ML platform
K8s deployment

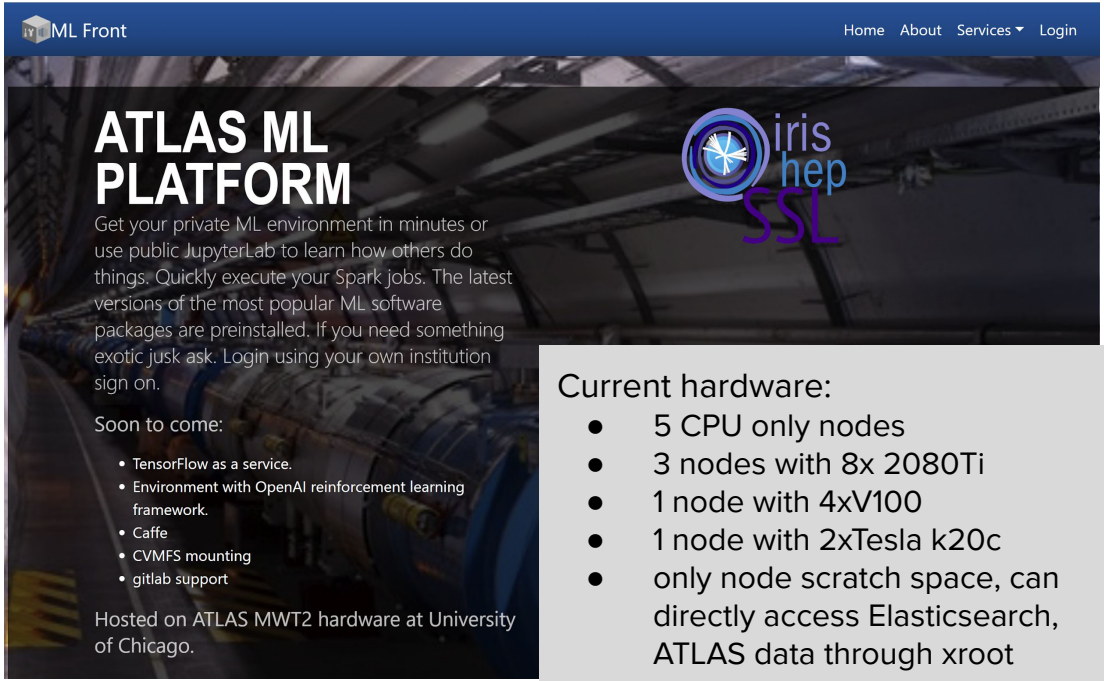
Ilija Vukotic

K8s-HEP Meetup

Jan 30, 2020

UC ML Platform

https://www.atlas-ml.org/



Current hardware:

- 5 CPU only nodes
- 3 nodes with 8x 2080Ti
- 1 node with 4xV100
- 1 node with 2xTesla k20c
- only node scratch space, can directly access Elasticsearch, ATLAS data through xroot

To add:

- NVidia Jetson Nano inference dev board.

- Nodejs, express/pug site running on k8s. Has full control of k8s deployments.
- Easy to customize/deploy elsewhere
  - ML workshops (eg. CoDaS-HEP)
  - Teaching computational physics courses (eg. Phys 250 at UChicago)
- Uses Globus+CILogon authentication.
- Authorization by the instance owner.
- Limits resources per instance.
- Extensible. Current services:
  - Private/Private JupyterLab
  - SparkJobs
  - Caffe ML service

Web frontend

- **Node.js** - easy async handling. Good K8s client.
- REST API - **Express**. Render engine **Pug**. Static pages easily customizable.
- **Globus** Authentication.
- Authorization by service manager.
- Mailing through **MailGun**.
- Backend DB and monitoring based on **Elasticsearch** at UC. Will add option of switching to **Redis.**
- Each instance runs in its own k8s namespace.
- It runs under service account and has its own **RBAC** allowing it to: ***get, create, delete, list, and watch*** all the ***pods, services, ingesses, and configmaps***.
- Frontend and all services created served through Ingress with **Letsencript *certmanager***.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ml-usatlas-org
  namespace: ml-usatlas-org
  labels:
    k8s-app: ml-usatlas-org
  annotations:
    kubernetes.io/ingress.class: slate
    certmanager.k8s.io/cluster-issuer:
letsencrypt-prod
spec:
  tls:
  - hosts:
    - www.atlas-ml.org
    secretName: auto-generated-ml-usatlas
  rules:
  - host: www.atlas-ml.org
    http:
      paths:
      - path: /
        backend:
          serviceName: ml-usatlas-org
          servicePort: 80
```

JupyterLab Notebooks

- **User chooses:**
  - Name, password
  - Time-To-Live
  - CPU cores, GPUs, RAM
  - Image
- **Regular lab, not a Hub**
- **Pod, Service, and Ingress are json templates** that get configured at request time.
- **Time-to-live** is stored as a pod label and is enforced by a one per minute checks of all pods by the frontend.
- **Client API completely follows k8s api**

```
client.api.v1.namespaces(ns).pods(name).delete()
client.apis.extensions.v1beta1.namespaces(ns).ingresses(name).delete()
client.api.v1.namespaces(ns).pods.get()
client.api.v1.namespaces(ns).pods.post({ body: jupyterPodManifest })
```

All k8s client **calls return promises**.
For some operations it is OK to **await**, for others there are k8s **events** that one can listen for.

You almost immediately get pod change to “running” but actually for service to start it can take minutes.
Handling pods stuck in “pending” adds complexity.

JupyterLabs – storage and services

- **Storage**
  - As these nodes are used primarily for ML, you want some fast local storage. We mount local path /data for scratch.
  - While we could easily provide nfs storage using NFS provisioner, finding out when to clean up PersistenVolumeClaims is tricky. Sharing them among instances needs a bit of web frontend development.
  - We thought about Amazon File System and cloud providers. Latencies would be prohibitive.
- **Services**
  - Ingress with Letsencript cert manager make things really simple and convenient. Users get nice and secure address eg. https://ilijalab.atlas-ml.org while spending no additional IPs.

Spark jobs

- User selects number of executors and gives web accessible executable
- Running with special role
- Results and data stored in /data

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: ml-usatlas-org
  name: spark-role
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods","services","configmaps"]
  verbs: ["get", "watch", "list", "create","delete"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: spark-runner
  namespace: ml-usatlas-org
subjects:
- kind: ServiceAccount
  name: spark-acc
  namespace: ml-usatlas-org
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: spark-role
  apiGroup: rbac.authorization.k8s.io
```

Future

- **Redis backend**
- **Federating deployments** - while it is already possible to deploy on a cluster different from the one running the frontend, services, ingresses are complicated. Would prefer kubernetes side solution.
- **Tensorflow As A Service** - nobody is pressing for it...
- **Special PyTorch image -** ditto
- **Optional CVMFS mounts** - should be ready, needs testing
- **Provide shared filesystem** - want to provide small distributed home directories for users to keep their code in. Looked at Amazon's Elastic FS, and Google's streaming FS - both are NFS mounted though. Skeptical of performance.
- **Extend service lifetimes** - users requested it.
- **Cloud overflow** - would help with short term bursts.
- **SLATE deployment** - ML platform needs to create new services, pods, at run time, so requires more permissions than SLATE would currently allow. Frontend would have to be more generic and easier to customize.
- Investigating a possibility to use google **AI-Hub**. Rather nice platform - one Tesla k80 for free, sharing and concurrent work on models stored in Google streaming FS, jupyterlab-like environment.