CERN openlab

DEVELOPMENT OF HIGH-PERFORMANCE BIOMEDICAL SIMULATION SOFTWARE AS A SERVICE - BIODYNAMO NOTEBOOK SERVICE

Berina Bandić

Mentor: Ahmad Hesam

Sarajevo

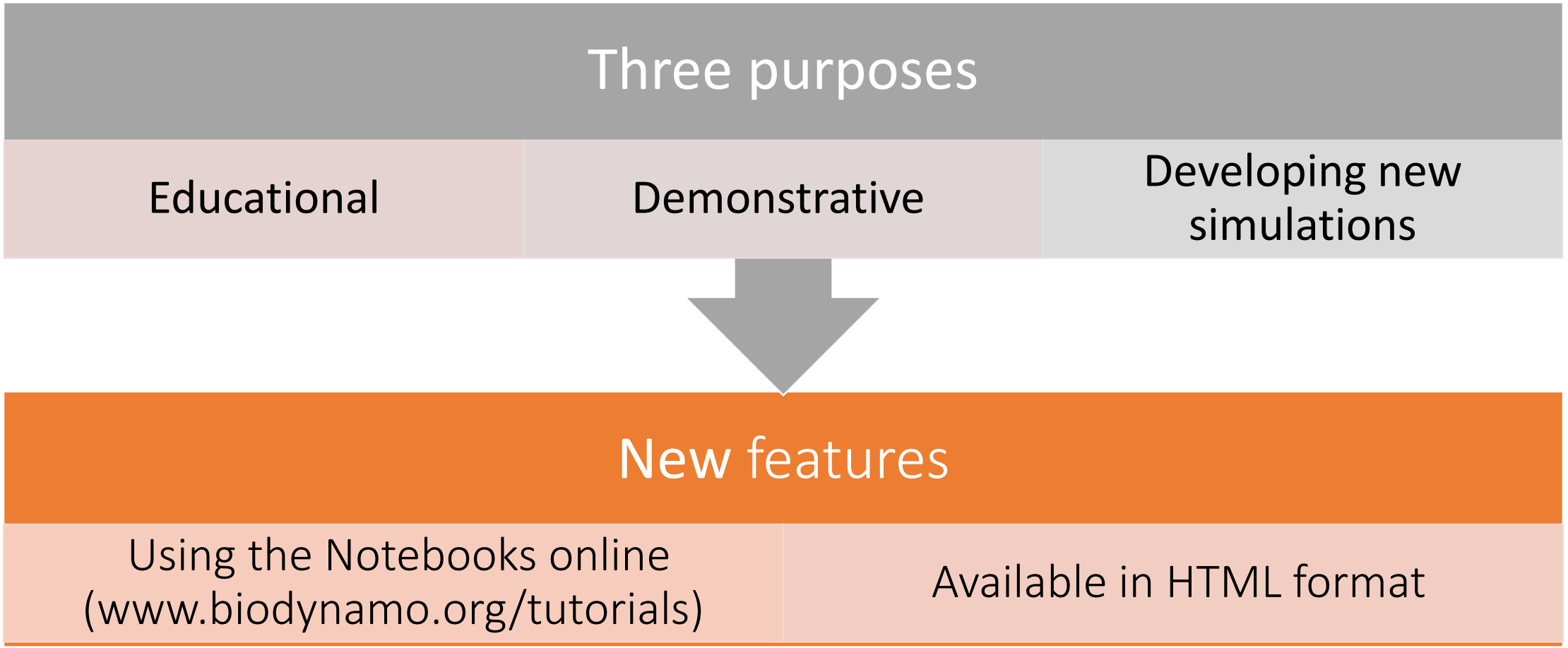
24/09/2020

Berina Bandić

BioDynaMo Framework

- Open-source C++ framework where life scientists can easily create, run, and visualize 3D agent-based biological simulations
- Supported by ROOT framework
- C++ API
- BioDynaMo Notebooks

BioDynaMo Notebooks

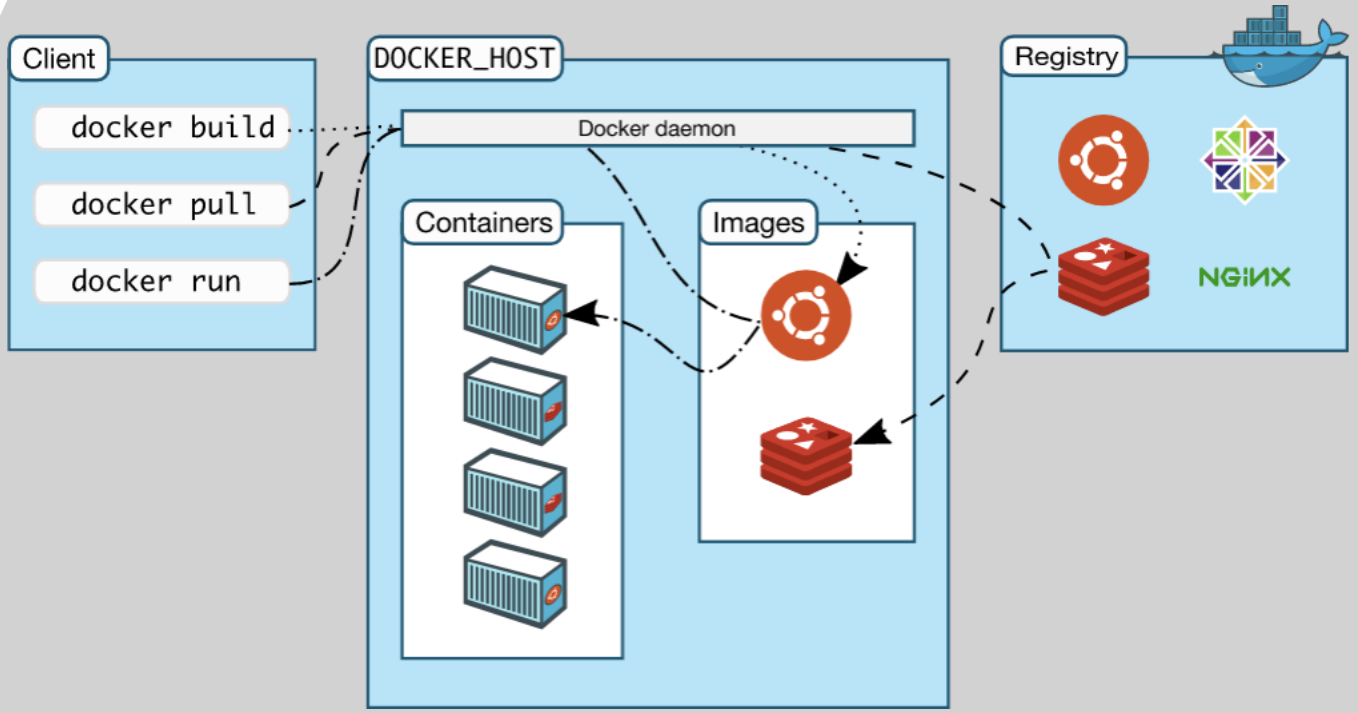


Technologies

- *Jupyter Notebook*
- *JupyterLab*
- *ROOT*
- *GatsbyJS*
- *Binder*
- *Docker*

Docker

- A tool which enables the delivery of software in a form of packages called containers
- A client-server architecture
- Docker image



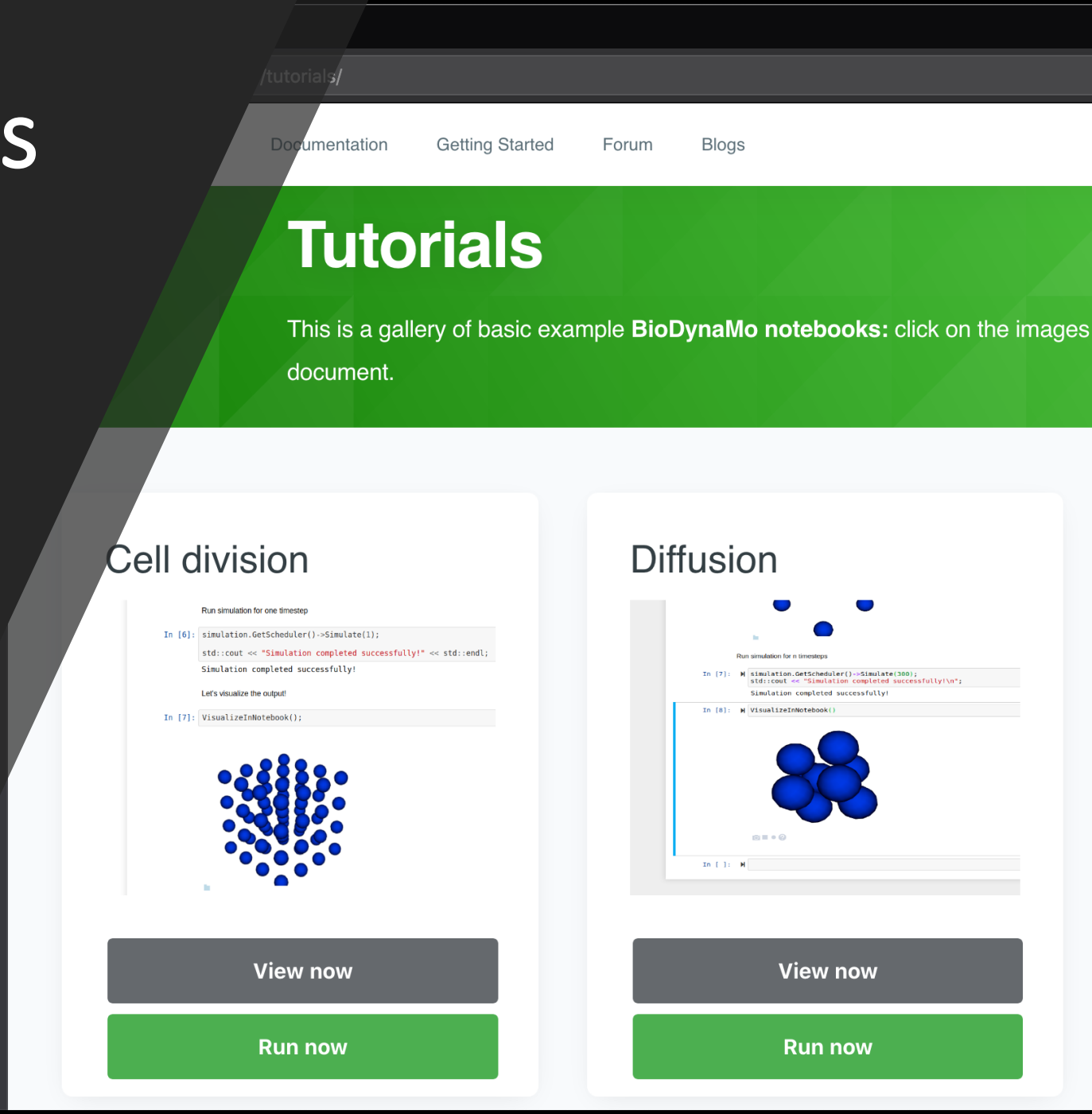
- *A tool which can turn Jupyter Notebooks from GitHub repositories into interactive notebooks*
- *Easy to use*
- *The process of building the Docker image can take up too much time*

Docker setup

BioDynaMo Docker image on DockerHub	Linux Ubuntu as a base image
	Has BioDynaMo framework installed
	Everything's prepared so that the BioDynaMo Notebooks service can work properly
Another Docker image which uses the BioDynamo Docker image as a base image	Default user for the system created
	The existing notebooks with some required C++ libraries that come with some of these notebooks copied
	Additional scripts which enable using ROOT C++ kernel with Ipython notebooks

BioDynaMo Tutorials Dashboard

- *'Tutorial cards' made with ReactJS*
- *Querying files in static folder with 'gatsby-source-filesystem' plugin (using GraphQL)*
- *'View now' button for Notebooks in HTML (for users who only want to see visualisation ASAP with no wish to execute any commands)*
- *''Run now' button to run BioDynaMo Notebooks with Binder*



Diffusion

This model creates 8 cells at each corner of a cube, and one in the middle. The cell in the middle secretes a substance. The cells are modeled to move according to the extracellular gradient; in this case to the middle.

In [1]:
```
%jsroot on
gROOT->LoadMacro("${BDMSYS}/etc/rootlogon.C");
```

In [2]:
```
#include <vector>
#include "biodynamo.h"
#include "diffusion_biology_modules.h"
```

Initialize biodynamo

In [3]:
```
Simulation simulation("diffusion");

auto construct = [](const Double3& position) {
  Cell* cell = new Cell(position);
  cell->SetDiameter(30);
  cell->SetMass(1.0);
  cell->AddBiologyModule(new Chemotaxis());
  Double3 secretion_position = {{50, 50, 50}};
  if (position == secretion_position) {
    cell->AddBiologyModule(new KaliumSecretion());
  }
  return cell;
};
std::vector<Double3> positions;
positions.push_back({0, 0, 0});
positions.push_back({100, 0, 0});
positions.push_back({0, 100, 0});
positions.push_back({0, 0, 100});
positions.push_back({0, 100, 100});
positions.push_back({100, 0, 100});
positions.push_back({100, 100, 0});
positions.push_back({100, 100, 100});
```

The cell responsible for secretion

In [4]:
```
positions.push_back({50, 50, 50});
ModelInitializer::CreateCells(positions, construct);
```

Define the substances that cells may secrete

In [5]:
```
ModelInitializer::DefineSubstance(kKalium, "Kalium", 0.4, 0, 25);
```

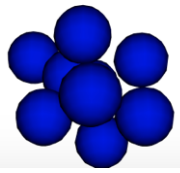
Run simulation for n timesteps

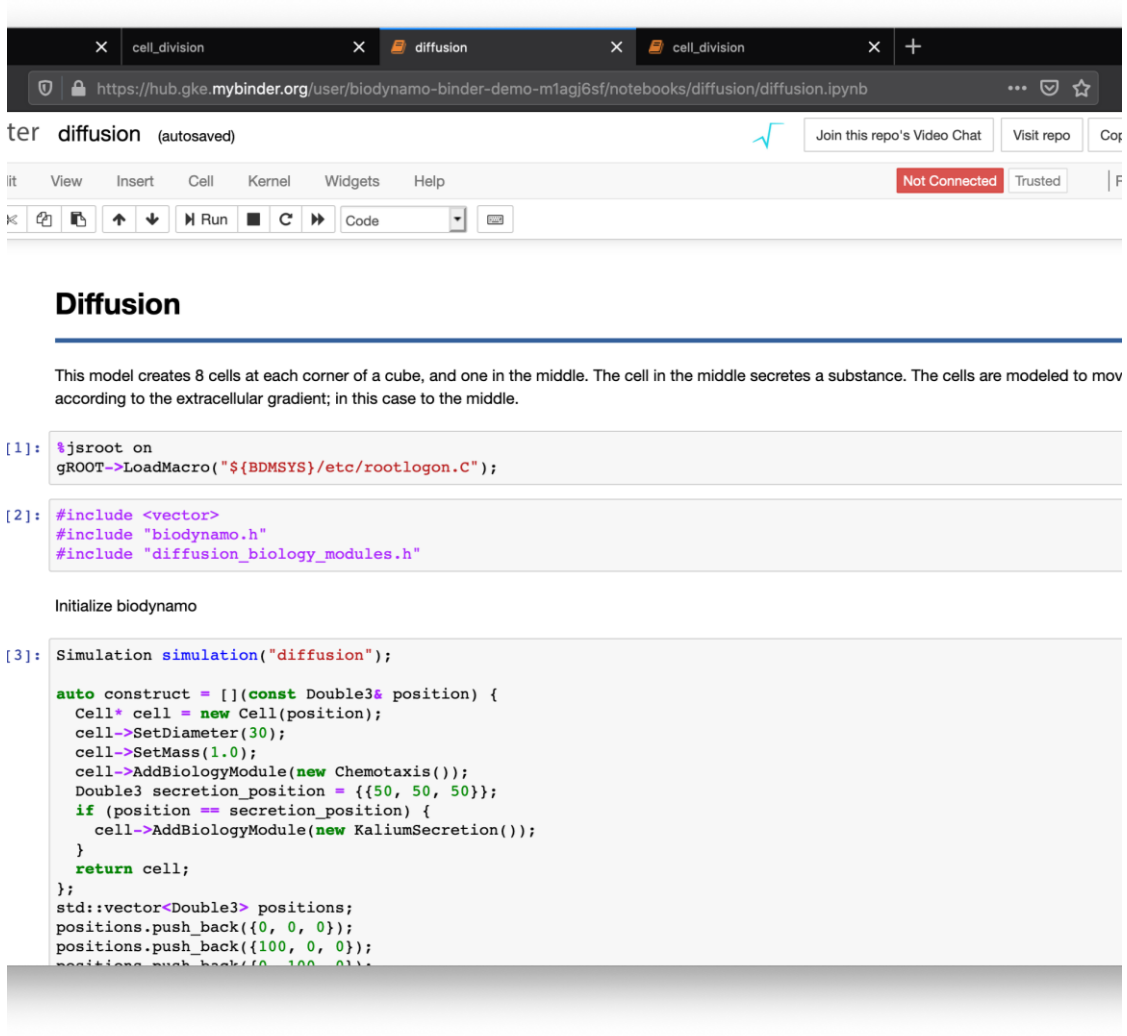
In [6]:
```
simulation.GetScheduler()->Simulate(300);
std::cout << "Simulation completed successfully!\n";
```

```
Simulation completed successfully!
```

Let's visualize the output!

In [7]:
```
VisualizeInNotebook();
```





Conclusion

Thank you for your attention!

QUESTIONS?

Contact

berina.bandic@gmail.com

https://www.linkedin.com/in/berinabandic/