Bucharest Big Data Meetup

Tech talks, use cases, all big data related topics

March 14th meetup

6:30 PM - 7:00 PM getting together

7:00 - 8:00 **Intro to Kubernetes – from containers to orchestrating the world,** Alex Sirbu, R&D Team Lead at Lentiq, a Bigstep Company

8:00 - 8:30: Pizza and drinks sponsored by **Netopia**.

Sponsored by

NETOPIA mobilPay SMS. CARD. CASH

web2sms™ Self Service Marketing

Organizer

Valentina Crisan

Intro to Kubernetes -
from containers to orchestrating the world

LENTIQ
a Bigstep company

Contents

LENTIQ
a Bigstep company

Who am I?

Alex Sirbu

R&D Team Lead @ Lentiq

Building the EdgeLake – flexible cloud data lake service

Built on Kubernetes!

Before – R&D Team Lead @ Bigstep, working on bare metal cloud orchestrator

LENTIQ
a Bigstep company

Containers

Container - What's in a name?

Coming from the shipping industry

Caused aquatic theme for domain

Shipping containers

Portability - can be used on any of supported types of ships

Wide variety of cargo that can be packed inside

Standard sizes - standard fittings on ships

Many containers on a ship

Isolates cargo from each other

LENTIQ
a Bigstep company

Translated to software

Portability - can be used on any supported system (system with container execution environment)

Wide variety of software that can be packed inside

Standard format

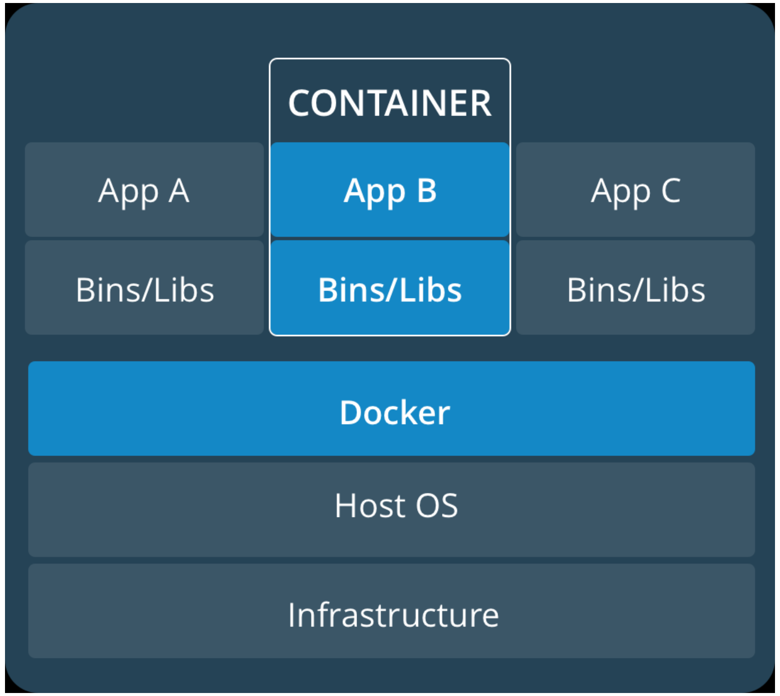
Many containers to a physical node

Isolates execution of one container from another

LENTIQ
a Bigstep company

What is a container?

way to pack code and dependencies together
can run anywhere
execute multiple containers to a physical machine



LENTIQ
a Bigstep company

Sounds familiar?

same concept as virtual machines

pack OS and software together, to run in isolated instances

can run anywhere the specific hypervisor runs

multiple VMs to a physical machine

LENTIQ
a Bigstep company

How do VMs work?

hypervisor = layer between VM and kernel

emulates system calls

allows multiple types of operating systems on a machine (Windows on Linux)

overhead for hypervisor

LENTIQ
a Bigstep company

Containers on the other hand ...

only contain application and application-related libraries and frameworks, that run on the host machine's kernel

smaller

lower overhead

differences in OS distributions and dependencies are abstracted - same kernel

LENTIQ
a Bigstep company

Working together, not against each other

Windows on Linux possible only with VMs

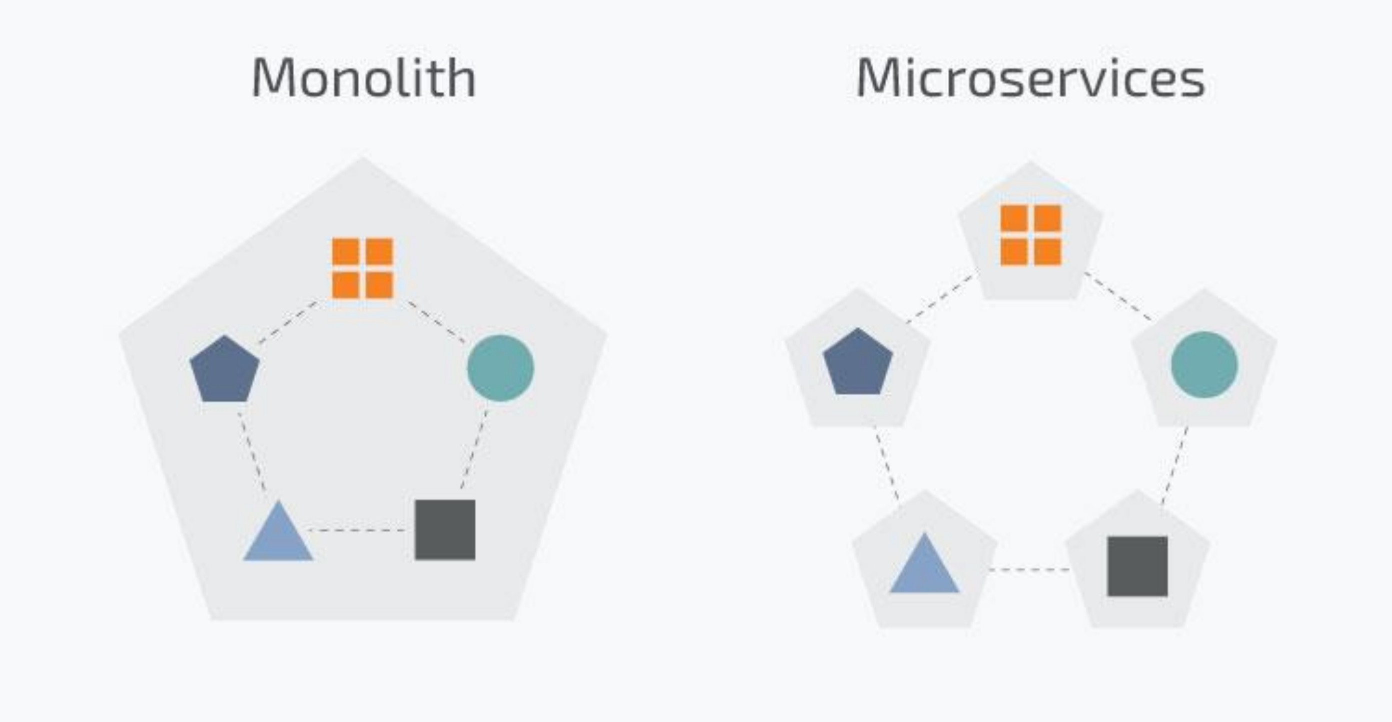
older software needs to be adapted to be run as containers (and won't)

usage of VMs as a medium for containers (better isolation and easier scaling)

LENTIQ
a Bigstep company

Greater modularity in software

Monolithic application → independent services that interact (microservices)



LENTIQ
a Bigstep company

Containers empowering microservices

quicker start times -> easy to prototype or scale

allow work to be done independently on modules -> independent releases for components (take care of interfaces)

isolated and abstracted runtime environments, that can be tailored for each module

shared runtime environment, for heterogenous applications

LENTIQ
a Bigstep company

Containers history – early days

need for resources to be shared among many users -> multiple terminals connected to the same mainframe

main problem - execution can cause the main computer to crash -> down for everybody

Containers history – isolating more and more

Chroot – 1979 – change root directory for a running process, along with children → segregate and isolate processes, protecting global environment
Jails – additional process sandboxing features for isolating filesystems, users, networks (limiting apps in their functionality)
Solaris Zones – full application environments, with full user, process and filesystem space
Cgroups – 2006 – process containers designed for isolating and limiting the resource usage of a process

LENTIQ
a Bigstep company

Containers history – Linux containers (lxc)

2008

Provides virtualization at OS level

Provides containers with its own process and network space

Containers history – Docker

2013

Container execution and management system

Originally started with lxc, then moved to libcontainer, which allows containers to work with:

- linux namespaces
- libcontainer control groups
- capabilities
- app armor security profiles
- network interfaces
- firewall rules

Containers history – OCI & CNCF

Open Container Initiative – 2015
industry format for a container format and container runtime software for all platforms
spend resources on developing additional software to support use of standard containers, instead of format alternatives
Cloud Native Container Foundation – 2015
Working on different projects to further standardize the market:

- Kubernetes
- Container Network Interface
- Containerd

LENTIQ
a Bigstep company

Container orchestration

LENTIQ
a Bigstep company

Need for something more?

docker started out with a CLI tool on top of lxc, that built, created, started, stopped and exec'd containers

does management at a node level, upon specific requests

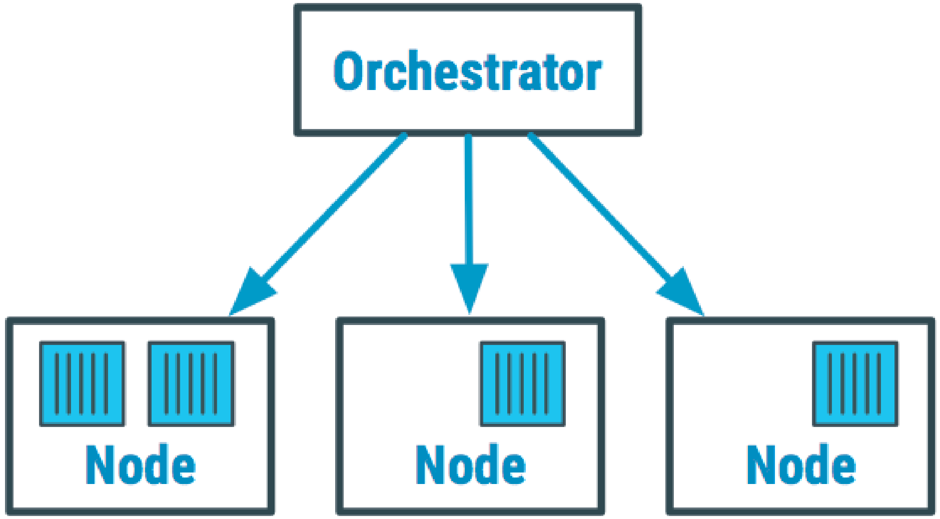
easy to manually manage with up to 100s of containers and 10s of nodes, but what next?

LENTIQ
a Bigstep company

Orchestrator

manage and organize both hosts and docker containers running on a cluster

main issue - resource allocation - where can a container be scheduled, to fulfill its requirements (CPU/RAM/disk) + how to keep track of nodes and scale



LENTIQ
a Bigstep company

Some orchestrator tasks

manage networking and access
track state of containers
scale services
do load balancing
relocation in case of unresponsive host
service discovery
attribute storage to containers
…

LENTIQ
a Bigstep company

Orchestrator options

Kubernetes – open-source, product of CNCF

Apache Mesos – cluster management tool, with container orchestration being only one of the things it can do, originally through a plugin called Marathon

Docker Swarm – integrated in docker container platform

Comparison – ease of use

Kubernetes - fairly complex, steeper learning curve, web UI helps management + great API and documentation

Swarm - easy to pick up, comes already installed with docker

Mesos - trickiest to set up, requires specific plugin knowledge

LENTIQ
a Bigstep company

Comparison – features and functionality - Kubernetes

most complete and fully integrated feature set

out of the box, comes with various features that are offered as add-ons for the others (auto-scaling, load balancing, etc)

customisable with controllers

supports different container runtimes

has evolved based on the Mesos experience and fixed most of the Mesos bugs and missing functionalities

LENTIQ a Bigstep company

Comparison – features and functionality - Swarm

can't compete feature-wise

out-of-the-box

requires third party tools for advanced configuration

faster deploy times

LENTIQ
a Bigstep company

Comparison – features and functionality - Mesos

wider array of potential functionality

more mature software than all (as a whole)

Marathon first, but lacks features – persistent storage in particular

allows combination of containers with normal applications on the same cluster

some features hidden behind enterprise version

LENTIQ
a Bigstep company

Comparison – scalability

Kubernetes - ability to easily schedule groups of complex applications, enabling scaling up to enterprise-level requirements

Swarm - better suited to small or medium clusters

Mesos - enables container orchestration on the largest scale

LENTIQ
a Bigstep company

To sum up

entry level and testing - Swarm

enterprise-level - Kubernetes

more complex or large clusters - Mesos

LENTIQ
a Bigstep company

Lately …

Mesos announced Kubernetes support as container orchestration, alongside Marathon

Docker Enterprise Edition - integration with Kubernetes alongside Swarm

→ Kubernetes becoming the de-facto standard for container orchestration (allowing developers to focus on building on top instead of alternatives)

LENTIQ
a Bigstep company

Kubernetes – What? Why? How?

LENTIQ
a Bigstep company

What is Kubernetes?

“Kubernetes” = Greek for governor, helmsman, captain

open-source container orchestration system

originally designed by Google, maintained by CNCF

aim to provide "platform for automating deployment, scaling and operations of application containers across clusters of hosts"

LENTIQ
a Bigstep company

kubernetes

Why Kubernetes? - Goals

Main objectives, stated by devs, for community

Achieve velocity

Allow scaling of both software and teams

Present abstract infrastructure

Gain efficiency

LENTIQ
a Bigstep company

Achieve velocity

Velocity = number of things you ship while maintaining a highly available service

Achieved by:

- immutability - created artifact cannot be changed
- declarative configuration - declare desired state and Kubernetes' job is to ensure it matches
- self-healing systems - trying to maintain desired states if something changes

LENTIQ
a Bigstep company

Allow scaling of software

encouraging decoupling in applications - separated components that communicate via defined APIs via load-balanced services

running in shared abstract environment, without interference

utilizing standard container format that runs on any machine

LENTIQ
a Bigstep company

Allow scaling of teams

separation of concerns for consistency and scaling

- application ops rely on the SLA provided by the platform
- orchestrator ops uphold SLA

LENTIQ
a Bigstep company

Present abstract infrastructure

decoupling container images and machines

cluster can be heterogenous and reduce overhead and cost

portability - container can be used on another cluster without being changed

LENTIQ
a Bigstep company

Gain efficiency

optimized usage of physical machines - multiple containers on same machine

isolated with namespaces, to not interfere with each other

LENTIQ
a Bigstep company

Computing

Communication

Coordination

Application versioning

Stateful

Container image format

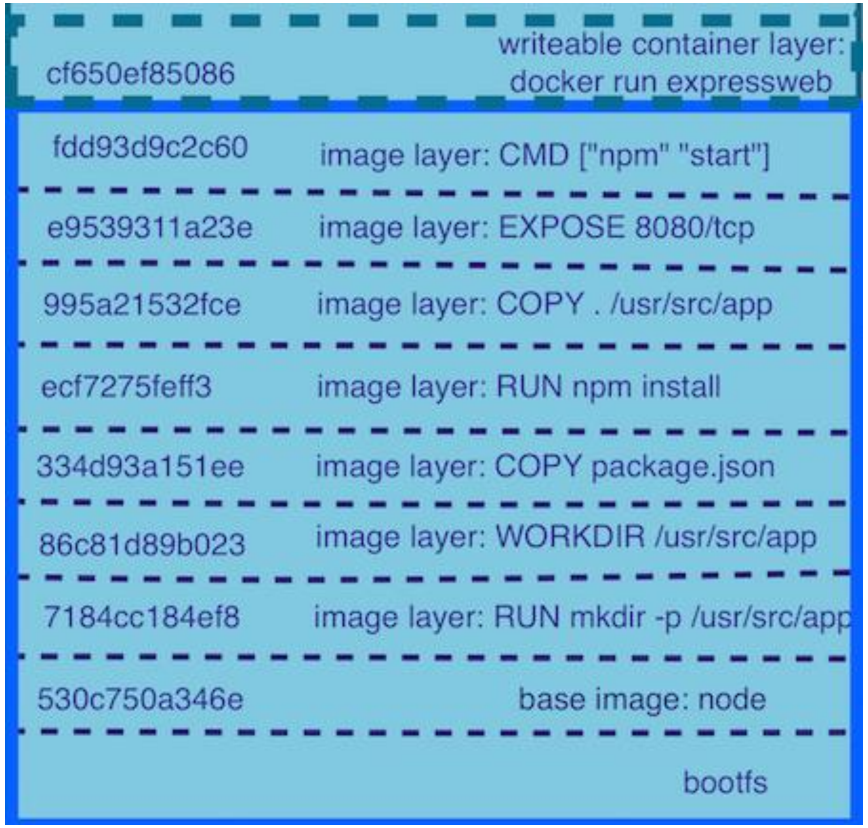
layered format, allowing to inherit from lower levels and to modify them by adding, changing or removing files

using unified file system that allows this layering

issue – deleted file remains in older layers

image size bigger and build time longer -> development of better tools

LENTIQ a Bigstep company



Running a container

image provides the filesystem base for execution

configuration, to interoperate with the rest of the system – environment variables, CPU/RAM requirements, process to execute, ports to expose, etc.

LENTIQ
a Bigstep company

Kubernetes and containers

Can you deploy a container in Kubernetes? NO (not directly)

Why not? Because the smallest deployable unit of computing is not a container, but ...

Pod

smallest deployable unit of computing in Kubernetes

colocated multiple apps(containers) into a single atomic unit, scheduled onto a single machine

upon creation, statically allocated to a certain node

Pod

each container runs in its own cgroup (CPU + RAM allocation), but they share some namespaces and filesystems, such as:

- IP address and port space
- same hostname
- IPC channels for communication

So, why a pod and not container directly?

all or nothing approach for a group of symbiotic containers, that need to be kept together at all times

pod considered running if all containers are scheduled and running

Can you deploy a container in Kubernetes? Yes, inside a pod!

LENTIQ
a Bigstep company

When to have multiple containers inside a pod?

when it's impossible for them to work on different machines (sharing local filesystem or using IPC)

when one of them facilitates communication with the other without altering it (adapter)

when one of them offers support for the other (logging/monitoring)

when one of them configures the other

LENTIQ
a Bigstep company

Pod scheduling

scheduler tries to scatter replicas for reliability and are never moved (immutability)

what happens when physical node dies? pod needs to be deleted in order to be rescheduled

LENTIQ
a Bigstep company

Pod health checks

process health check - main process is always running and has not exited (for each container)

liveness probe - application specific, determines if application actually does what the probe knows it should do

readiness probe - on start, it might take a while until the application fully loads and can process requests as expected

LENTIQ
a Bigstep company

ConfigMaps and Secrets

configure pods, make images more reusable

live update on change (application needs to be able to reload)

secrets mounted as ram disk => not written to actual filesystem

LENTIQ
a Bigstep company

Labels and annotations

labels - key/value pairs attached to objects arbitrarily, providing foundation for grouping objects

annotations - key/value pairs designed to hold non-identifying information that can be leveraged by tools and libraries

metadata needed by the system to provide identification, grouping and higher-level features

Computing

Communication

Coordination

Application versioning

Stateful

LENTIQ
a Bigstep company

Communication challenges

between pods - using hardcoded IPs would be the wrong way to do it, as pods might be rescheduled on different nodes and change IPs

from outside - keep track of all pods that provide a certain service and loadbalance between them

LENTIQ
a Bigstep company

Service discovery

find which processes are listening at which addresses for which services

do it quickly and reliably, with low-latency, storing richer definitions of what those services are

public DNS isn't dynamic enough to deal with the amount of updates

LENTIQ
a Bigstep company

Service

abstraction which defines a logical set of Pods (selected using label selector), that provide the same functionality (same microservice)

different types, for different types of exposure provided by the service

ClusterIP Service

used for intra-cluster communication

special IP that will load-balance across all of the pods identified by service selector (which is a label selector)

allocated on create and cannot be changed until deletion of service, irrespective of number of pods

LENTIQ
a Bigstep company

ClusterIP Service

internal Kubernetes DNS service allows service name to be used to access pods - *my-svc.my-namespace.svc.cluster.local*

if using readiness probes, only ready pods will be loadbalanced

LENTIQ
a Bigstep company

NodePorts Service

used to access pod from outside of cluster

system picks a port and on all cluster nodes, traffic on that port is sent to service to loadbalance to pods

external support needed to acces (ex - load-balanced DNS record with all cluster nodes)

LENTIQ
a Bigstep company

LoadBalancer Service

implements the needed external support, to make access to pods easier in cloud environments

cloud providers provide support inside Kubernetes, to provision whatever is needed inside their environment to access service directly

Ex: IP gets allocated in Google, URL in AWS

LENTIQ
a Bigstep company

Computing

Communication

Coordination

Application versioning

Stateful

Managing multiple pods

even though the pod is the building computing block for Kubernetes, working directly with Pods is tedious

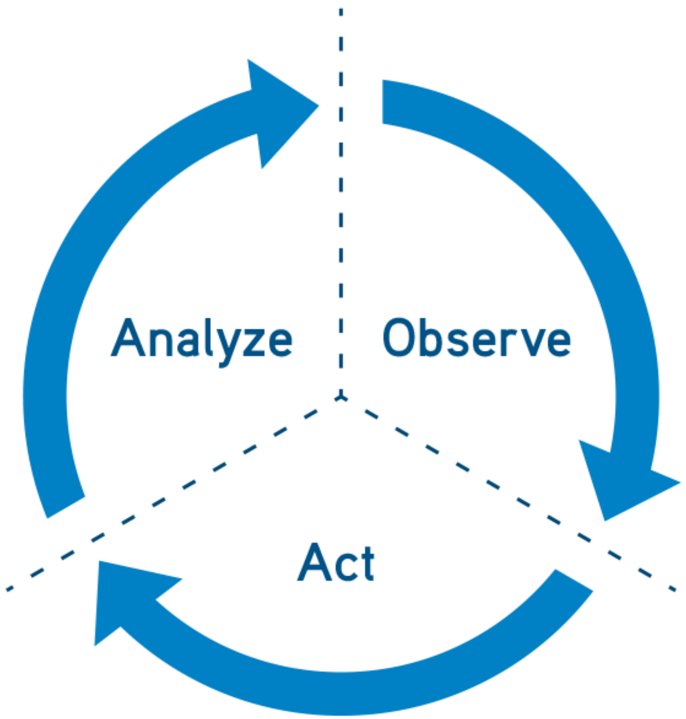
Kubernetes abstracts different needs, to make it easier

Reconciliation loop

start with a desired state

observe current state and take action to try to make it match the desired one

goal-driven, self-healing system



LENTIQ
a Bigstep company

ReplicaSet

makes sure a given number of identical pods are up at any time

does not "own" the pods it manages - selected with labels

can adopt existing pods with those labels

quarantine containers - remove label!

by default, on delete it deletes the pods, but can be set not to (--*cascade*=*false*)

LENTIQ
a Bigstep company

ReplicaSet

homogenous environment

designed for nearly stateless services

on shrink, the pod to be deleted is chosen arbitrarily

usage example – static web server

LENTIQ
a Bigstep company

DaemonSet

makes sure a pod is executed on each physical node

usual usage - agent, logging, monitoring

can select nodes using node labels and node selector

can perform rollingUpdate on pods

LENTIQ
a Bigstep company

Jobs

with ReplicaSets and DaemonSets, if pod's process exit (even with 0), it gets restarted, to keep consistency

Job manages Pods that need to run and exit with 0 (restarted until they do so)

configured with number of allowed parallel pods and number of expected completions

Job patterns

one shot - pods = 1, completion = 1

parallelism - pods >= 1, completion >= 1 (nr. pods will never be > completion)

work queue - pods >= 1, completion not set (pods will run until they all finish, no new ones created after one finishes)

LENTIQ
a Bigstep company

Computing

Communication

Coordination

Application versioning

Stateful

Application versioning

there comes a time when a new version needs to be released

usually with no service downtime

check new version works before going through with the full release

Kubernetes has an abstraction for this!

Deployment

manages replica set through time and versions for pod spec

scale != version update

using health checks, makes sure a new version works

allows rollbacks to older versions (keeps track of changes)

LENTIQ
a Bigstep company

Deployment strategies - recreate

all previous pods are destroyed and new pods are created

quickest

downtime while new pods start

in case of problems and rollback, even more downtime

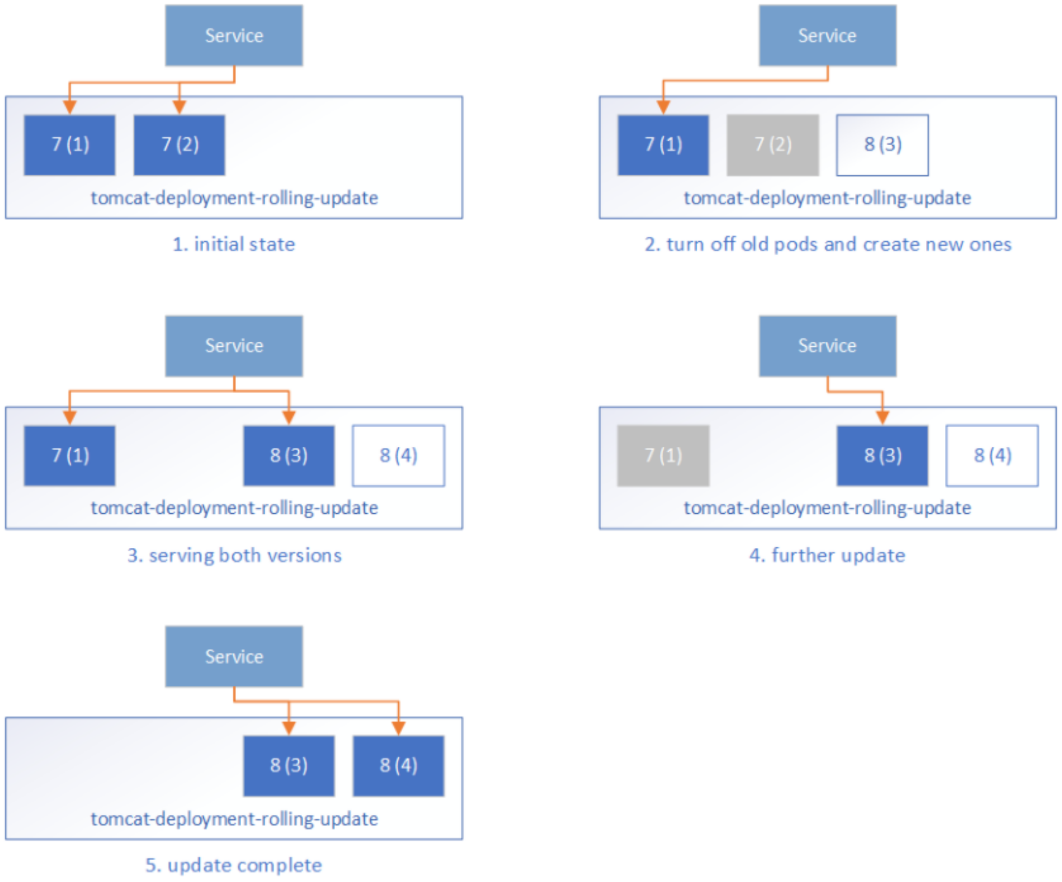
LENTIQ
a Bigstep company

Deployment strategies – rolling update

configured with max unavailable and max surge

max unavailable = number of pods that can be doing updates/rollbacks at a time, from the number of replicas

max surge = number of additional pods to be used for update/rollback



LENTIQ
a Bigstep company

Horizontal Pod Autoscaling

built-in feature

automatically shrink/increase based on certain parameters

works with *heapster* pod, that gathers information from containers

works on top of replicaSets as well as Deployments

LENTIQ
a Bigstep company

Computing
Communication
Coordination
Application versioning
Stateful

stateless -> stateFULL

not all applications are stateless (most aren't)

state = unique pod identity (not interchange-able anymore) + persistence of data (even when rescheduled on different nodes)

Kubernetes comes to the rescue again!

LENTIQ
a Bigstep company

StatefulSet

each replica gets a persistent hostname with unique id

created in order of index low -> high

delete in order of index high -> low

usage example - database

LENTIQ
a Bigstep company

Headless service

no need for loadbalancing and a single service IP

service with *clusterIP: none*

used in order to create DNS records for replicas, which can be used to uniquely identify them (0.svc..., 1.svc...)

LENTIQ
a Bigstep company

Persistence of data

how to define the physical location of data and how much should be allocated to each

how to actually allocate and link certain data to specific pods

abstracted and decoupled through PersistentVolume subsystem

LENTIQ a Bigstep company

PersistentVolume

abstraction of a piece of storage in the cluster that can be used

lifecycle independent of any individual pod that uses it

can be manually put by an operator or can be provisioned dynamically (usually in cloud services)

LENTIQ
a Bigstep company

PersistentVolumeClaim

request for storage by an user

storage equivalent of a pod

links a persistent volume to a pod for the pod's lifetime

doesn't affect the persistent volume upon pod deletion (unless explicitly specified)

LENTIQ
a Bigstep company

Wrap-up

computing building block = Pod
communication building block = Service
grouping = Labels and Annotations
configuration = ConfigMap and Secrets
stateless pod coordination = ReplicaSet, DaemonSet, Job
application updates = Deployment
stateful pod coordination = StatefulSet
storage building block = PersistentVolumeClaim on top of PersistentVolume

LENTIQ
a Bigstep company

Extending Kubernetes

LENTIQ
a Bigstep company

Soooo many things to configure :(

at least one controller

some services

some configMaps and Secrets

preallocate persistentVolumes or create storage class for dynamic provisioning

Solution: another level of abstraction

higher-level controller that can manage lower-level elements

for the moment, not included in Kubernetes ... YET!

BUT can be added, through third-party controllers

What is Helm?

package manager for Kubernetes

provides higher-level abstraction (Chart) to configure full-fledged applications

manage complexity, easy upgrades, simple sharing of full application setups, safe rollbacks

LENTIQ
a Bigstep company

How does Helm work?

Helm CLI + Tiller server in Kubernetes (which is a controller)

CLI responsible for management + requests for releases of charts on Kubernetes

Tiller - listens for requests, combines chart + configuration = release, install release, track release

Helm++

Helm release controller - current Lentiq way to manage applications

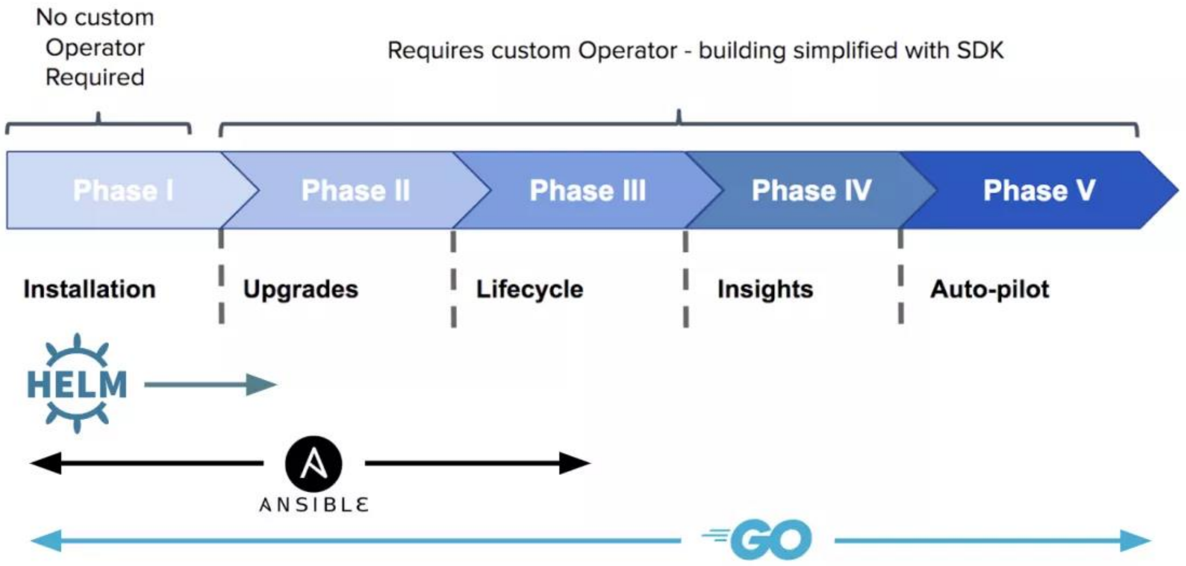
expose HelmRelease as a CRD (custom resource definition) in Kubernetes, to work directly with Kubernetes to manage apps

What are Operators?

domain-specific controller

manages lifetime of a single application

works with Kubernetes primitives, as well as performing application-specific steps



LENTIQ
a Bigstep company

Operators

pre and post provision hooks, for application-specific operations

single tool to perform all management (kubectl)

work in a scalable, repeatable, standard fashion

improve resiliency while reducing burden on IT teams

Operator framework

open-source toolkit to manage Kubernetes native applications

provides ways to implement Operators

multiple companies adopted and implemented Operators for their software, to provide the quickest start

LENTIQ
a Bigstep company

Operator gallery

https://github.com/operator-framework/awesome-operators

- Spark
- Kafka
- Cassandra
- MongoDB
- MySQL
- Airflow

LENTIQ
a Bigstep company

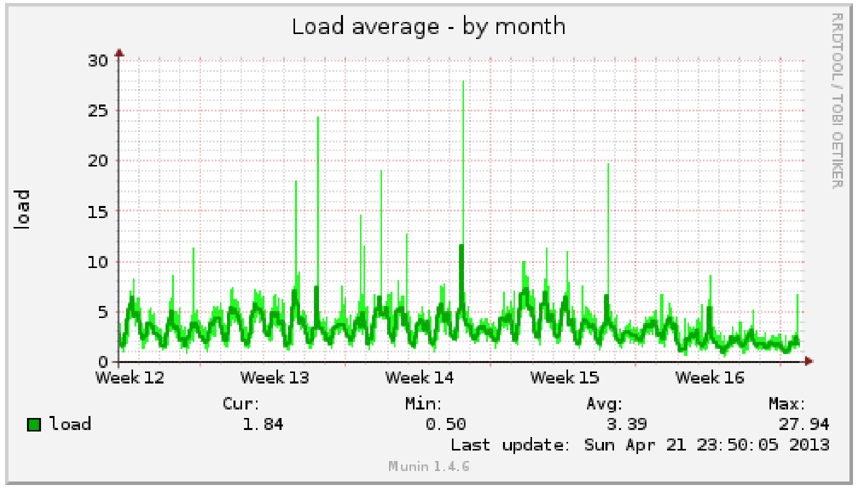
Kubernetes and Big Data

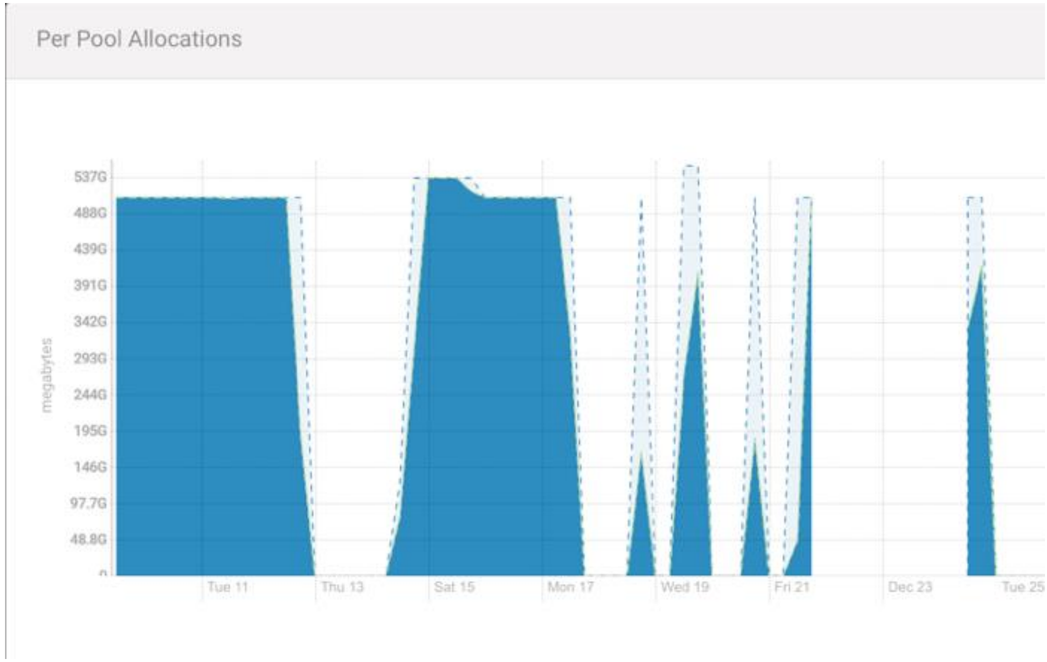
LENTIQ
a Bigstep company

Usual workload

bursts of 100% utilization, followed by zero

→ quick startup times + scaling capabilities helps to properly scale and use cluster efficiently





LENTIQ
a Bigstep company

Vs Hadoop distributions

can run any workload (vs supported apps)

smaller footprint

robust

no vendor lock-in

cloud native

designed for self-service

LENTIQ
a Bigstep company

Effects

fewer ops people needed (but more highly qualified), but with no domain specific knowledge (remember decoupling?)

higher abstraction level => focus on how to use the software and not how to set it up

shift more to cloud

LENTIQ
a Bigstep company

Why cloud?

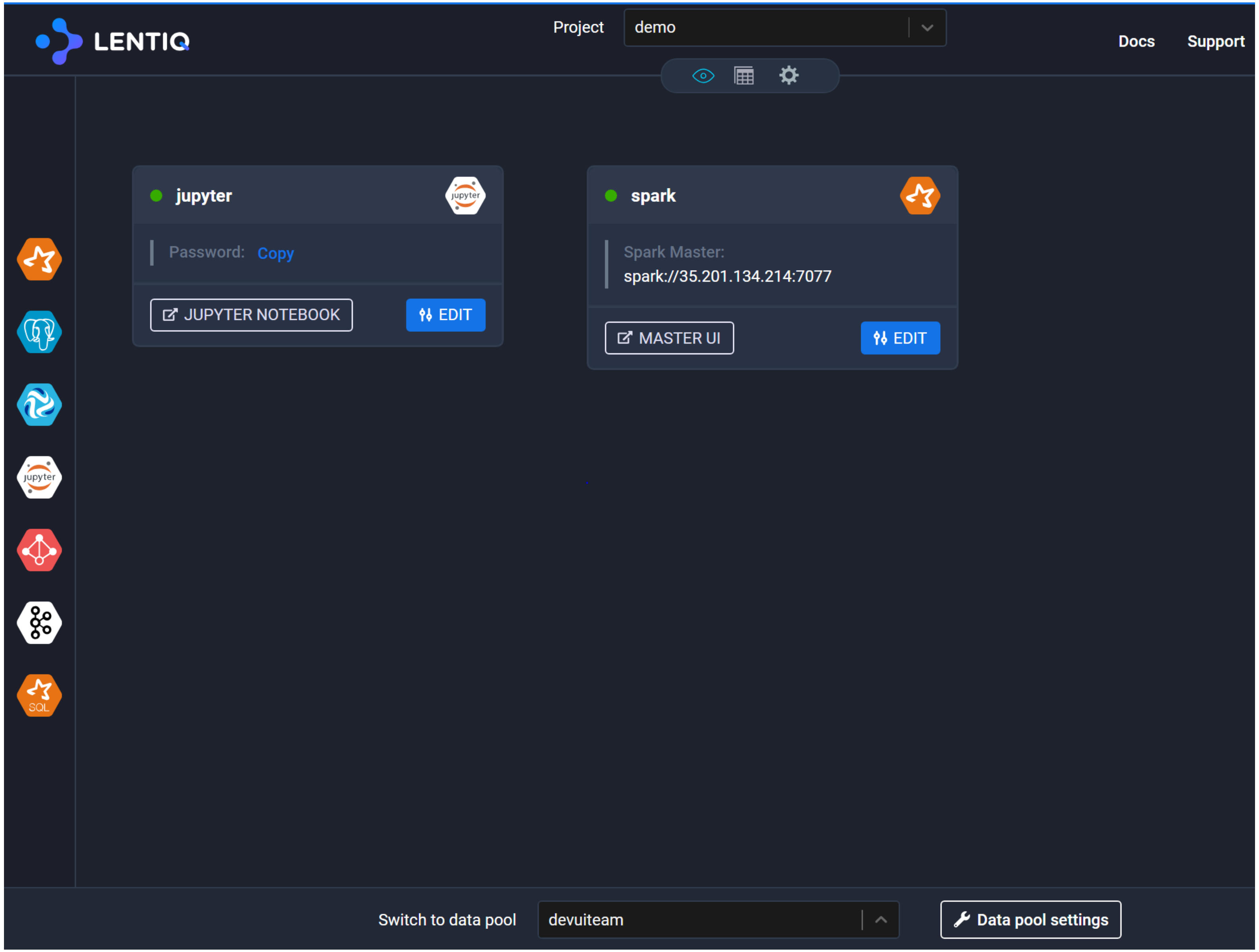
use Kubernetes offered as managed service

on demand use, due to quickness of setup of cluster + applications

benefit from cloud scaling, if needed

LENTIQ
a Bigstep company

Demo



APPS CREATED FROM INTERFACE

LENTIQ
a Bigstep company

```
user@cloudshell:~ (demo)$ kubectl get helmreleases --namespace=demo
NAME               AGE
internal-sparksql  24m
jupyter            17m
spark              12m

user@cloudshell:~ (demo)$ kubectl get statefulsets --namespace=demo
NAME                                      DESIRED  CURRENT  AGE
demo-internal-sparksql-bdl-sparksql-master  1        1        24m
demo-internal-sparksql-bdl-sparksql-worker  1        1        24m
demo-jupyter-bdl-jupyter                    1        1        17m
demo-spark-bdl-spark-master                 1        1        13m
demo-spark-bdl-spark-worker                 1        1        13m

user@cloudshell:~ (demo)$ kubectl get pods --namespace=demo
NAME                                          READY  STATUS   RESTARTS  AGE
demo-internal-sparksql-bdl-sparksql-master-0  2/2    Running  2         24m
demo-internal-sparksql-bdl-sparksql-worker-0  1/1    Running  0         24m
demo-jupyter-bdl-jupyter-0                    1/1    Running  0         18m
demo-spark-bdl-spark-master-0                 2/2    Running  0         13m
demo-spark-bdl-spark-worker-0                 1/1    Running  0         13m
```

LENTIQ
a Bigstep company

```
user@cloudshell:~ (demo)$ kubectl get services --namespace=demo
NAME                                              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                                                   AGE
demo-internal-sparksql-bdl-sparksql-master        ClusterIP      None            <none>           4040/TCP,8080/TCP,7077/TCP,10000/TCP                      24m
demo-internal-sparksql-bdl-sparksql-master-public LoadBalancer   10.31.252.242   34.80.170.123    80:32438/TCP,4040:30330/TCP,7077:30605/TCP,10000:32755/TCP 24m
demo-internal-sparksql-bdl-sparksql-worker        ClusterIP      None            <none>           8081/TCP                                                  24m
demo-internal-sparksql-bdl-sparksql-worker-public LoadBalancer   10.31.254.85    34.80.191.210    8081:32181/TCP                                            24m
demo-jupyter-bdl-jupyter                          ClusterIP      None            <none>           8888/TCP                                                  18m
demo-jupyter-bdl-jupyter-public                   LoadBalancer   10.31.253.99    34.80.208.250    8888:30753/TCP                                            18m
demo-spark-bdl-spark-master                       ClusterIP      None            <none>           4040/TCP,8080/TCP,7077/TCP                                13m
demo-spark-bdl-spark-master-public                LoadBalancer   10.31.249.147   35.201.134.214   80:31575/TCP,4040:32010/TCP,7077:30910/TCP                13m
demo-spark-bdl-spark-worker                       ClusterIP      None            <none>           8081/TCP                                                  13m
demo-spark-bdl-spark-worker-public                LoadBalancer   10.31.243.213   35.201.202.111   8081:31535/TCP                                            13m

user@cloudshell:~ (demo)$ kubectl get pvc --namespace=demo
NAME                                              STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-demo-internal-sparksql-bdl-sparksql-master-0 Bound    pvc-f37593fa-43e4-11e9-90c0-42010a8c0026   15Gi       RWO            standard       1h
data-demo-internal-sparksql-bdl-sparksql-worker-0 Bound    pvc-f376aa77-43e4-11e9-90c0-42010a8c0026   15Gi       RWO            standard       1h
data-demo-jupyter-bdl-jupyter                     Bound    pvc-d9e302f5-43e5-11e9-90c0-42010a8c0026   15Gi       RWO            standard       1h
data-demo-spark-bdl-spark-master-0                Bound    pvc-7e301d28-43e6-11e9-90c0-42010a8c0026   15Gi       RWO            standard       1h
data-demo-spark-bdl-spark-worker-0                Bound    pvc-7e314394-43e6-11e9-90c0-42010a8c0026   15Gi       RWO            standard       1h
```

LENTIQ
a Bigstep company

UPLOAD DATA VIA JUPYTER

Not secure | 34.80.208.250:8888/notebooks/Getting%20Started%20Guide.ipynb

jupyter Getting Started Guide (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

How to upload data to the datapool from the Jupyter filesystem

The bdl utility will use '/' by default as a destination if none is specified which is in turn short for **bdl://datalake_name.project_name/**

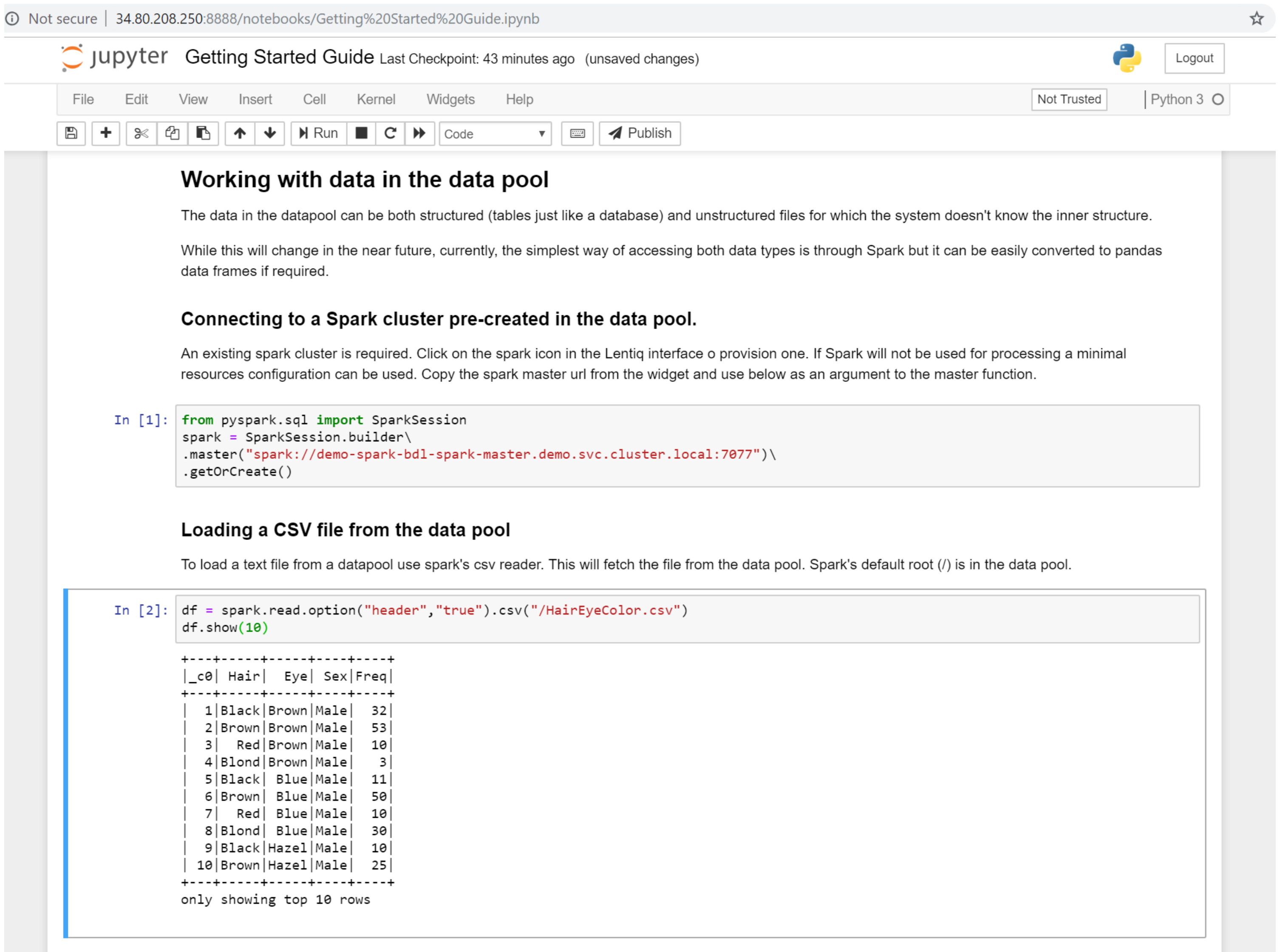
In [3]:

```
#download some test data onto the Jupyter's file system
!wget -q https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/csv/datasets/HairEyeColor.csv
#use the bdlcl put command to upload data the -f means force overwrite
!bdl -put -f HairEyeColor.csv /
!bdl -ls /
```

```
0    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Mar 11, 2019 11:22:12 AM com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase <clinit>
INFO: GHFS version: hadoop2-1.9.7
Mar 11, 2019 11:22:14 AM com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase configure
WARNING: No working directory configured, using default: 'gs://project-devuiteam-demo/'
0    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Mar 11, 2019 11:22:18 AM com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase <clinit>
INFO: GHFS version: hadoop2-1.9.7
Mar 11, 2019 11:22:20 AM com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase configure
WARNING: No working directory configured, using default: 'gs://project-devuiteam-demo/'
Found 1 items
-rwx------   3 root root       1005 2019-03-11 11:22 /HairEyeColor.csv
```

LENTIQ a Bigstep company

CONNECT TO SPARK AND RUN QUERY



Not secure | 34.80.208.250:8888/notebooks/Getting%20Started%20Guide.ipynb

jupyter Getting Started Guide Last Checkpoint: 43 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3

Run Code Publish

Working with data in the data pool

The data in the datapool can be both structured (tables just like a database) and unstructured files for which the system doesn't know the inner structure.

While this will change in the near future, currently, the simplest way of accessing both data types is through Spark but it can be easily converted to pandas data frames if required.

Connecting to a Spark cluster pre-created in the data pool.

An existing spark cluster is required. Click on the spark icon in the Lentiq interface o provision one. If Spark will not be used for processing a minimal resources configuration can be used. Copy the spark master url from the widget and use below as an argument to the master function.

In [1]:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder\
.master("spark://demo-spark-bdl-spark-master.demo.svc.cluster.local:7077")\
.getOrCreate()
```

Loading a CSV file from the data pool

To load a text file from a datapool use spark's csv reader. This will fetch the file from the data pool. Spark's default root (/) is in the data pool.

In [2]:

```
df = spark.read.option("header","true").csv("/HairEyeColor.csv")
df.show(10)
```

```
+---+-----+-----+----+----+
|_c0| Hair|  Eye| Sex|Freq|
+---+-----+-----+----+----+
|  1|Black|Brown|Male|  32|
|  2|Brown|Brown|Male|  53|
|  3|  Red|Brown|Male|  10|
|  4|Blond|Brown|Male|   3|
|  5|Black| Blue|Male|  11|
|  6|Brown| Blue|Male|  50|
|  7|  Red| Blue|Male|  10|
|  8|Blond| Blue|Male|  30|
|  9|Black|Hazel|Male|  10|
| 10|Brown|Hazel|Male|  25|
+---+-----+-----+----+----+
only showing top 10 rows
```

LENTIQ
a Bigstep company

SPARK UI SHOWS CONNECTION FROM JUPYTER

Not secure | 35.201.134.214/proxy:demo-spark-bdl-spark-master.demo.svc.cluster.local:8080

Spark 2.4.0 **Spark Master at spark://demo-spark-bdl-spark-master.demo.svc.cluster.local:7077**

URL: spark://demo-spark-bdl-spark-master.demo.svc.cluster.local:7077
Alive Workers: 1
Cores in use: 1 Total, 1 Used
Memory in use: 1024.0 MB Total, 1024.0 MB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190311101512-10.28.1.41-8581	10.28.1.41:8581	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duratior
app-20190311120356-0000 (kill)	pyspark-shell	1	1024.0 MB	2019/03/11 12:03:56	root	RUNNING	26 s

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

LENTIQ
a Bigstep company

Q&A

Test how apps work on Kubernetes (and more), by testing EdgeLake @

datalake.lentiq.com

LENTIQ
a Bigstep company

Text resources

https://techcrunch.com/2016/10/16/wtf-is-a-container/
https://linuxacademy.com/blog/containers/history-of-container-technology/
https://jpetazzo.github.io/2017/02/24/from-dotcloud-to-docker/
https://www.fasthosts.co.uk/blog/cloud/kubernetes-vs-docker-swarm-vs-apache-mesos
https://www.infoq.com/presentations/kubernetes-stateful-containers
"Kubernetes Up & Running" - Brendan Burns, Kelsey Hightower & Joe Beda
"Mastering Kubernetes" - Gigi Sayfan

LENTIQ
a Bigstep company

Image resources

[1] https://rominirani.com/learning-docker-move-to-the-cloud-3326369300ad
[2] https://docs.docker.com/get-started/
[3] https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/
[4] https://thenewstack.io/happens-use-java-1960-ibm-mainframe/
[5] https://blog.docker.com/2017/10/least-privilege-container-orchestration/
[6] https://rancher.com/comparing-rancher-orchestration-engine-options/
[7] https://softwareengineeringdaily.com/2019/01/11/why-is-storage-on-kubernetes-is-so-hard/
[8] https://medium.com/@jessgreb01/digging-into-docker-layers-c22f948ed612
[9] https://www.bluedata.com/blog/2018/07/operation-stateful-bluek8s-and-kubernetes-director/kubernetes-reconciliation-loop/
[10] https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/
[11] https://github.com/helm/helm
[12] https://blog.openshift.com/make-a-kubernetes-operator-in-15-minutes-with-helm/

LENTIQ
a Bigstep company