Crops in silico Documentation

Mike Lambert, Craig Willis

May 01, 2019

Contents:

CHAPTER 1

Overview

Crops in silico is an integrative and multi-scale modeling platform designed to combine modeling efforts toward the generation of virtual crops.

The Crops *in silico* platform consists of:

- Model integration framework (`cis_interface`)
- Model composer and execution framework
- Model catalog

1.1 Model integration framework: cis_interface

The `cis_interface` model integration framework provides support for combining scientific models written in different programming languages. To combine two models, modelers add simple communications interfaces to the model code and provide simple declarative specification files that identfy the models that should be run and the inputs and outputs those models expect.

For more information, see the documentation

1.2 Model composer and execution framework

The Model Composer and execution framework is available to the Cis community at https://cropsinsilico.ndslabs.org.

The Model Composer is a tool for composing systems of models built using the `cis_interface` framework. For documentation on using the Model Composer UI, please see the User Guide. For documentation on developing the Model Composer UI, please see the Developer Guide

1.3 Model catalog

The model catalog is a GitHub repository containing community contributed model definitions and predefined systems of models as graphs. Models in this repository are available to all users of the platform. Models are contributed through GitHub issues and pull requests.

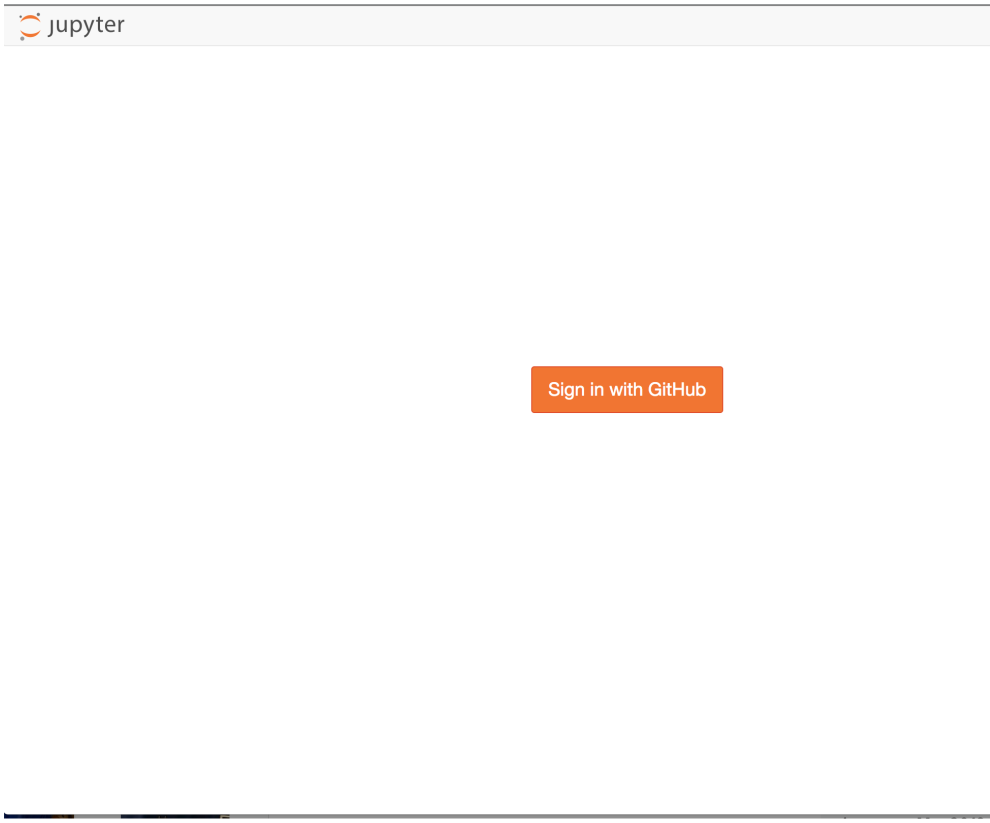
CHAPTER 2

Getting started

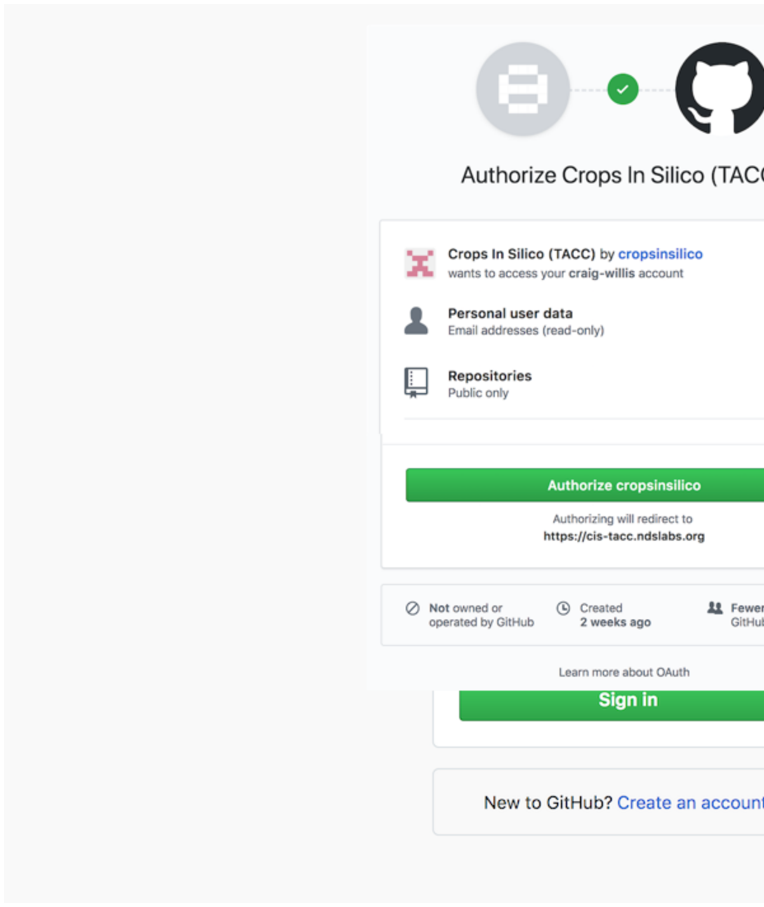
The Crops *in silico* model composer is based on the popular JupyterLab web-based interactive analysis and development environment. JupyterLab provides a complete development environment for the creation and execution of models using the `cis_interface` integration framework. The model composer is a JupyterLab plugin that supports a visual-programming approach to model composition.

2.1 Logging in

To access the web service, go to https://cropsinsilico.ndslabs.org. The CiS model composer uses GitHub for authentication. You will be prompted to sign-in via GitHub and to authorize CiS to access information about you. username.



Click on the “Sign in with GitHub” button:



Authorize Crops In Silico (TAC

Crops In Silico (TACC) by cropsinsilico
wants to access your **craig-willis** account

Personal user data
Email addresses (read-only)

Repositories
Public only

Authorizing will redirect to
https://cis-tacc.ndslabs.org

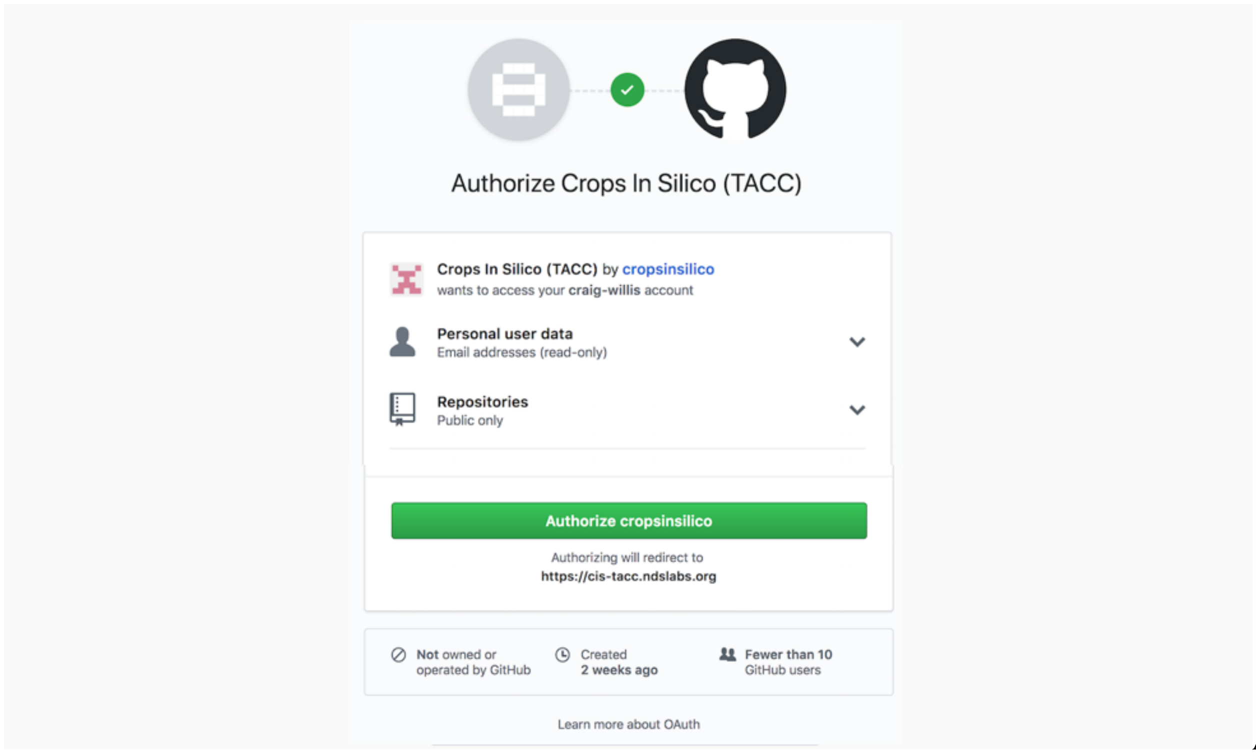
Not owned or operated by GitHub

Created **2 weeks ago**

Fewer GitHu

Enter your GitHub credentials or create an account, if necessary:
login

If prompted, authorize “Crops in silico” to access your profile information:


Authorize Crops In Silico (TACC)

Crops In Silico (TACC) by cropsinsilico
wants to access your craig-willis account

Personal user data
Email addresses (read-only)

Repositories
Public only

Authorizing will redirect to
https://cis-tacc.ndslabs.org

Not owned or operated by GitHub

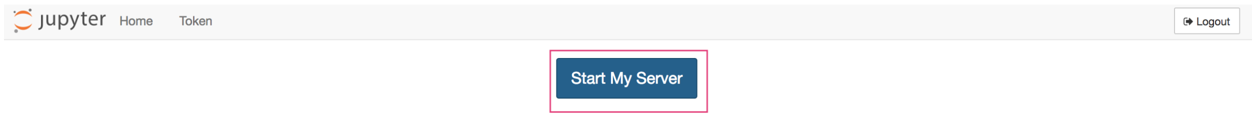
Created 2 weeks ago

Fewer than 10 GitHub users

Authorize CiS

2.2 Starting JupyterLab

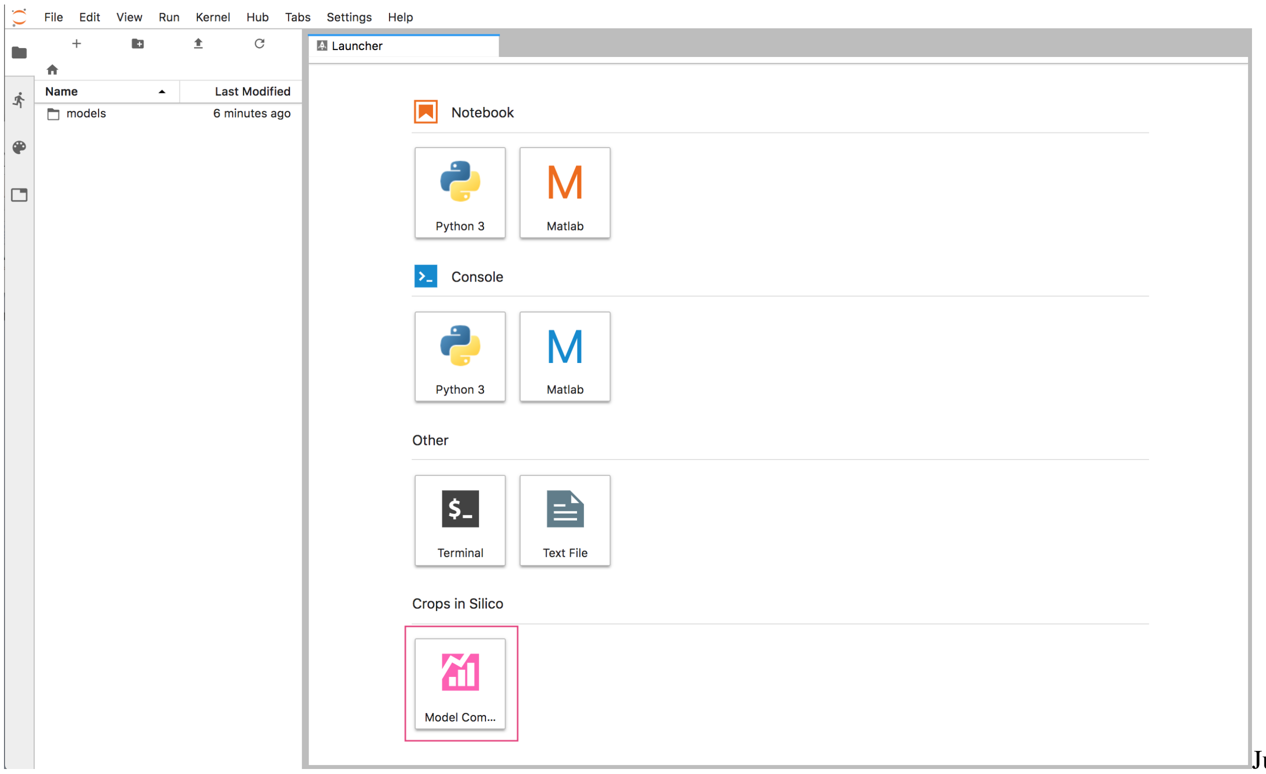
Once logged in, you will be able to start your JupyterLab instance. Select “Start My Server”:



Start server

2.3 Launching the Model Composer

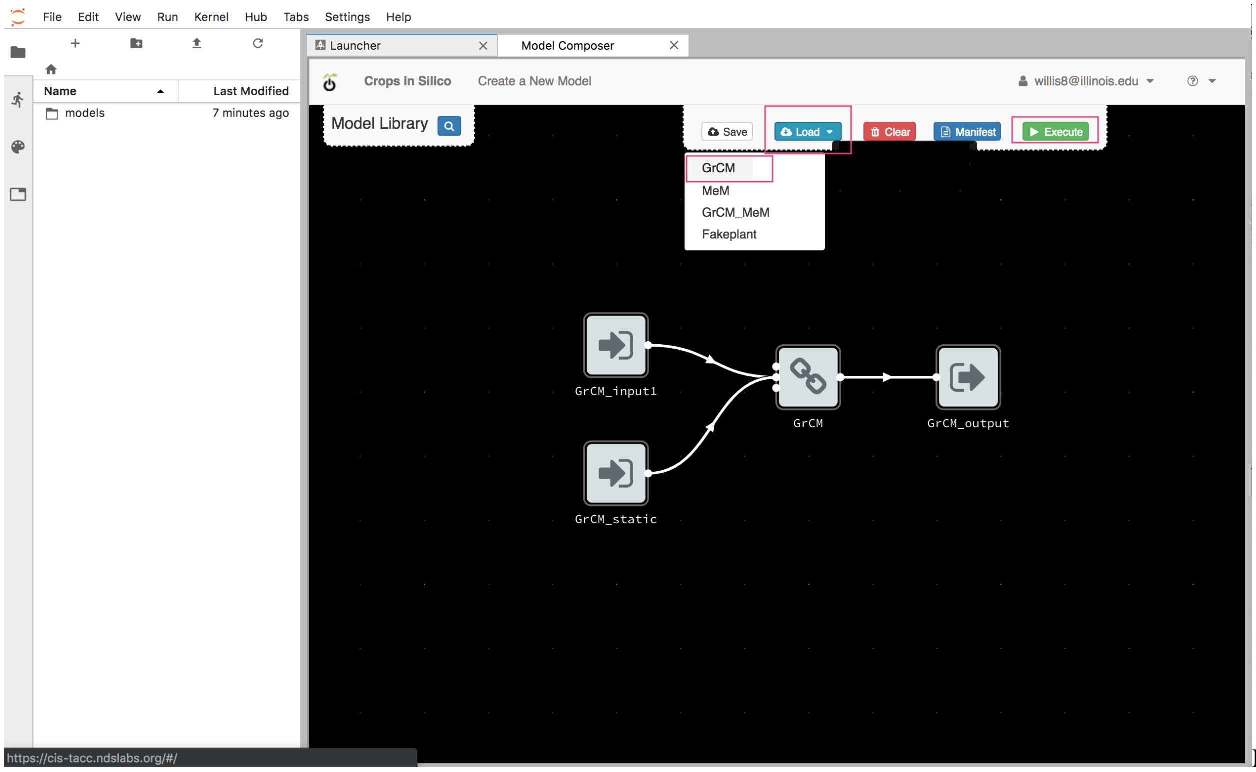
Once started, you will see the JupyterLab launcher. This JupyterLab environment is a complete development environment for use with the `cis_interface` framework. You can create notebooks, execute commands from the terminal, or launch the model composer. Select the “Model Composer” icon:



JupyterLab launcher

2.4 Loading an Existing Model Graph

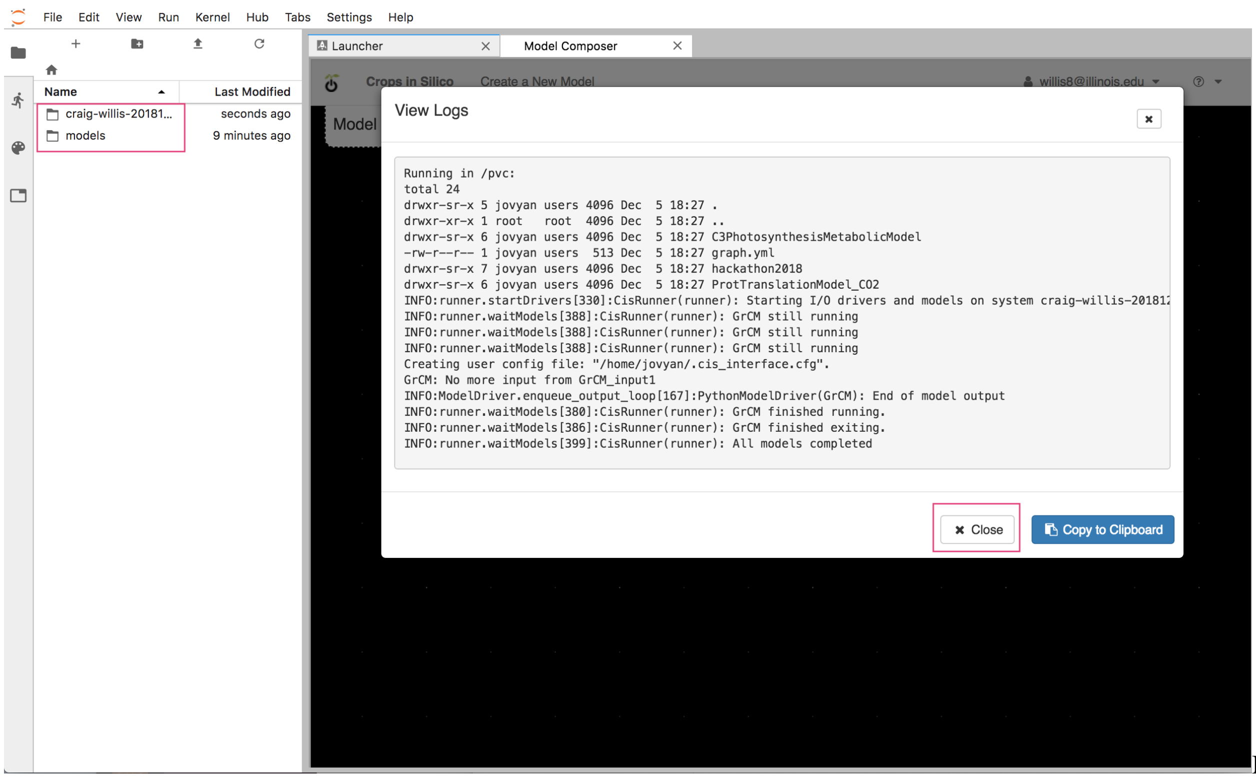
The model composer allows to to create and use models from the model library to compose execution graphs. Select the "Load" button to load the "GrCM" model: and



Model composer

2.5 Executing the Graph

Select "Execute" to run the model. Model output will be displayed on screen and written to an model output directory for further exploration:

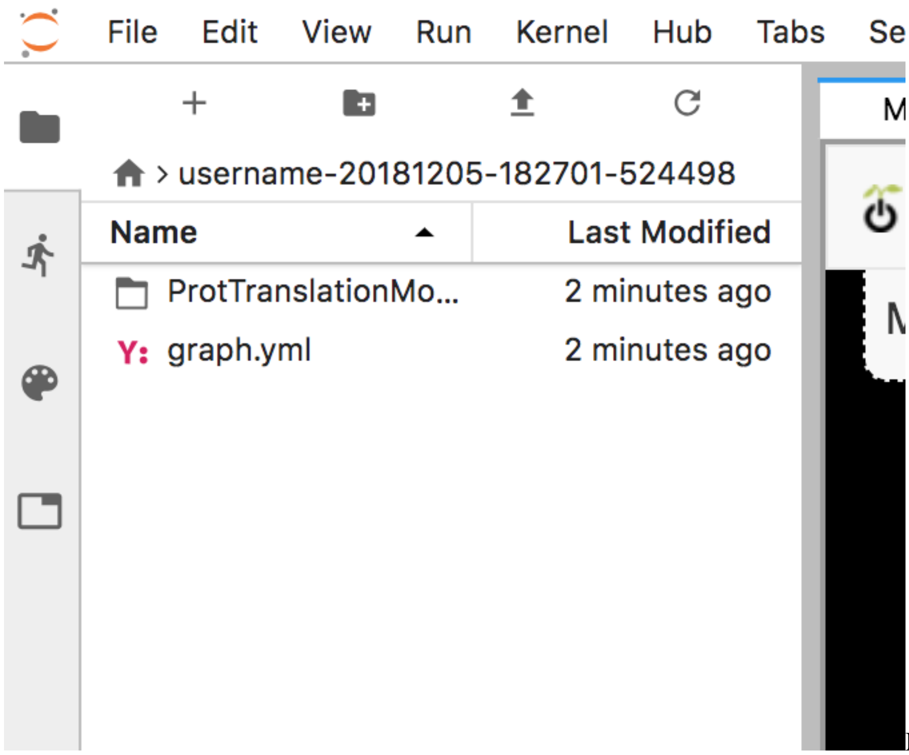


```
Running in /pvc:
total 24
drwxr-sr-x 5 jovyan users 4096 Dec  5 18:27 .
drwxr-xr-x 1 root   root  4096 Dec  5 18:27 ..
drwxr-sr-x 6 jovyan users 4096 Dec  5 18:27 C3PhotosynthesisMetabolicModel
-rw-r--r-- 1 jovyan users  513 Dec  5 18:27 graph.yml
drwxr-sr-x 7 jovyan users 4096 Dec  5 18:27 hackathon2018
drwxr-sr-x 6 jovyan users 4096 Dec  5 18:27 ProtTranslationModel_CO2
INFO:runner.startDrivers[330]:CisRunner(runner): Starting I/O drivers and models on system craig-willis-20181
INFO:runner.waitModels[388]:CisRunner(runner): GrCM still running
INFO:runner.waitModels[388]:CisRunner(runner): GrCM still running
INFO:runner.waitModels[388]:CisRunner(runner): GrCM still running
Creating user config file: "/home/jovyan/.cis_interface.cfg".
GrCM: No more input from GrCM_input1
INFO:ModelDriver.enqueue_output_loop[167]:PythonModelDriver(GrCM): End of model output
INFO:runner.waitModels[380]:CisRunner(runner): GrCM finished running.
INFO:runner.waitModels[386]:CisRunner(runner): GrCM finished exiting.
INFO:runner.waitModels[399]:CisRunner(runner): All models completed
```

Model logs

2.6 Viewing Execution Output

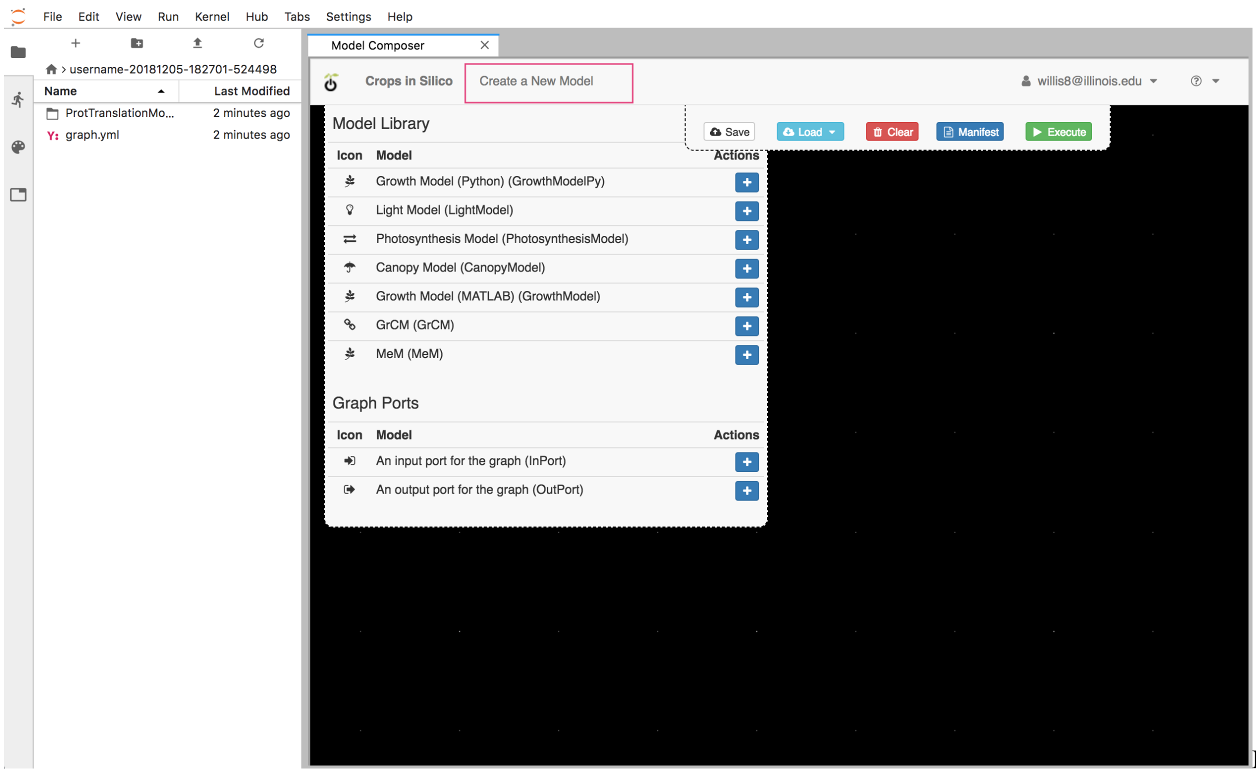
The model output directory contains the graph, model source, inputs and outputs used during execution:



Model output

2.7 Using the Model Library

The model library lists all official models approved by the community as well as any private models you have developed and added to the system. You can use the library to compose new graphs using this user interface:



username-20181205-182701-524498

Name	Last Modified
ProtTranslationMo...	2 minutes ago
graph.yml	2 minutes ago

Model Library

Icon	Model	Actions
	Growth Model (Python) (GrowthModelPy)	+
	Light Model (LightModel)	+
	Photosynthesis Model (PhotosynthesisModel)	+
	Canopy Model (CanopyModel)	+
	Growth Model (MATLAB) (GrowthModel)	+
	GrCM (GrCM)	+
	MeM (MeM)	+

Graph Ports

Icon	Model	Actions
	An input port for the graph (InPort)	+
	An output port for the graph (OutPort)	+

Model libraryl

2.8 Next step

For more detailed usage information, please see the User Guide.

CHAPTER 3

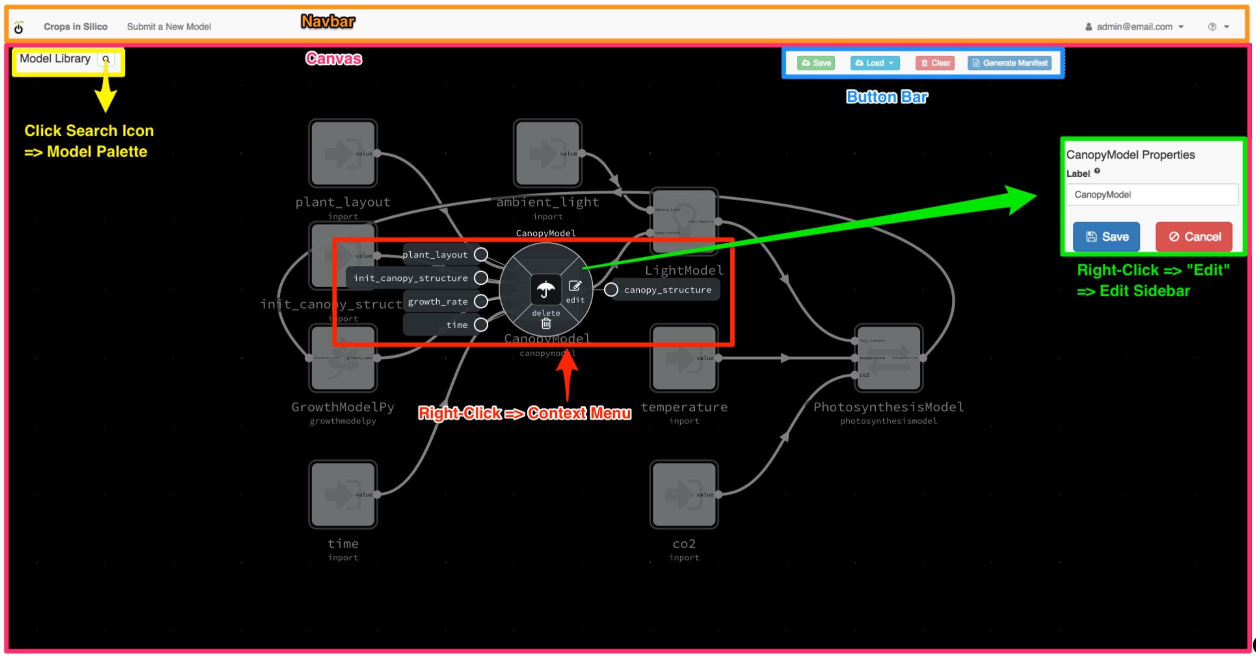
User Guide

Welcome to the User's Guide for the Crops *in silico* Model Composer UI. The purpose of this document is to describe the usage of the User Interface. If any of the steps described in this document are unclear or confusing, please direct your questions to Crops *in silico* Support

To assist the Support Team in their investigation, please be sure to include the browser type/version and operating system that you were using when the troublesome behavior was encountered.

3.1 Table of Contents

3.2 Orientation



Orientaton

The Model Composer UI consists of several smaller components:

- Navbar
- Canvas (aka "The Graph")

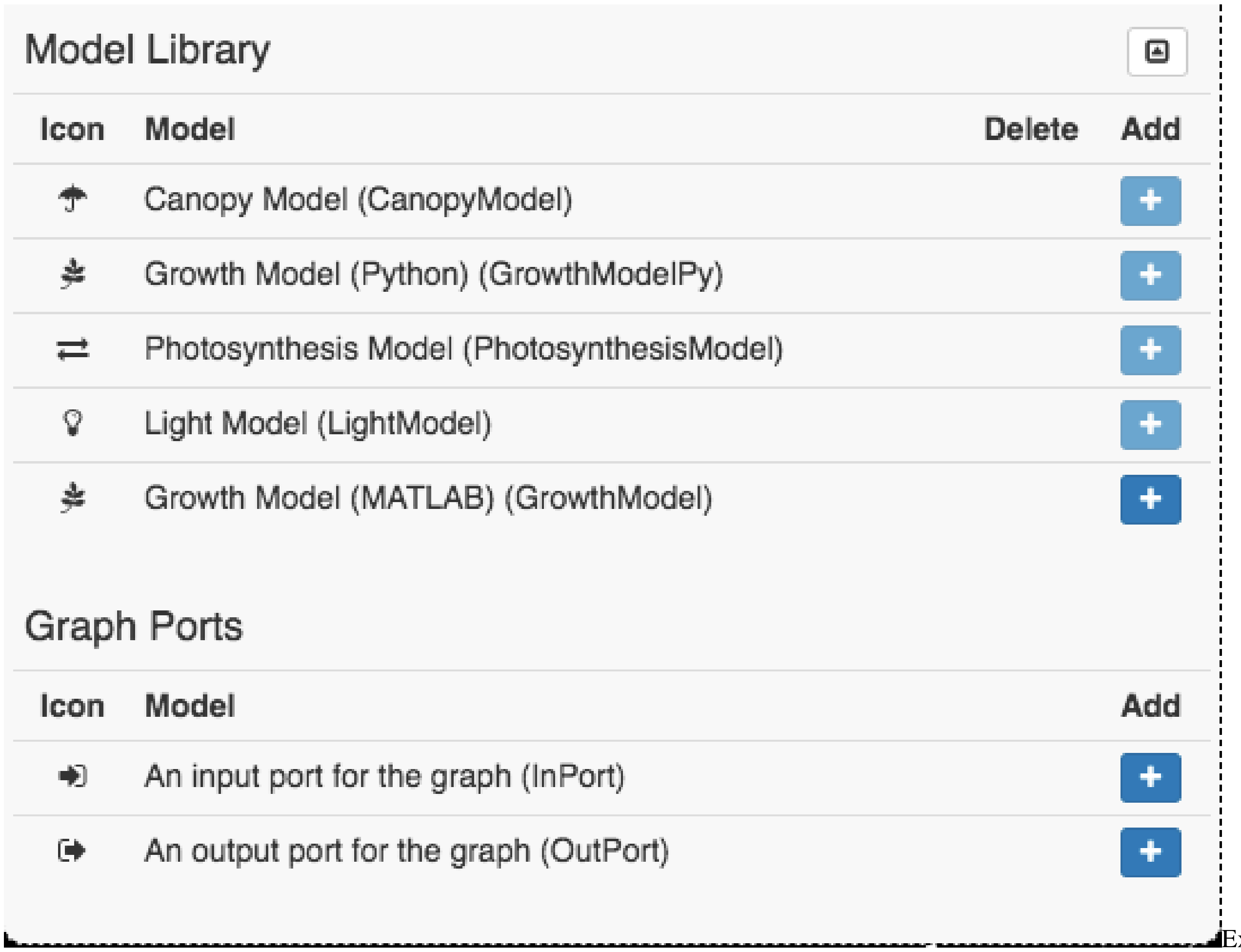
The *Navbar* runs along the top of the UI and contains the Crops *in silico* brand name and logo. On the left side, you will find a link to submit a new model. On the right side, you will also find the `Log In` button and a `Help` dropdown which contains links to this user guide, the developer's guide, and the documentation for the `cisrun` CLI tool.

The *Canvas* is the grid with the black background in the center of the screen. This displays the current state of the graph that the user is working on.

In front of the Canvas, there are also a few helpful floating windows:

- Model Library (aka "Palette")
- Button Bar
- Context Menu + Edit Sidebar

The *Model Library* starts collapsed, and can be found on the top-left of the canvas. This consists of a simple table listing of the existing models that the system knows about. Here we can also add InPorts and OutPorts to our graph. Simply click the `Add` (+) button and the new node will appear in the Canvas. If the node appears off-screen, you can easily find it by pressing the `F` key to focus on the entire graph. For models that you have created, here you can also find the option to delete them (see Creating a New Model).


Model Library

Icon	Model	Delete	Add
	Canopy Model (CanopyModel)		+
	Growth Model (Python) (GrowthModelPy)		+
	Photosynthesis Model (PhotosynthesisModel)		+
	Light Model (LightModel)		+
	Growth Model (MATLAB) (GrowthModel)		+

Graph Ports

Icon	Model	Add
	An input port for the graph (InPort)	+
	An output port for the graph (OutPort)	+

Expanded Model Palette

The *Button Bar* runs along the top-right of the canvas. It contains actions such as `Save`, `Load`, `Clear`, and `Generate Manifest`. `Save` will store the state of your current graph to the database, while `Load` will import a given state from the database. `Clear` will remove all nodes and edges from the current canvas, but does not affect graphs that have been stored in the database. `Generate Manifest` will convert the current graph on the canvas to the format that is required by the `cisrun` CLI.

The *Context Menu* appears when an entity in the canvas is right-clicked. It appears as a round menu that centers on where the mouse click occurred. `Edit` and `Delete` are offered the only actions offered, and only when right-clicking on Edges or Nodes (including InPorts and OutPorts).

If `Edit` is chosen, the Edit Sidebar will be displayed on the right edge of the canvas. The sidebar allows you to edit the metadata of the entity you have selected. This sidebar will also appear when adding an InPort or OutPort to specify the source/destination of the data.

3.3 Canvas Controls

The Canvas accepts a few mouse/keyboard inputs:

- `F`: Auto-focus the graph so that you can see all elements
- `Left-click`: Select a port on a node
- `Left-click (hold)`: Drag a node around the canvas

- `Right-click`: Display the context-menu for the clicked graph element

3.4 Loading an Example Graph

The Button Bar at the top-right should offer a `Load` button. Clicking on this button will expand a dropdown list of all saved graphs and examples currently accessible. Choosing one of these examples will load its contents into your Canvas, allowing you to visualize and edit the graph before generating a manifest for use with `cisrun`.

3.5 Adding a Node

The Model Library on the left side offers an "Add" button beside each model. Click this button to add a new node to the canvas representing the model you've chosen. Once added, you can left-click this node and hold to drag it around the canvas. On the new node, you should see grey dots on the left/right sides - these are the inputs (left side) and outputs (right side) that this model accepts.

For more details, see the documentation for Model file input/output

3.5.1 About the Bounding Box

While dragging, you may notice a lighter gray circle around the dragged node. This is the "bounding box" that indicates valid areas where your node can be dragged - as your node moves the bounding box moves with it! This behavior is meant to prevent you from dragging a node so fast that the mouse cursor exits the node before the "drag" event is picked up.

3.6 Adding an Edge

To create an edge, simply click on an output on a model - this should display a floating edge that ends at your cursor. Then, select an input on another model. You can also choose an input first and connect that to an output.

NOTE: Inputs can only be connected to outputs and vice versa. An input cannot be connected to another input. An output cannot be connected with another output.

For more details, see the documentation for Model-to-model communication

3.7 Adding an InPort / OutPort

While each node has its own set of inputs and outputs, the entire graph likely will need InPorts/OutPorts of its own. The Model Palette at the left of the view offers buttons to add these InPorts and OutPorts.

3.8 Right-Click Context Menu

In general, right-clicking an entity on the canvas will bring up a context menu for the clicked object. We currently support right-click operations on Nodes, InPorts/OutPorts, and Edges.

3.8.1 Nodes

Right-clicking a node will bring up the context menu, and allows you to `Delete` the node or `Edit` its metadata:

- Label (optional): The identifier for this node that will appear in the canvas

You should also see the model inputs/ouputs displayed with the context menu open. Just to the left of the context menu, you should see any inputs offered by this node. To the right, you will see its outputs. This allows you to easily select an input or output and connect it to another model.

3.8.2 InPorts / OutPorts

Right-clicking an InPort or OutPort will bring up the context menu, and allows you to delete it or edit its metadata:

- Label (optional): The identifier for this node that will appear in the canvas
- Type (required): The type of this port - this can be either "File" or "Queue" (if using AMQP queues)
- Value (required): The value of this port - this will either be a filepath (for Type=File) or a queue name (for Type=Queue)
- Read/Write Method (optional): how are the contents formatted? can be any of 'table', 'table_array', 'pandas' or 'line'

3.8.3 Edges

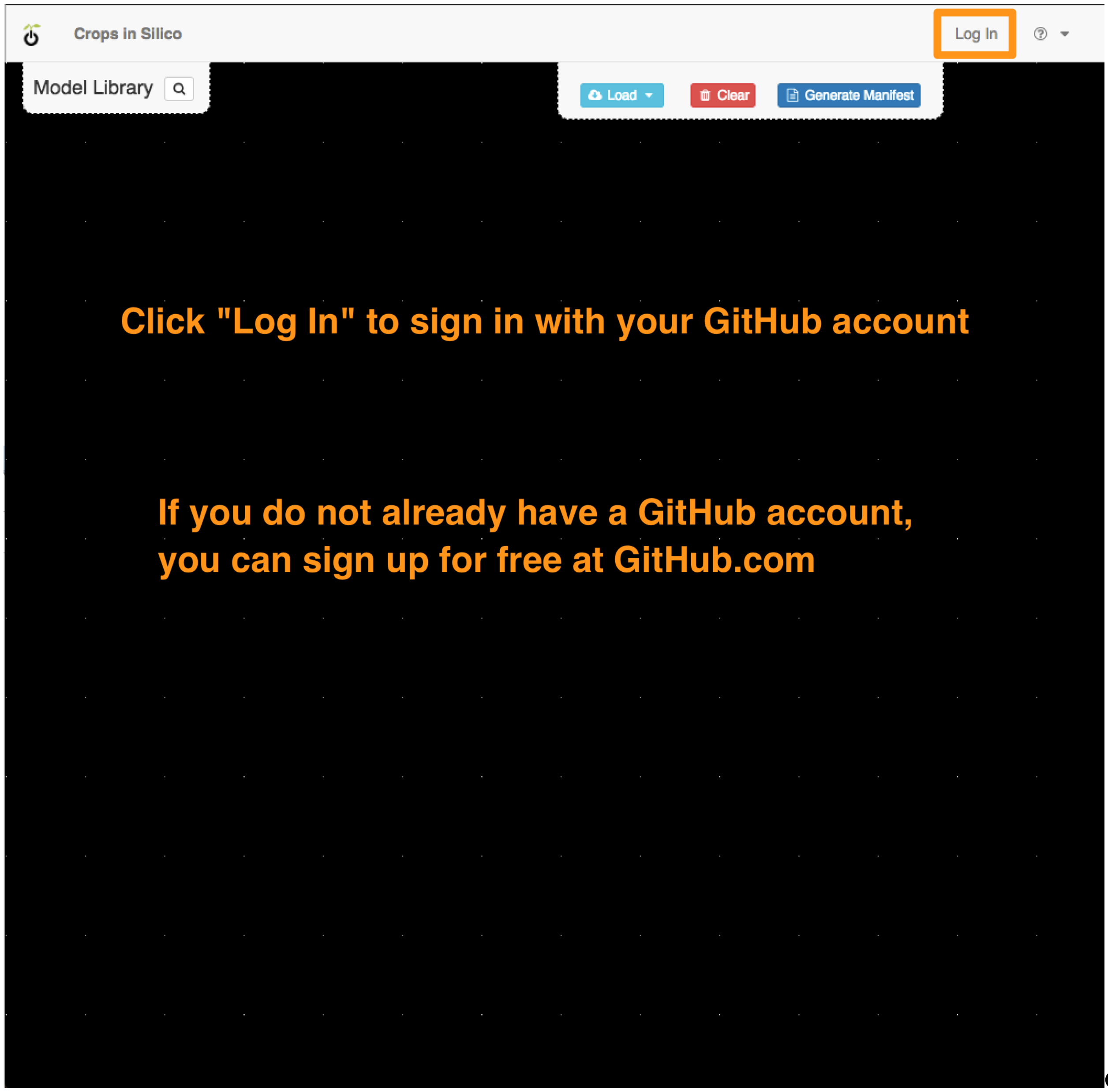
Right-clicking an edge will bring up the context menu, and allows you to delete the edge or edit its metadata:

- Label (optional): A friendly identifier for this edge (currently unused)
- Field Names (optional): If this edge contains multiple fields, you can specify their names as a comma-separated list of values
- Field Units (optional): If this edge contains multiple fields, you can specify their units as a comma-separated list of values

3.9 Logging In

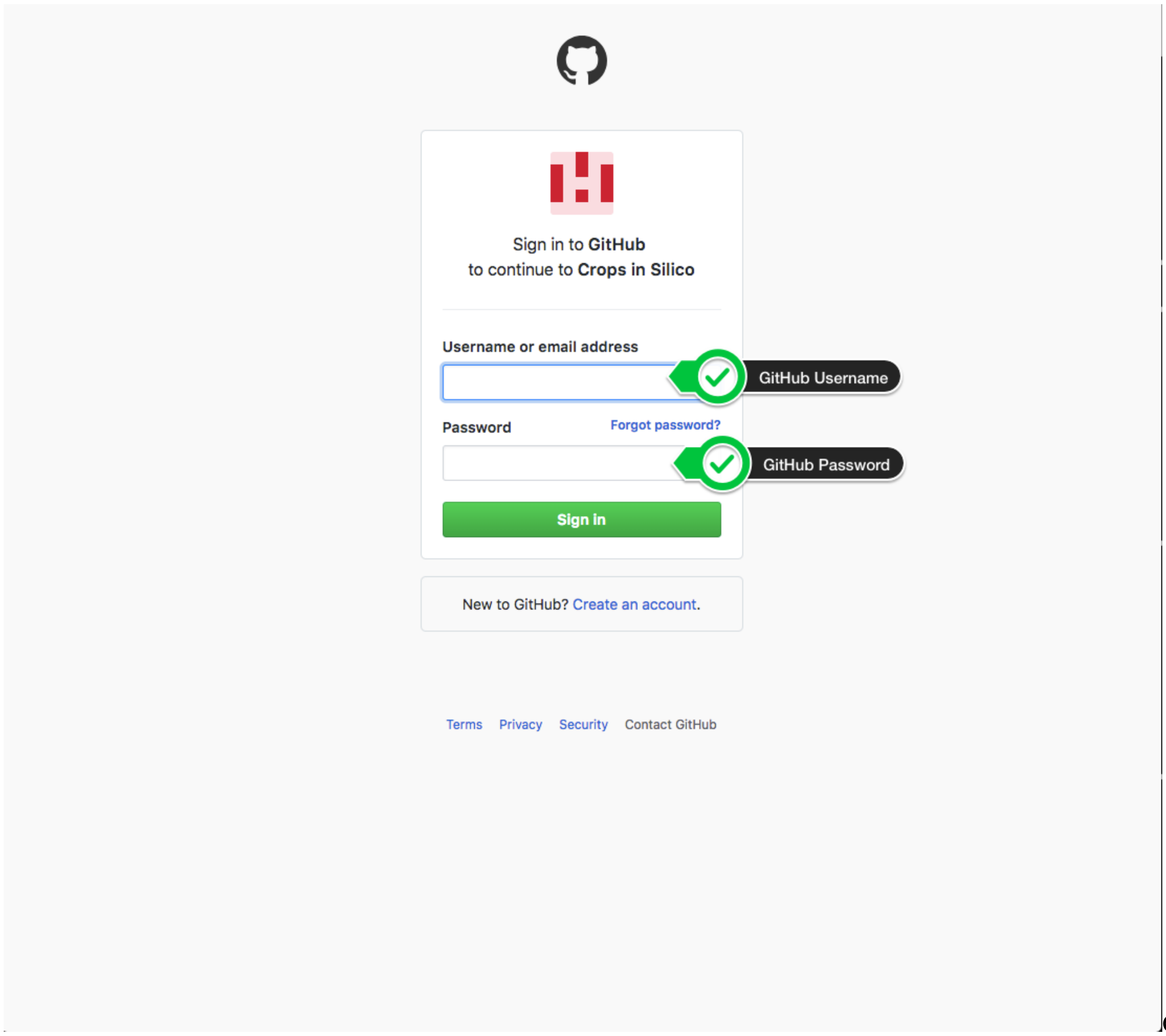
For users who simply wish to build new graphs from our existing sets of models, we do not require them to create an account or log into the system. Anyone who wishes to Create a New Model or Save a Composed Graph to the database, however, will need to sign up for an account. This is simply to track which users created which models, to avoid showing unvetted or defunct options to all users.

To begin the login process, click `Log In` at the top-right in the Navbar:


Crops in Silico
Log In
Model Library
Load
Clear
Generate Manifest
Click "Log In" to sign in with your GitHub account
If you do not already have a GitHub account, you can sign up for free at GitHub.com

OAuth Start

You will then be prompted for your GitHub account credentials. If you do not already have a GitHub account, you can sign up for one for free on GitHub.com

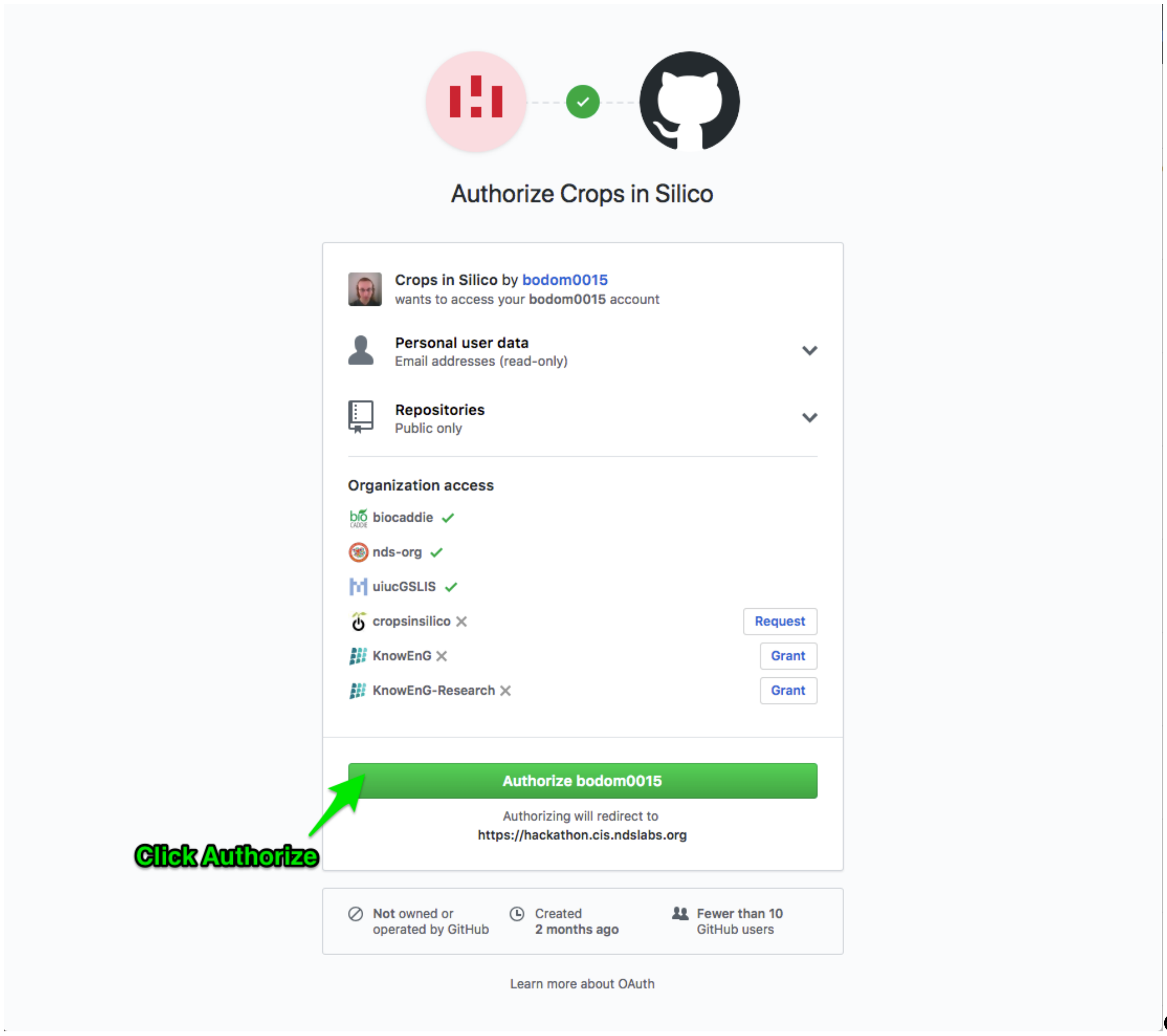

Sign in to GitHub
to continue to **Crops in Silico**

Username or email address

Password

OAuth Authentication

After signing in, you will be asked to Authorize the Crops *in silico* Platform to access your GitHub account information. We only examine your user profile information and use it to create al inked account in our system.



Authorize Crops in Silico

Crops in Silico by bodom0015
wants to access your bodom0015 account

Personal user data
Email addresses (read-only)

Repositories
Public only

Organization access
biocaddie ✓
nds-org ✓
uiucGSLIS ✓
cropsinsilico ✗ Request
KnowEnG ✗ Grant
KnowEnG-Research ✗ Grant

Authorize bodom0015

Authorizing will redirect to
https://hackathon.cis.ndslabs.org

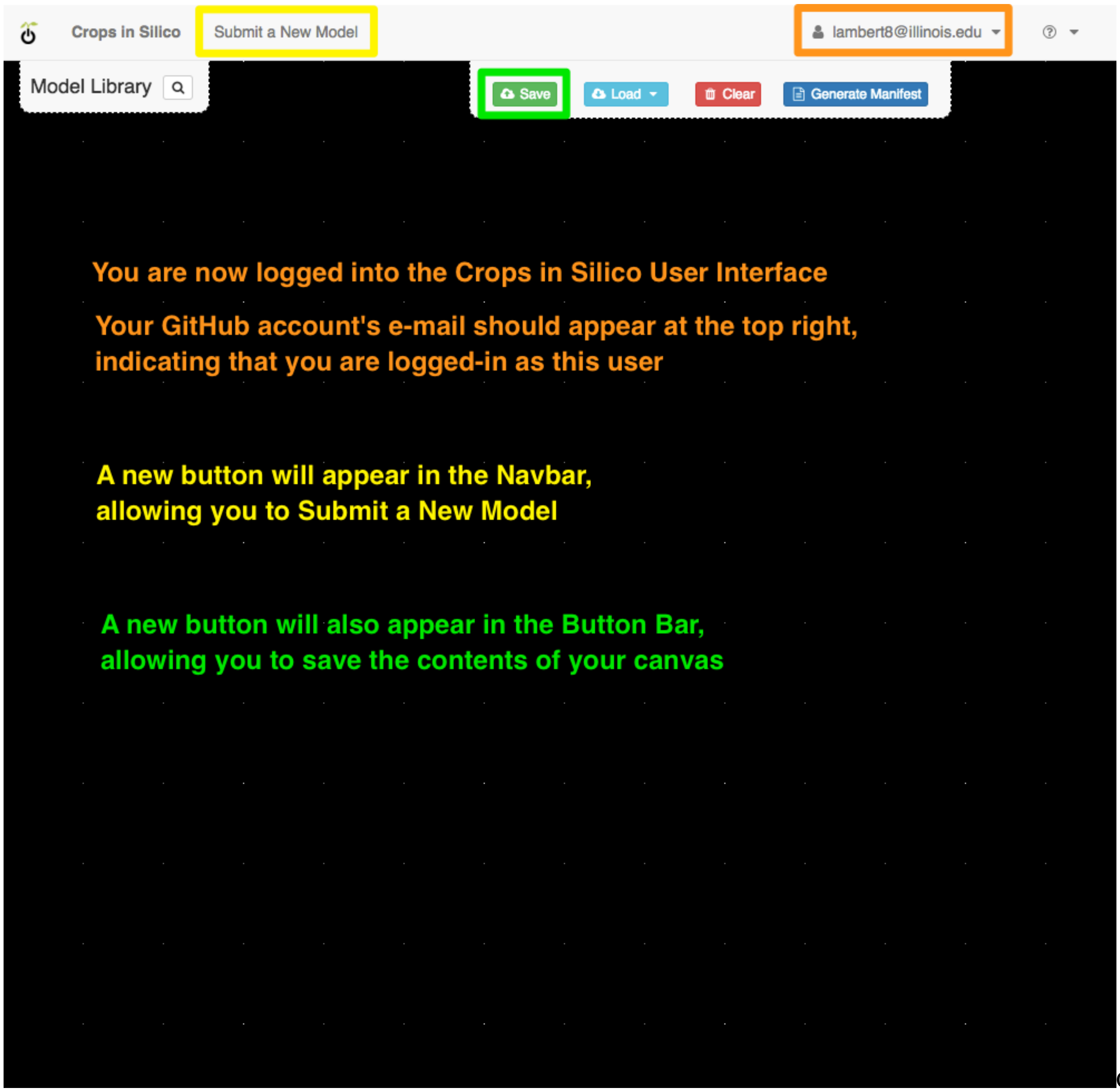
Not owned or operated by GitHub
Created 2 months ago
Fewer than 10 GitHub users

Learn more about OAuth

OAuth Authorization

Click “Authorize” to be redirected the Crops *in silico* application. You are now logged in as your GitHub user, and should see a couple of new buttons have appeared in the User Interface:

- Save Graph
- Submit a New Model



You are now logged into the Crops in Silico User Interface

Your GitHub account's e-mail should appear at the top right, indicating that you are logged-in as this user

A new button will appear in the Navbar, allowing you to Submit a New Model

A new button will also appear in the Button Bar, allowing you to save the contents of your canvas

OAuth Authorized

3.10 Saving a Composed Graph

After logging in, the Button Bar will include a `Save` button. Clicking this button will offer you a prompt to name this graph. If a valid name is entered, the graph will be saved to the database.

3.10.1 Loading a Previously-Saved Graph

After a graph has been saved, it should appear in the `Load` dropdown on the Button Bar. Choosing a previously-saved graph will load its contents into the Canvas.

NOTE: In order to facilitate composing different models and examples, `Load` does not Clear your existing graph state.

3.10.2 Deleting a Previously-Saved Graph

For graphs that you have saved, you should see a `Delete` button beside them. Simply click this button to remove the saved graph from the database.

3.11 Creating a New Model

Do you have a new model that you would like to contribute? After logging in, the Navbar offers a link that will allow you to submit your own custom model metadata. Simply `Log In` at the top-right, then click `Submit a New Model` at the top-left. A pop-up should appear allowing you to enter all necessary metadata fields of you model. Once created, your model appears in your personal catalog for testing and debugging.

3.11.1 Deleting a Model

If you have created a model that you would like to remove, you should see a `Delete` button listed on the Model Palette once when it is expanded. Click the search icon to expand the Model Palette. You should see a red `Delete` button beside any unofficial specs that you have created.

NOTE: Once a model or graph has been officially submitted, it cannot be deleted

3.12 Official Submission

When you are satisfied with the working state of your model or graph, you can submit it to our official catalog as an issue or pull request to the cis-specs repository. Here it will go through a peer review process where it will be testing and vetted. If it passes the review, it will be accepted into our official catalog, where all users will be able to consume and use the new model.

We are working to automate this submission process further, and we thank you for your patience while we determine the best course of action.

CHAPTER 4

Developer's Guide

The purpose of this document is to familiarize you with the patterns used to develop the Crops *in silico* framework.

4.1 Running the Crops in silico Platform

No matter how you choose to run the platform, you will need a few things:

1. MongoDB: An open-source NoSQL database
2. Girder: An open-source data management
3. The `cis-ui` web application

4.2 Running Under Kubernetes (Production-Ready)

The platform has been most thoroughly-tested while running under Kubernetes.

To get started with a single Kubernetes master node, you can follow the instructions here: https://github.com/nds-org/kubeadm-bootstrap#setting-up-a-cluster

You can continue testing with a single node, or add additional workers by following these instructions: https://github.com/nds-org/kubeadm-bootstrap#setting-up-a-worker-node

You should not need more than one Kubernetes node to run the CiS Platform.

Once you have Kubernetes up and running, you can use the templates in `cis-startup` to quickly get up and running:

```
git clone https://github.com/cropsinsilico/cis-startup && cd cis-startup
```

NOTE: You may need to edit the hostnames the Ingress resources located in `platform/` to match your desired hostname.

If everything came up as it should, navigating to http://desired.hostname.com:80 should bring you to the CiS Model Composer UI.

4.3 Running Under Docker (Development Only)

If Kubernetes feels like too much overhead, running under Docker is simple enough.

The following command will start up MongoDB:

```
docker run -it --name=mongodb -v $(pwd):/data/db -p 27017:27017 mongo:3.3
```

Then run a Girder container:

```
docker run -it --name=girder --link mongodb:mongodb -p 8080:8080 cropsinsilico/
↪girder:stable --host 0.0.0.0 --database mongodb://mongodb:27017/girder
```

Finally, run our NGINX container with the `cis-ui` source:

```
docker run -it --name=cis-ui --link girder:girder -p 80:80 cropsinsilico/cis-ui:stable
```

If everything came up as it should, navigating to http://localhost:80 should bring you to the CiS Model Composer UI.

4.4 Running Without Containers (Not recommended)

It is highly recommended that you run each of these services as a separate container, but installing them on the host should work as well.

A few steps are necessary to get the platform up and running on a new host:

1. Install MongoDB
2. Install Girder and configure it to talk to MongoDB
3. Install NodeJS and use it to run `cis-ui`, which will talk to Girder

Such steps are highly manual and will likely vary from version to version of the softwares mentioned above.

For this reason, we do not provide such instructions and instead suggest running everything in a container, as it greatly simplifies the complexity of running, configuring, and connecting these applications.

4.5 Girder Configuration

Coming Soon!

4.6 Adjusting the Server Root

Coming Soon!

4.7 Enabling OAuth

Coming Soon!

4.8 Enabling `cis-girder-plugin`

Coming Soon!

4.9 Crops *in silico* Development Environment

The following instructions should help you get a development environment up and running for modifying the Crops *in silico* Model Composer UI

4.10 Prerequisites

- Git
- Docker or NodeJS

Clone this repository:

```
git clone https://github.com/cropsinsilico/cis-ui && cd cis-ui
```

4.11 Build

Build the webserver Docker image:

```
docker build -t cis/ui .
```

4.12 Run

Start a development webserver on port `8080`:

```
docker run -it -p 8080:80 --name=cis-ui -v $(pwd):/usr/nginx/share/html cis/ui
```

NOTE: `-v src:dest` tells Docker to map `src` from your host to `dest` within the container

4.12.1 Without Docker

If you don’t have Docker, then you will need to install NodeJS and run the following:

```
npm start
```

4.13 Develop

Start a Cloud9 IDE on port `8081`:

```
docker run -it -p 8081:80 --name=cloud9-cis -v $(pwd):/workspace -w /workspace␣
↪ndslabs/cloud9-nodejs
```

4.13.1 Regenerate API client

If you modify the swagger spec, you will need to regenerate the API client.

A `grunt` task has been provided to do this for you:

```
npm install -g grunt
npm install
grunt swagger
```

Ensure that the client and server always have matching API specs.

4.13.2 Rebuilding Jekyll Locally

Execute the following command to run a `jekyll` development server:

```
docker rm -f cis-jekyll
docker run -itd \
    --name=cis-jekyll \
    --label=jekyll \
    --volume=$(pwd):/srv/jekyll \
    -p 127.0.0.1:4000:4000 \
    -p 127.0.0.1:35729:35729 \
    jekyll/jekyll \
    jekyll server --watch --livereload
```

You can then view the logs of your running container by executing:

```
docker logs -f cis-jekyll
```

CHAPTER 5

Administrator's Guide

5.1 Jetstream

The platform is currently hosted on XSEDE Jetstream.

5.2 Kubernetes

The Cis platform is deployed via the Kubernetes container orchestration system. Kubernetes is deployed on Jetstream/OpenStack using either kubeadm-bootstrap or kubeadm-terraform.

Configuration options include:

- NFS dynamic volume provisioner
- Cert manager for TLS certificate management
- NGINX ingress controller
- Flannel network provider

5.3 JupyterHub

The primary entrypoint for the model composer and execution framework is a customized JupyterHub instance. Configuration options include:

- Use of JupyterLab as default environment
- Github authentication
- Pre-populated model source code via `gitpuller`
- MATLAB mounted via `hostPath`

5.4 JupyterLab

JupyterHub and the model execution framework share a common JupyterLab image defined in cropsinsilico/jupyterlab.

This image includes:

- Model composer extension

5.5 MATLAB support

MATLAB support is enabled by installing MATLAB on the host and mounting the installation directory into the JupyterLab and job execution environments.

5.6 Dockerhub

All images are hosted on Dockerhub under the cropsinsilico organization. Most images should have autobuild configured.

5.7 Github

Source code is managed on Github under the cropsinsilico organization.

5.8 Sphinx

```
git clone https://github.com/cis-ui/ -b gh-pages cis-ui-ghpages
```

```
git clone https://github.com/cis-ui/
cd cis-ui/docs
make html
cp -a _build/html/ ../../cis-ui-ghpages/
```

5.9 VM Setup

To setup a new Jetstream project, see the [Jetstream documentation] (https://iujetstream.atlassian.net/wiki/spaces/JWT/pages/44826638/S
You'll need to create a network, subnet, router, and security groups for HTTP/S and SSL.

Jetstream has tons of images intended for use with Atmosphere. It's generally easiest to upload or `glance` in your own images for use in the system. Download Ubuntu 16.04 cloud and either upload or use the `openstack` client:

```
openstack image create   --disk-format qcow2 --container-format bare  \
     --file xenial-server-cloudimg-amd64-disk1.img "Ubuntu 16.04 LTS"
```

Launch a new instance following the Horizon documentation above or via the `openstack` client.

Once the instance is up, install `nfs-common`:

```
sudo apt-get install nfs-common
```

Install Kubernetes via `kubeadm-bootstrap`:

```
git clone https://github.com/data-8/kubeadm-bootstrap
cd kubeadm-bootstrap
sudo ./install-kubeadm.bash
sudo -E ./init-master.bash
```

Install cert-manager (See also https://opensource.ncsa.illinois.edu/confluence/display/~lambert8/Kubernetes+Cert-Manager):

```
mkdir cert-manager
vi cert-manager-values.yaml
sudo helm repo update
sudo helm install --debug  --name cert-manager -f cert-manager-values.yaml stable/
↪cert-manager
kubectl create -f letsencrypt-staging.yaml
```

Upgrade Helm (required by JupyterHub 0.7):

```
curl https://storage.googleapis.com/kubernetes-helm/helm-v2.10.0-linux-amd64.tar.gz |␣
↪tar xvz
sudo mv linux-amd64/helm /usr/local/bin
helm init --upgrade
```

Install JupyterHub:

```
sudo helm repo add jupyterhub https://jupyterhub.github.io/helm-chart/
sudo helm repo update

cd /home/ubuntu/cis-startup/jupyterhub
helm upgrade --install hub jupyterhub/jupyterhub \
  --namespace hub  \
  --version 0.7.0 \
  --values config.yaml
```

To upgrade an existing chart:

```
helm upgrade hub jupyterhub/jupyterhub \
  --version=0.7.0 \
  --values config.yaml
```

To install MATLAB, download R2018a ISO disk images from UIUC webstore and transfer to host

Create `input.txt`:

```
fileInstallationKey=<your key>
licensePath=/home/ubuntu/matlab-cis/license.dat
agreeToLicense=yes
mode=silent
```

Install. When prompted to enter disk 2, unmount first iso and mount second:

```
sudo su -
mkdir /mnt/matlab
mount -t iso9660 -o loop /home/ubuntu/matlab/R2018a_glnxa64_dvd1.iso /mnt/matlab
```

(continues on next page)

(continued from previous page)

```
/mnt/matlab/install -inputFile /home/ubuntu/matlab-cis/input.txt
umount /mnt/matlab
mount -t iso9660 -o loop /home/ubuntu/matlab/R2018a_glnxa64_dvd2.iso /mnt/matlab
umount /mnt/matlab
```

MATLAB is now installed in `/usr/local/MATLAB/R2018a`

To test via standard Jupyter environment:

```
docker run -v /usr/local/MATLAB:/usr/local/MATLAB -it jupyter/scipy-
↪notebook:2bfbb7d17524 bash

export PATH=/usr/local/MATLAB/R2018a/bin/:$PATH
export LM_LICENSE_FILE=<license server>

matlab -nodisplay -nosplash -nodesktop -nojvm
```

Optionally, test `cis_interface`:

```
# Install python engine
cd /usr/local/MATLAB/R2018a/extern/engines/python
python setup.py build -b /tmp install

# Install cis_interface
pip install cis_interface

#Run example
git clone https://github.com/cropsinsilico/cis_interface
cd cis_interface/cis_interface/examples/hello/
cisrun hello_matlab.yml
```

To install CiS, clone cis-startup, cis-ui, cis-girder-plugin. Modify the specs as needed.

```
cd cis-startup/platform
kubectl apply -f rbac/
kubectl apply -f girder.dev.yaml -f girder.staging.yaml -f girder.prod.yaml

cd pvcs/nfs
kubectl create -f deployment.yaml  -f rbac.yaml  -f storageclass.yaml
```

Register the admin user, enable and configure Oauth, Jobs and CiS plugins. Restart Girder.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search