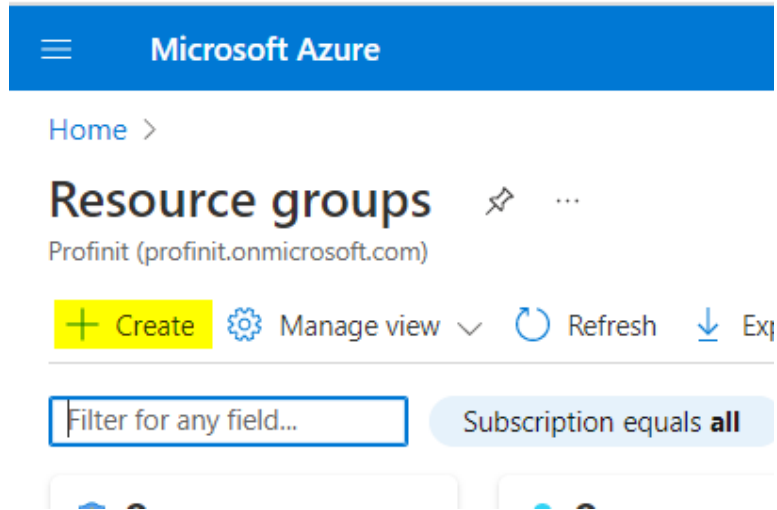


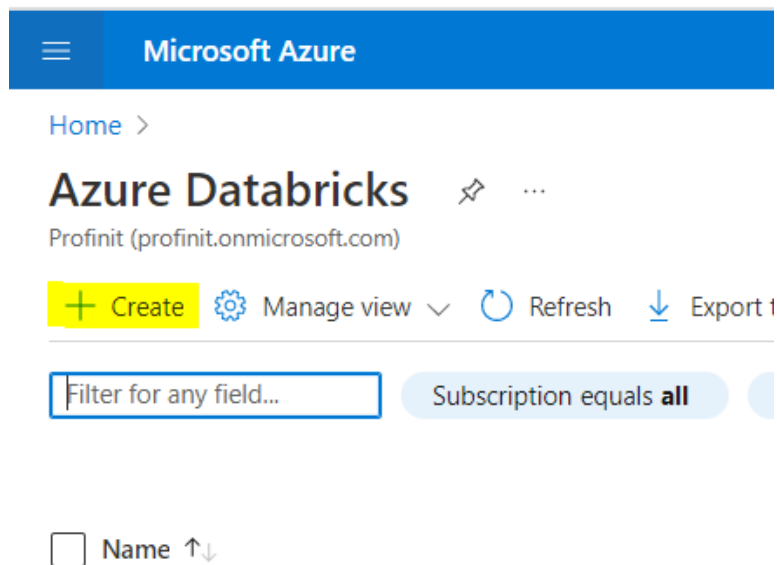
Initial steps in Azure

Login to your azure students account <https://azure.microsoft.com/cs-cz/free/students/> and create a resource, where you will host your services.



The screenshot shows the Microsoft Azure portal interface. At the top is a blue header with the Microsoft Azure logo. Below the header, there is a breadcrumb trail 'Home >'. The main heading is 'Resource groups' with a pin icon and a menu icon. Below the heading, it says 'Profinit (profinit.onmicrosoft.com)'. There are several action buttons: a yellow '+ Create' button, a 'Manage view' button with a gear icon and a dropdown arrow, a 'Refresh' button with a circular arrow icon, and an 'Export' button with a download icon. Below these buttons is a search bar with the placeholder text 'Filter for any field...' and a filter button that says 'Subscription equals all'. The bottom of the screenshot shows the top of two resource group cards.

Create Databricks Workspace



The screenshot shows the Microsoft Azure portal interface for the 'Azure Databricks' page. At the top is a blue header with the Microsoft Azure logo. Below the header, there is a breadcrumb trail 'Home >'. The main heading is 'Azure Databricks' with a pin icon and a menu icon. Below the heading, it says 'Profinit (profinit.onmicrosoft.com)'. There are several action buttons: a yellow '+ Create' button, a 'Manage view' button with a gear icon and a dropdown arrow, a 'Refresh' button with a circular arrow icon, and an 'Export' button with a download icon. Below these buttons is a search bar with the placeholder text 'Filter for any field...' and a filter button that says 'Subscription equals all'. At the bottom left, there is a checkbox and the text 'Name ↑↓'.

Select your newly created Dbx workspace, Launch, Sign in and voila.



Launch Workspace

▼ Initial steps in Databricks

▼ Connect your GIT repo

Generate personal access token from your git provider, see [github](#), [gitlab](#)



User Settings

Access tokens **Git integration** Notebook settings Email preferen

With co-versioned repo

Databricks Repos allows you to clone a remote Git repo, which you can specif

With individual notebooks

Although we recommended using co-versioned repo for Git integration, Data

Set your Git provider and credentials

You can also set your Git provider credentials via API.[Learn More](#)

Select your Git provider from the dropdown.

Git provider

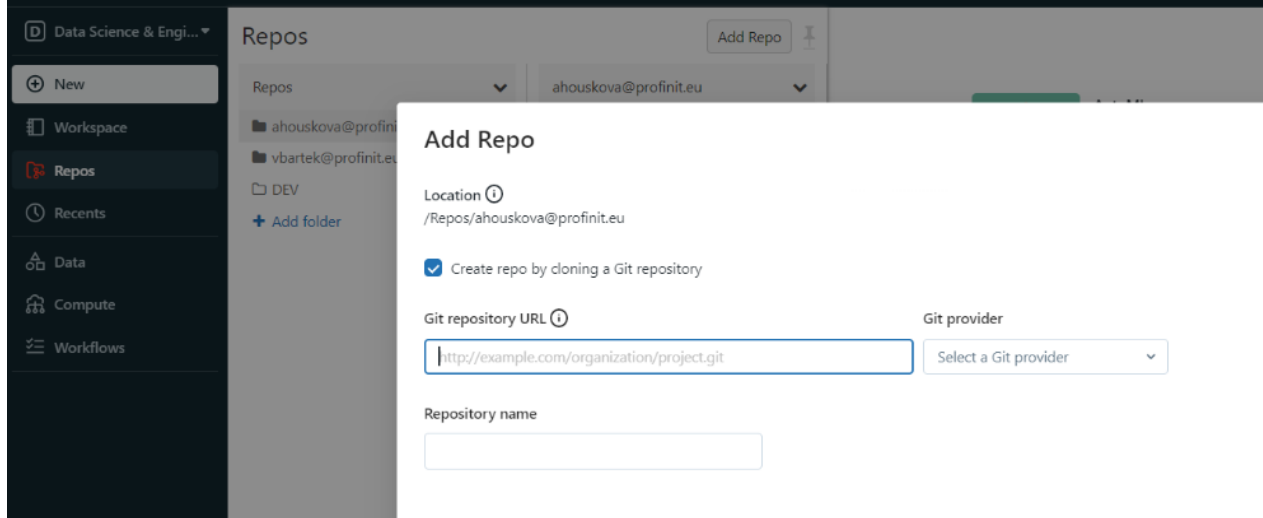
Git provider email

To generate a GitLab personal access token, follow the [GitLab documentation](#)

Token

Save

Navigate to Repos and add your repository; pull if anything present



made with
Picsart

▼ Create a cluster

Navigate to Computer and create desired cluster. Be reasonable with resources and termination time.

Be advised, while streaming you will need to terminate the cluster manually, because the termination time is only for clusters that are idle.

For first runs and debug, single node is enough.

Clusters / New Compute [UI Preview](#) [Provide feedback](#)

Alisa Benešová's Cluster [✎](#)

Multi node Single node

Access mode [?](#) **Single user access** [?](#)

Single user |

Performance

Databricks runtime version [?](#)

Runtime: 11.2 (Scala 2.12, Spark 3.3.0) |

Use Photon Acceleration [?](#)

Node type [?](#)

Standard_DS3_v2 | 14 GB Memory, 4 Cores | [?](#)

Terminate after minutes of inactivity [?](#)

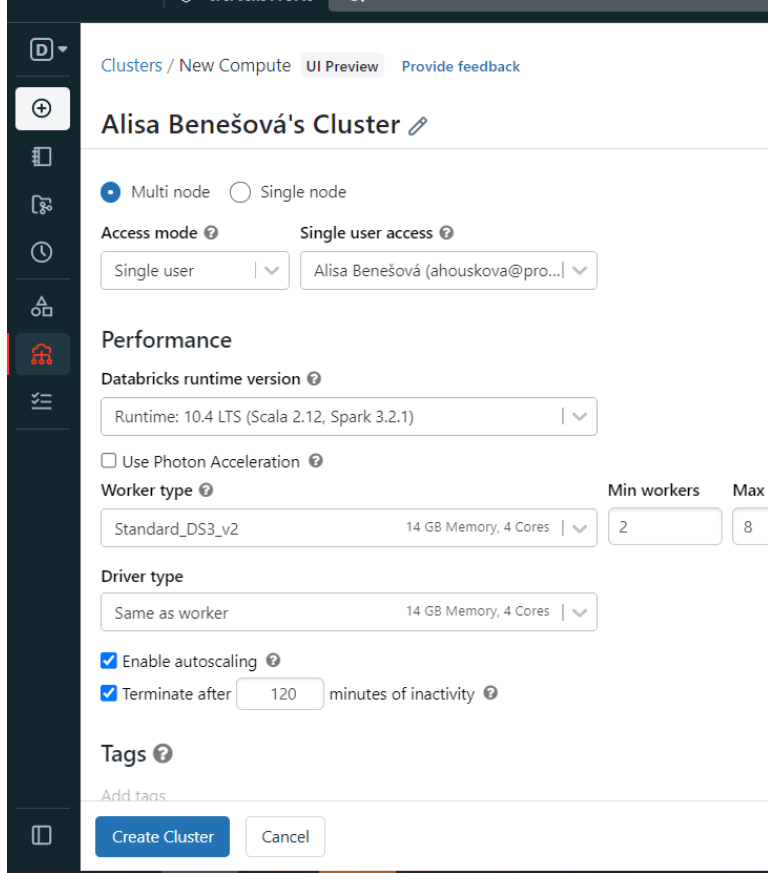
Tags [?](#)

Add tags

Key	Value
<input data-bbox="147 852 591 900" type="text"/>	<input data-bbox="618 852 819 900" type="text"/>

> Automatically added tags

Once you are sure with solution, you may test your solution on a stronger cluster with bigger amount of data, see multi node cluster below.



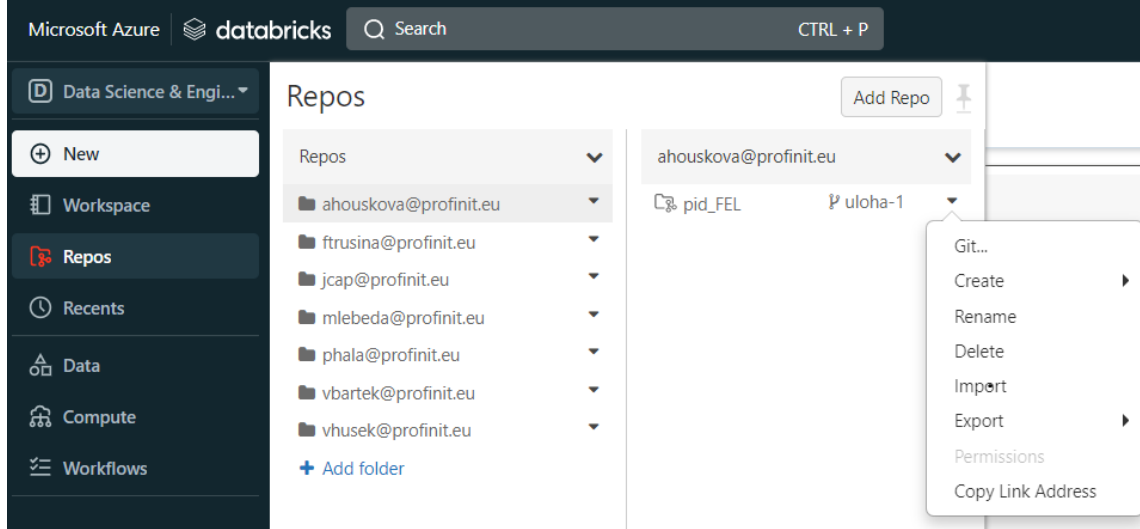
▼ Create a Notebook

You can create either notebook, which is not connected to your repository, or create a Repo files, which you can git control. **We will use git controlled files.** GIT in Databricks environment has some limitations, eg. merging is not possible.

Repos/choose_your_repo/choose_your_branch/Create/Notebook

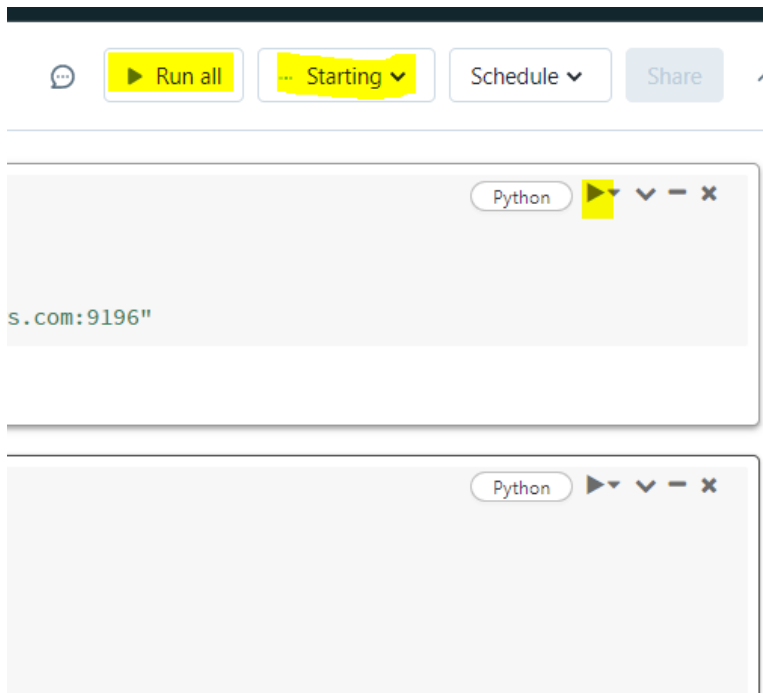
You can Create branch via *Git...* option or in git gui/console.

When you want to use Databricks cluster and Databricks features, it is not necessary to use Databricks GUI. It is also possible to use Visual Code, with simple Azure verification and Databricks plugin. Feel free to explore, but it is beyond this course.



Once your Notebook is created, you are free to code and run it against cluster.

Starting a cluster will take some time, few minutes.



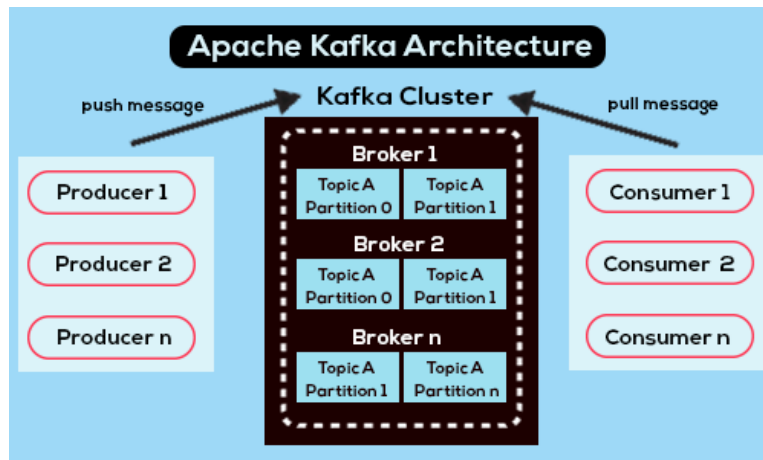
▼ Let's get started with kafka

▼ Kafka brief review

Kafka is

- Scalable - distributed architecture capable of handling huge amount of incoming messages in high speed.
- High Throughput - Thousands messages per second, depends on the cluster.
- Low Latency - the latency = time take for a system to process a single event is under 10ms.
- Fault Tolerance - it is possible to use replication, which helps Kafka handle failures at nodes in the cluster. Durability - Kafka can persist data for desired amount of time.
- Reliability - all of above makes Kafka very popular, the most used and reliable messaging system.

Ability to handle real-time data: Kafka supports real-time data handling and is an excellent choice when data has to be processed in real-time.



Broker Broker works as a container that can hold multiple topics with different partitions.

Topic Kafka stores its data in streams, called topics. Topics are append-only, immutable logs of events. Typically, events of the same type, or events that are in some way related, would go into the same topic.

Consumer Can read/pull/subscribe present data from kafka topics.

Producer Can write/push/publish new data to kafka topics.

▼ Run Kafka

Download desired version in <https://kafka.apache.org/downloads> When on local, you will need to start zookeeper and kafka server. For this, you will need to navigate into your kafka folder in console.

```
1 tar -xzf kafka_2.13-3.3.1.tgz
2
3 cd kafka_2.13-3.3.1 seznamu
4
5
6 # Start the ZooKeeper service
7 bin/zookeeper-server-start.sh config/zookeeper.properties
8
9 # Start the Kafka broker service
10 bin/kafka-server-start.sh config/server.properties
```

Upravíte dvojitým kliknutím (nebo stisknutím klávesy Enter)

▼ Create a topic

We will be using local server as a bootstrap server, so the address is localhost and standard port 9092.

```
1 bin/kafka-topics.sh --create --topic first-topic --bootstrap-server localhost:9092
```

▼ Create some events

Once you start console producer (it is prepared script for reading from console), it will wait for your input. You can kill it with `Ctrl+C`.

```
1 bin/kafka-console-producer.sh --topic first-topic --bootstrap-server localhost:9092
2 First message
3 Second message
```

▼ Read events from topic

Open a new console window, not to kill producer. Run console consumer to read messages you have sent earlier. You can read from beginning (by adding `--from-beginning`). Once script started, you should see appearing messages sent earlier, or real time in the producer window.

```
1 bin/kafka-console-consumer.sh --topic first-topic --from-beginning --bootstrap-server localhost:9092
```