoie, dar si alte elemente necesare functionarii, iar in cele din urma spatiul “default” in care se afla aplicatia.

Programatorul interactioneaza cu un “API” care la randul sau comunica cu un agent numit “kubelet”, acesta se ocupa cu configurarea tuturor resurselor din “cluster”. Utilizatorii au acces la aplicatie trecand prin doua porti numite “ingress gateway”, una in afara “cluster-ului”, iar una in interior. Ultima face legatura cu serviciile formand un “service mesh”, asa cum vom vedea in urmatorul slide.

**[Slide 4]:**

Poarta “istio-ingress gateway” expune serviciul “frontend” in afara “cluster-ului” folosind un “External IP” si portul 80 caruia Knative ii atribuie un DNS. Doar din “frontend” utilizatorul poate apela API-ul, acesta fiind un serviciu privat. API-ul apeleaza un serviciu “model” care este definit prin metoda “Canary Deployment” in doua versiuni: “v1” si “v2”. Aceasta metoda presupune testarea in productie a unui nou model, in acest caz “model-v2” prin redirectionarea a 10% din trafic catre acesta, iar 90% prin “model-v1”. Astfel, putem testa si monitoriza comportamentul noului model fara a afecta toti utilizatorii in caz de vreo problema.

**[Slide 5]:**

Video

**[Slide 6]:**

Q&A