BuildStock Documentation

Release 0.1

Noel Merket

Jun 28, 2019

Contents:

CHAPTER 1

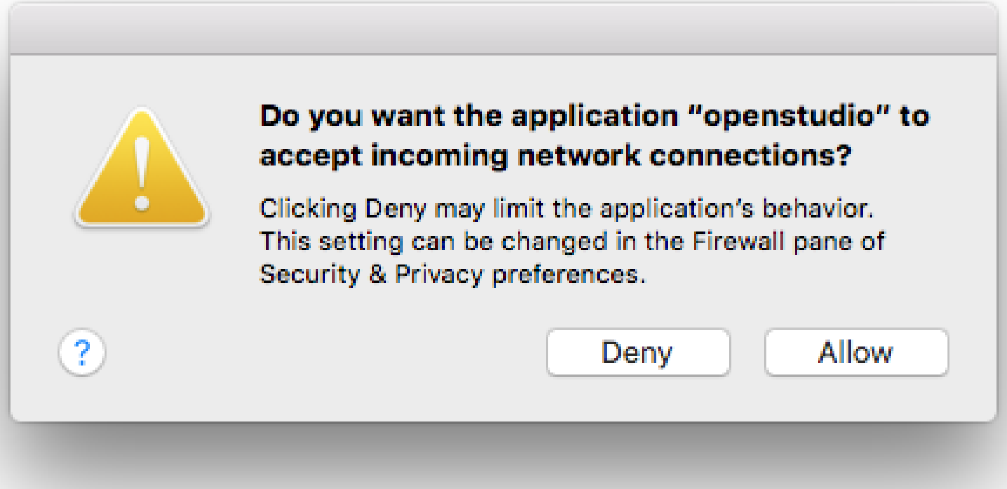
Tutorial

1.1 Installation

1.1.1 Install OpenStudio and PAT

Download the latest stable software version of OpenStudio (2.8.0) from the OpenStudio developer website. This is necessary because the latest critical changes to run ResStock projects are only available in the latest version. Do a default install including Parametric Analysis Tool (PAT).

Open the Parametric Analysis Tool (PAT). You may be asked if you would like to allow openstudio to allow connections. Select “Allow”.



Do you want the application "openstudio" to accept incoming network connections?

Clicking Deny may limit the application's behavior. This setting can be changed in the Firewall pane of Security & Privacy preferences.

1.1.2 Download the ResStock repository

Go to the repository page on GitHub and either `git clone` or download a zip of the repository. Make sure that you have checked out the *master* branch of the repository.

1.2 Set Up the Analysis Project

At the top level of the ResStock repository you just downloaded, you will see two analysis project folders:

- project_resstock_national
- project_resstock_multifamily
- project_resstock_testing

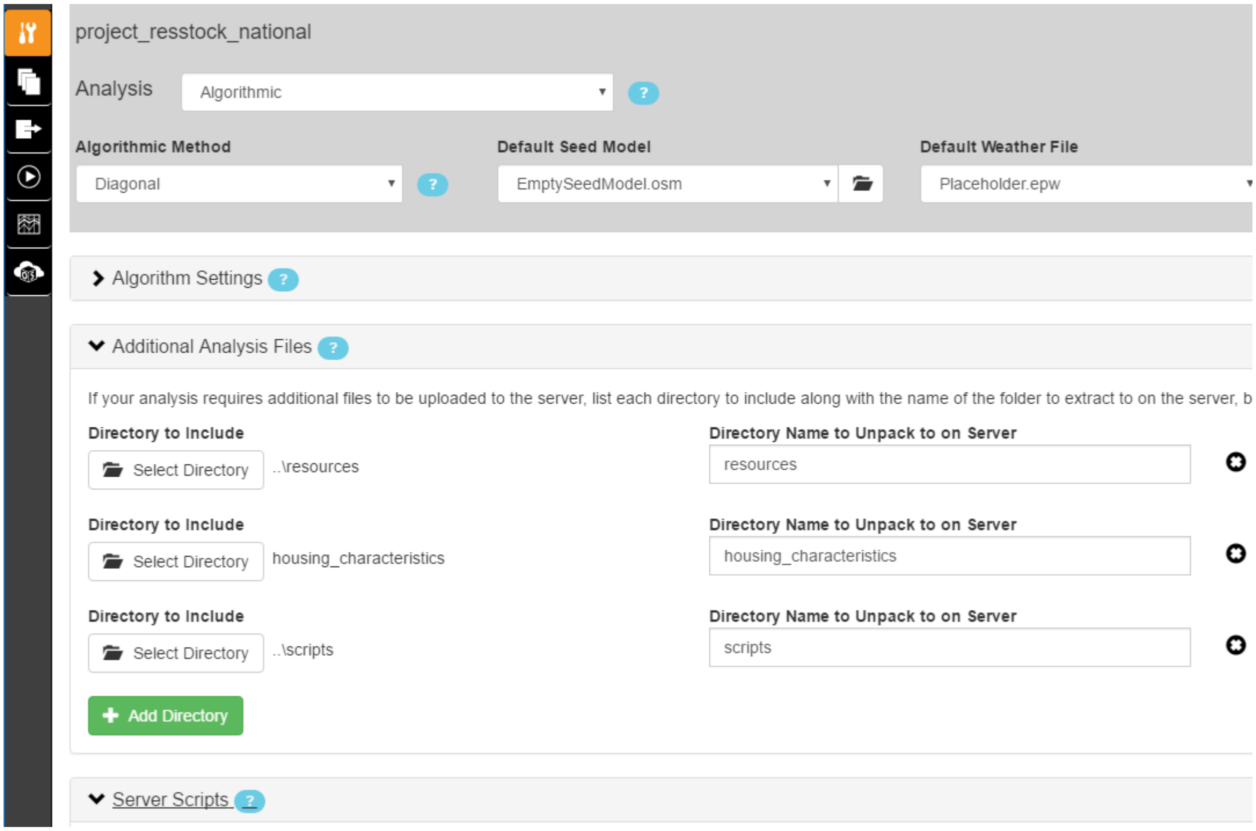
Open PAT, select "Open Existing Project", and choose the `project_resstock_national` directory. You may be asked if you want "mongod" to accept incoming connections. Select "Allow".

You will leave dropdown options for **Algorithmic Method**, **Default Seed Model**, and **Default Weather File** alone. Additionally, you will leave the settings in **Algorithm Settings**, **Additional Analysis Files**, and **Server Scripts** alone for most analyses.

Note: The number of simulations per upgrade scenario is set in *Build Existing Model*.

1.2.1 Additional Analysis Files

Ensure that you have the following directories included to be uploaded to the server:



project_resstock_national

Analysis
Algorithmic

Algorithmic Method
Diagonal

Default Seed Model
EmptySeedModel.osm

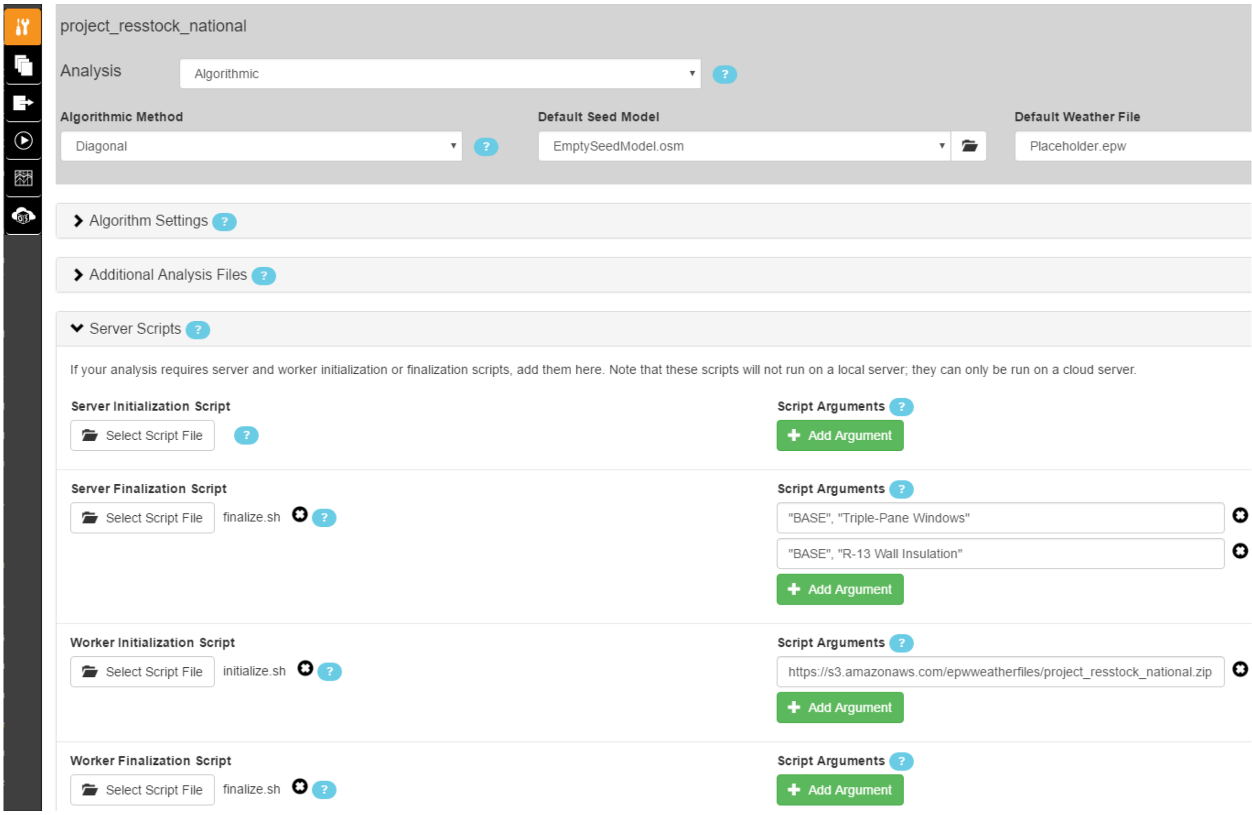
Default Weather File
Placeholder.epw

If your analysis requires additional files to be uploaded to the server, list each directory to include along with the name of the folder to extract to on the server, b

Directory to Include	Directory Name to Unpack to on Server
..\resources	resources
housing_characteristics	housing_characteristics
..\scripts	scripts

1.2.2 Server Scripts

Although you will leave these settings alone for most analyses, you do have the ability to change arguments for initialization and finalization scripts that are run on the remote server. In the case you do NOT want to run savings calculations for upgrades or include additional outputs, see *Server Finalization Script*. In the case you want to change the set of epw weather files used for your project, see *Worker Initialization Script*.



project_resstock_national

Analysis: Algorithmic

Algorithmic Method: Diagonal

Default Seed Model: EmptySeedModel.osm

Default Weather File: Placeholder.epw

If your analysis requires server and worker initialization or finalization scripts, add them here. Note that these scripts will not run on a local server; they can only be run on a cloud server.

Server Finalization Script: finalize.sh

Script Arguments:
"BASE", "Triple-Pane Windows"
"BASE", "R-13 Wall Insulation"

Worker Initialization Script: initialize.sh

Script Arguments:
https://s3.amazonaws.com/epwweatherfiles/project_resstock_national.zip

Worker Finalization Script: finalize.sh

Algorithm Settings
Additional Analysis Files
Server Scripts
Server Initialization Script
Select Script File
Script Arguments
Add Argument
Server Finalization Script
Worker Initialization Script
Worker Finalization Script

Server Initialization Script

Ignore this for now.

Server Finalization Script

After all datapoints have been simulated, this script calls a method for calculating the incremental cost and savings for upgrades. You can specify one or more reference scenarios for the cost and energy subtraction by entering `"reference", "upgrade"` (each enclosed in double quotation marks and separated by a comma) pair(s) (one argument for each pair) in the **Script Arguments** section (see image above).

> `upgrade` indicates the upgrade scenario for calculating savings, and should exactly match the “Upgrade Name” string for one of the upgrade measures (see *Apply Upgrade*).
>
> `reference` indicates the upgrade scenario to be used as the reference. Enter “BASE” to use the “as is” existing housing stock baseline as the reference (typical for envelope upgrades; see example above). Enter an “Upgrade Name” string for one of the upgrade measures to use an upgrade scenario as the reference savings and costs to be subtracted from the `upgrade` scenario to calculate incremental savings and costs (typical for equipment upgrades where there exists a minimum efficiency standard).

An example of this latter situation is when an old SEER 8 AC is replaced at wear out, and a user wishes to calculate the incremental savings and cost of upgrading it to SEER 18 compared to a SEER 14 AC (U.S. federal minimum efficiency in southern states).

Entering no pairs will default to calculating savings for all upgrades relative to the baseline building. Note that if you specify one scenario in this way, then you must explicitly define all scenario pairs, even cases where the reference is the baseline (using “BASE”, “upgrade”).

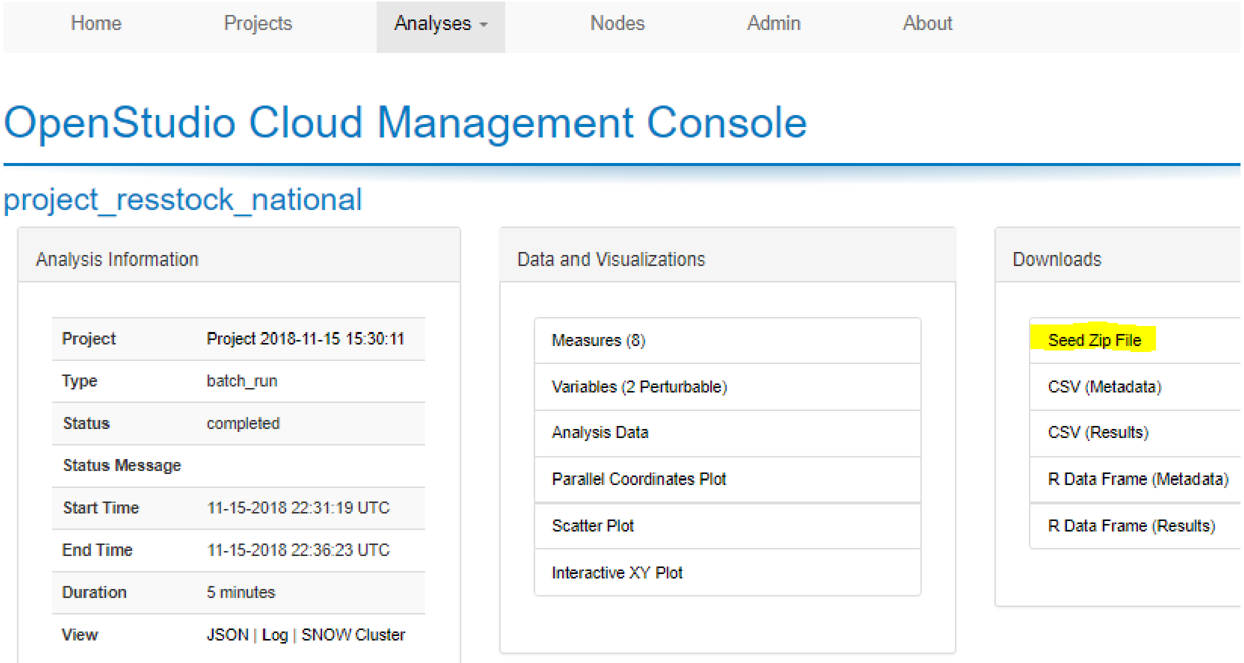
Savings are calculated as follows:

etc.

By default this script also attaches additional outputs to the results.csv file, including:

- reportable domain (according to RECS 2009)
- source energy (using conversion factors from BSR/ASHRAE Standard 105-2013)
- eGRID subregion (see the entire eGRID subregion map)
- utility bill calculations (for now using simple average flat rates for each state)
- simple payback
- net present value
- savings-to-investment ratio

A new csv file, `results_savings.csv`, containing upgrade savings and additional outputs is produced. You can retrieve this file by downloading the **Seed Zip File** from the OpenStudio Cloud Management Console analysis page:



Project	Project 2018-11-15 15:30:11
Type	batch_run
Status	completed
Status Message	
Start Time	11-15-2018 22:31:19 UTC
End Time	11-15-2018 22:36:23 UTC
Duration	5 minutes
View	JSON | Log | SNOW Cluster

Worker Initialization Script

Something you might want to change is the set of weather files used with your project. To update the argument for the path to the zip file containing epw weather files, open the Server Scripts box on the Measures Selection tab.

Look for the **Script Arguments** box corresponding to the **Worker Initialization Script**. By default, this argument value points to the set of weather files corresponding to the specific project (i.e., set of `housing_characteristics`) you are working with. For example, the `project_resstock_national` project folder will by default use the set of weather files with national geographic coverage. In the illustration above, the argument value path points to a zipped file stored in the epwweatherfiles bucket on Amazon S3. You should have read-only access to objects in this bucket.

You can control what set of weather files are unpacked and accessible on the remote server by changing the argument value for this initialization script. If you wish to change this argument value to point to a different file in the S3 bucket, replace the path's basename with the path of the new file. If the desired file does not exist in the S3 bucket, you will need to zip up a set of weather files and upload it to some location of your choice (e.g., your own S3 bucket). Be sure to change the entire argument value path to point to this chosen file location.

To zip and upload new weather files:

- First ensure that the weather files you will be using do not already exist in the S3 bucket. If they do, just point to the appropriate zip that already contains your desired weather files.
- If they do not, on your local computer highlight all the new epw weather files and compress them into a single zip file. (Your zip should contain only files with either the ".epw" or ".ddy" extension.)
- Upload your newly zipped file that contains the weather files to your new location.
- Go back to your project and update the argument value to the path of the newly uploaded file.

Note: Changing this path from the default will most likely require additional changes to your project. Any weather file names in your `housing_characteristics` folder's tsv files will need to be updated to reflect those in the S3 bucket file. Any simulation on the remote server that points to an invalid weather file path will fail.

Worker Finalization Script

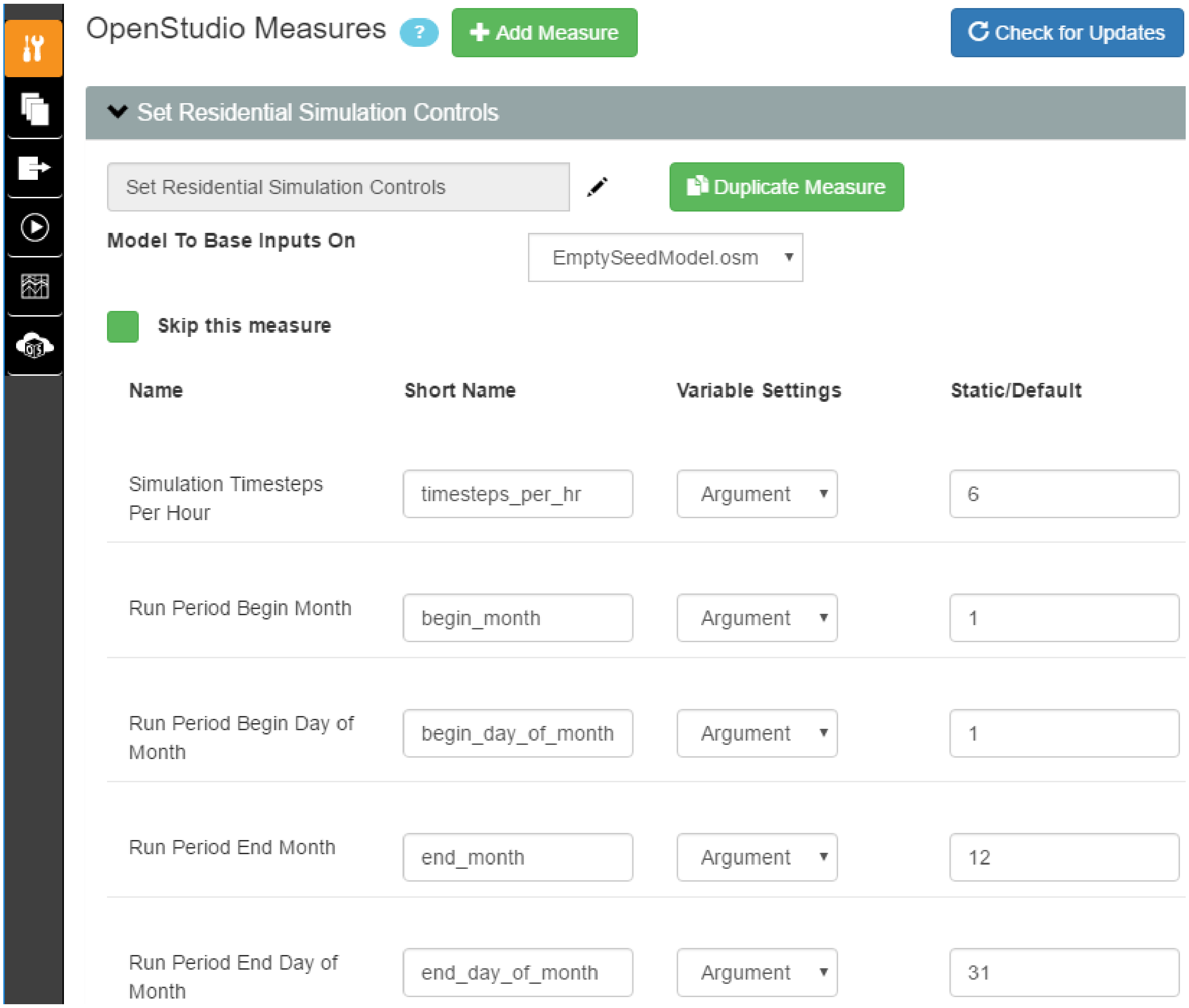
Ignore this for now.

1.2.3 OpenStudio Measures

Continuing on the Measures Selection tab, scroll down to the **OpenStudio Measures** section. This section is where you will define the parameters of the analysis including the baseline case and any upgrade scenarios.

Simulation Controls

Using this measure you can set the simulation timesteps per hour, as well as the run period begin month/day and end month/day. By default the simulations use a 10-min timestep (i.e., the number of timesteps per hour is 6), start on January 1, and end on December 31.



Set Residential Simulation Controls

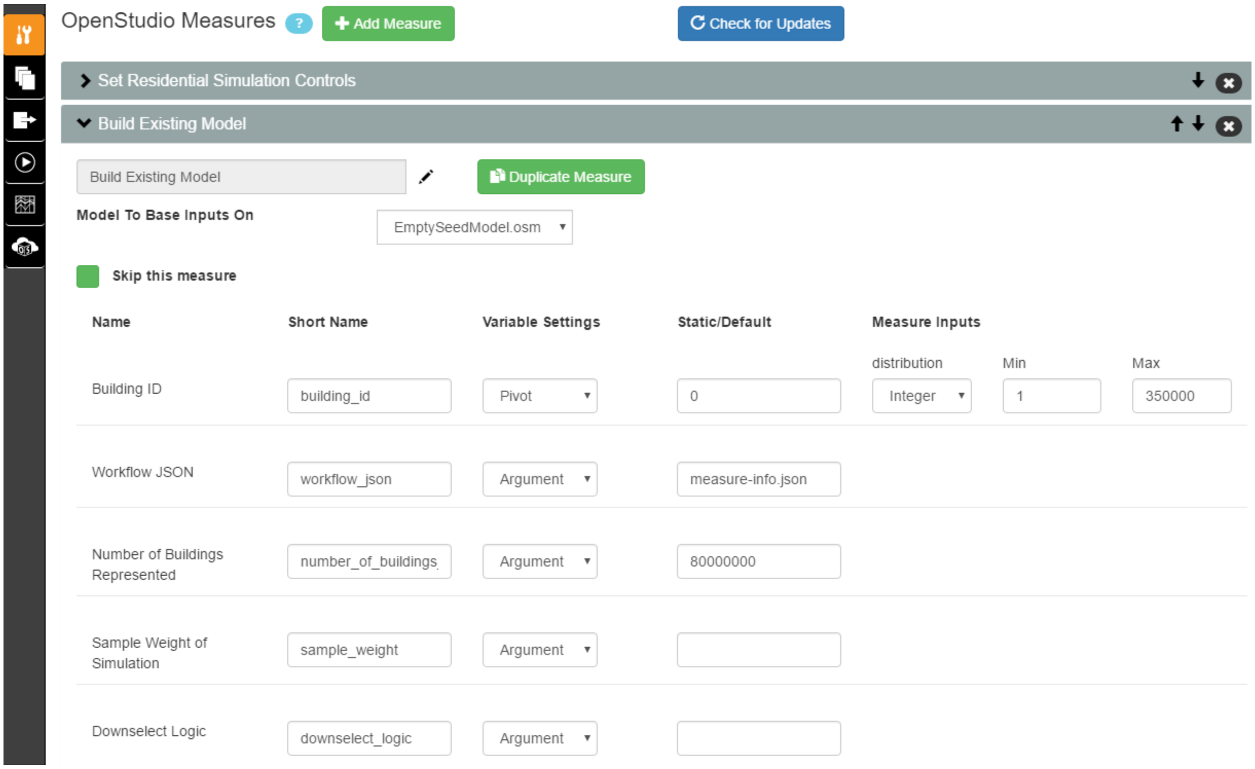
Model To Base Inputs On: EmptySeedModel.osm

🟩 Skip this measure

Name	Short Name	Variable Settings	Static/Default
Simulation Timesteps Per Hour	timesteps_per_hr	Argument	6
Run Period Begin Month	begin_month	Argument	1
Run Period Begin Day of Month	begin_day_of_month	Argument	1
Run Period End Month	end_month	Argument	12
Run Period End Day of Month	end_day_of_month	Argument	31

Build Existing Model

This measure creates the baseline scenario. It incrementally applies OpenStudio measures (located in the `resources` directory, which should be at the same level as your project directory) to create residential building models. Set the following inputs:



Build Existing Model

Model To Base Inputs On: EmptySeedModel.osm

🟩 Skip this measure

Name	Short Name	Variable Settings	Static/Default	Measure Inputs		
				distribution	Min	Max
Building ID	building_id	Pivot	0	Integer	1	350000
Workflow JSON	workflow_json	Argument	measure-info.json			
Number of Buildings Represented	number_of_buildings	Argument	80000000			
Sample Weight of Simulation	sample_weight	Argument				
Downselect Logic	downselect_logic	Argument				

Building ID – Max This sets the number of simulations to run in the baseline and each upgrade case. For this tutorial I am going to set this to 1000. Most analyses will require more, but we're going to keep the total number small for simulation time and cost.

Number of Buildings Represented The total number of buildings this sampling is meant to represent. This sets the weighting factors. For the U.S. single-family detached housing stock, this is 80 million homes.

Sample Weight of Simulation The number of buildings each simulation represents. Total number of buildings / Number of simulations. This argument is optional (it is only needed for running simulations on NREL HPC), so you can leave it blank.

Downselect Logic Logic that specifies the subset of the building stock to be considered in the analysis. Specify one or more `parameter|option` as found in the `resources/options_lookup.tsv`. (This uses the same syntax as the *Apply Upgrade* measure.) For example, if you wanted to only simulate California homes you could enter `Location Region|CR11` in this field (CR refers to "Custom Region", which is based on RECS 2009 reportable domains aggregated into groups with similar climates; see the entire custom region map).

Note: **Manual Sampling**: To run the sampling script yourself, from the command line execute, e.g. `ruby resources/run_sampling.rb -p project_resstock_national -n 10000 -o buildstock.csv`, and a file `buildstock.csv` will be created in the `resources` directory.

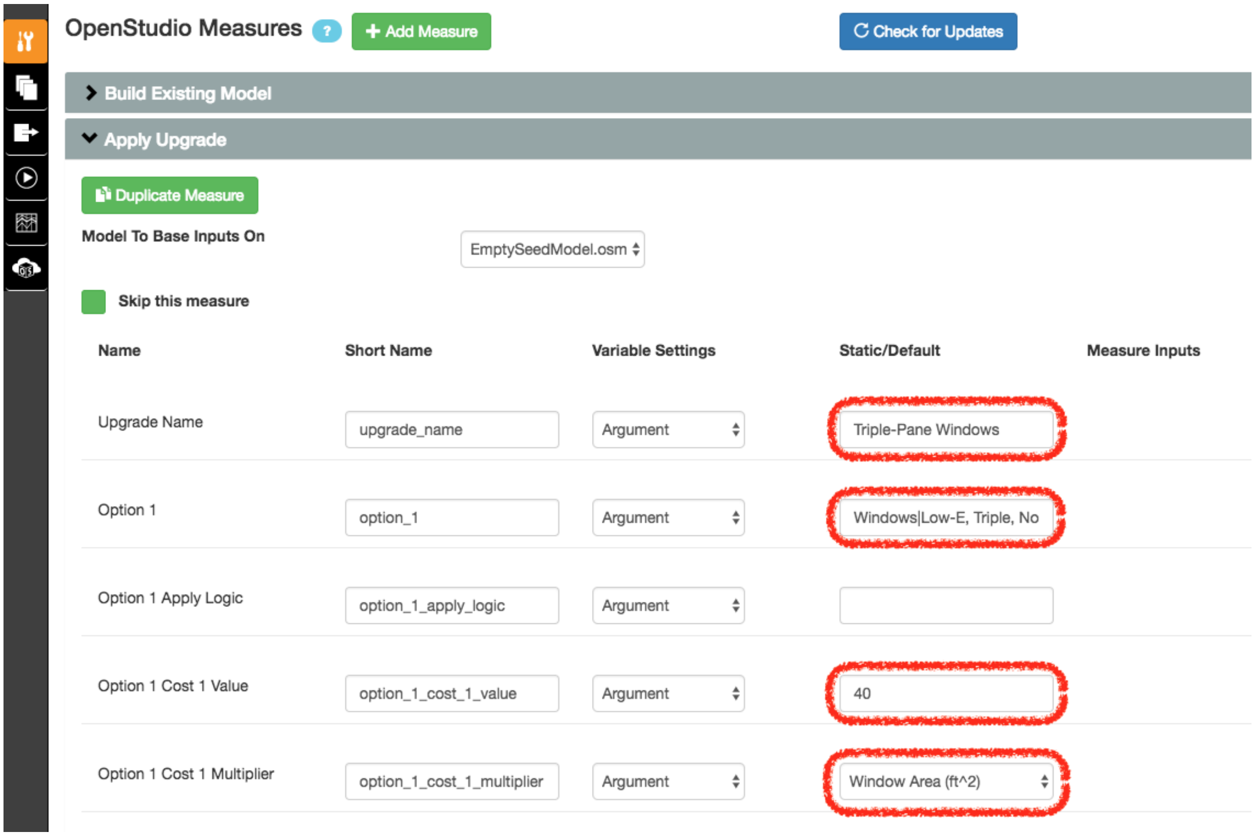
If a custom `buildstock.csv` file is located in a project's `housing_characteristics` directory when you run the PAT project, it will automatically be used to generate simulations. If it's not found, the `run_sampling.rb` script will be run automatically on OpenStudio-Server to create one. You'll also want to make sure that the number of buildings in the sampling csv file matches the max value for the Building ID argument in the Build Existing Model, as that tells OpenStudio how many datapoints to run. (For each datapoint, the measure will then look up its building description from the sampling csv.)

You can use this manual sampling process to downselect which simulations you want to run. For example, you can use the command above to generate a `buildstock.csv` for the entire U.S. and then open up this file in Excel and delete all of the rows that you don't want to simulate (e.g., all rows that aren't in New York). Keep in mind that if you do this, you will need to re-enumerate the “Building” column as “1” through the number of rows.

Apply Upgrade

Each “Apply Upgrade” measure defines an upgrade scenario. An upgrade scenario is a collection of options exercised with some logic and costs applied. In the simplest case, we apply the new option to all houses. The available upgrade options are in `resources/options_lookup.tsv` in your git repository.

For this example, we will upgrade all windows by applying the `Windows|Low-E, Triple, Non-metal, Air, L-Gain` option to all houses across the country. We do this by entering that in the **Option 1** box on the Apply Upgrade measure. Also, we’ll give the upgrade scenario a name: “Triple-Pane Windows” and a cost of \$40/ft^2 of window area by entering the number in **Option 1 Cost Value** and selecting “Window Area (ft^2)” for **Option 1 Cost Multiplier**.



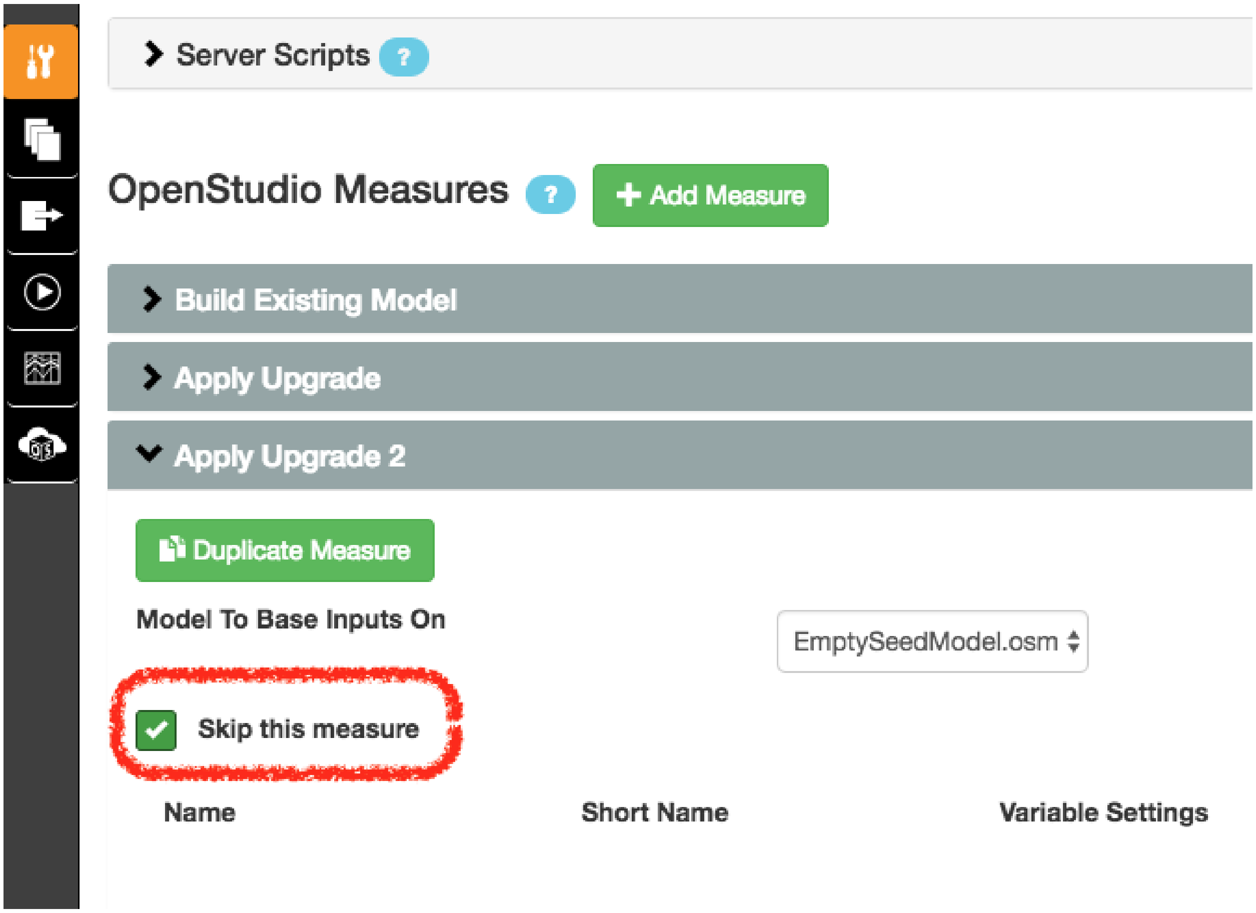
Model To Base Inputs On: EmptySeedModel.osm

☐ Skip this measure

Name	Short Name	Variable Settings	Static/Default	Measure Inputs
Upgrade Name	upgrade_name	Argument	Triple-Pane Windows	
Option 1	option_1	Argument	Windows|Low-E, Triple, No	
Option 1 Apply Logic	option_1_apply_logic	Argument		
Option 1 Cost 1 Value	option_1_cost_1_value	Argument	40	
Option 1 Cost 1 Multiplier	option_1_cost_1_multiplier	Argument	Window Area (ft^2)	

For a full explanation of how to set up the options and logic surrounding them, see *Upgrade Scenario Configuration*.

Measures can be skipped in an analysis without losing their configuration. For this tutorial we will skip the second measure of applying wall insulation. To do so, select the **Apply Upgrade 2** measure, open it, and check the box **Skip this measure**.



1.2.4 Reporting Measures

Scroll down to the bottom on the Measures Selection tab, and you will see the **Reporting Measures** section. This section is where you can request timeseries data and utility bills for the analysis. In general, reporting measures process data after the simulation has finished and produced results. As a note, make sure that the **Timeseries CSV Export** and **Utility Bill Calculations** measures are placed before the **Server Directory Cleanup** measure.

Building Charactertistics Report

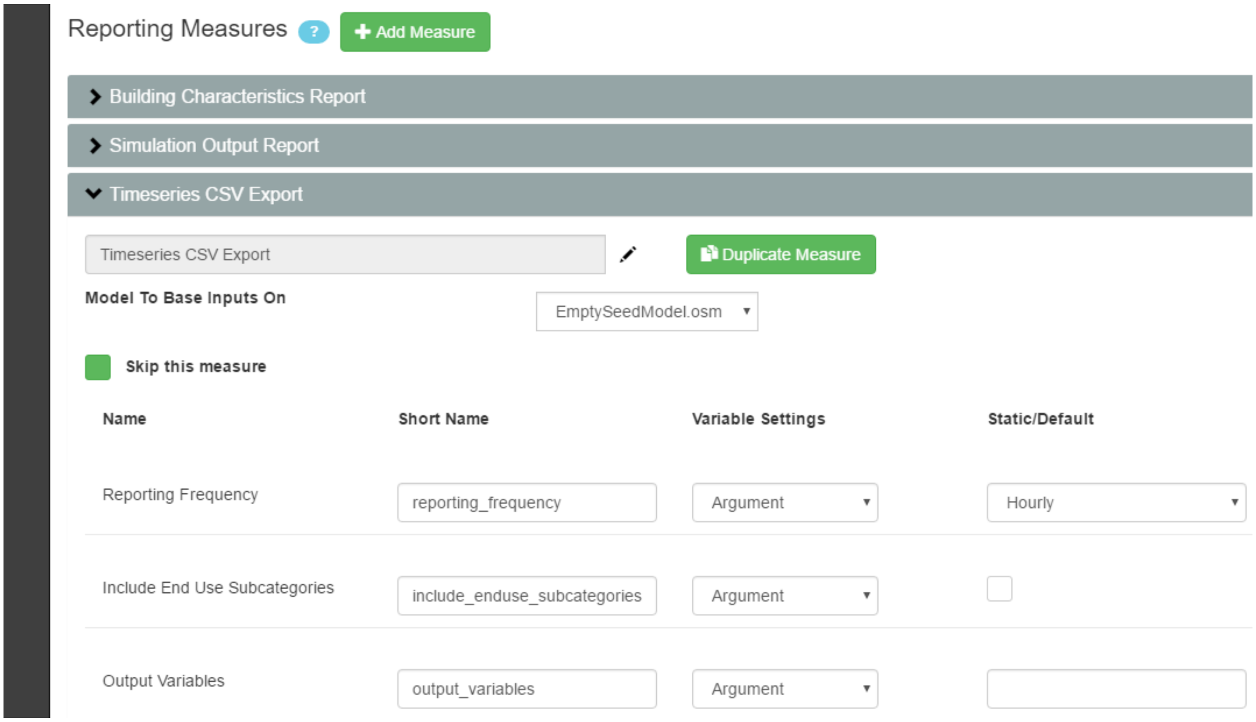
Leave this alone.

Simulation Output Report

Leave this alone.

Timeseries CSV Export

If you do not need the timeseries data for your simulations, you can skip this measure to save disk space. Otherwise, one csv file per datapoint will be written containing end use timeseries data for their model. After downloading all datapoints to your project's localResults folder, each datapoint's `enduse_timeseries.csv` file will be contained in a zipped `data_point.zip` file along with all other simulation input and output files.



Timeseries CSV Export

Model To Base Inputs On: EmptySeedModel.osm

🟩 Skip this measure

Name	Short Name	Variable Settings	Static/Default
Reporting Frequency	reporting_frequency	Argument	Hourly
Include End Use Subcategories	include_enduse_subcategories	Argument	□
Output Variables	output_variables	Argument	

End uses include:

- heating [electric/gas/propane/oil] [kWh/kBtu/kBtu/kBtu]
- cooling [kWh]
- interior lights [kWh]
- exterior lights [kWh]
- interior equipment [electric/gas/propane/oil] [kWh/kBtu/kBtu/kBtu]
- fans [kWh]
- pumps [kWh]
- water heating [electric/gas/propane/oil] [kWh/kBtu/kBtu/kBtu]
- water [gal]
- pv [kWh]

Reporting Frequency The timeseries data will be reported at hourly intervals unless otherwise specified. Alternative reporting frequencies include:

- timestep
- daily
- monthly
- run period

Setting the reporting frequency to “timestep” will give you interval output equal to the zone timestep set by the “Simulation Controls” measure. Thus, this measure will produce 10-min interval output when you select “timestep” and leave the “Simulation Controls” measure at its default settings. Setting the reporting frequency to “detailed” will give you interval output equal to the calculation step (i.e., either zone timestep or HVAC system timestep).

Include End Use Subcategories Select this to include end use subcategories. The default is to not include end use subcategories. End use subcategories include:

- refrigerator [kWh]
- dishwasher [kWh]
- cooking range [electric/gas/propane] [kWh/kBtu/kBtu]
- clothes washer [kWh]
- clothes dryer [electric/gas/propane] [kWh/kBtu/kBtu]
- ceiling fan [kWh]
- mech vent house fan [kWh]
- mech vent range fan [kWh]
- mech vent bath fan [kWh]
- heating supply fan [kWh]
- cooling supply fan [kWh]
- misc plug loads [kWh]
- extra refrigerator [kWh]
- freezer [kWh]
- pool heater [electric/gas] [kWh/kBtu]
- pool pump [kWh]
- hot tub heater [electric/gas] [kWh/kBtu]
- hot tub pump [kWh]
- well pump [kWh]
- gas fireplace [kBtu]
- gas grill [kBtu]
- gas lighting [kBtu]

Output Variables If you choose to report any output variables (e.g., “Zone Air Temperature” or “Site Outdoor Air Humidity Ratio”), enter a comma-separated list of output variable names. A list of available output variables can be viewed in EnergyPlus’s `.rdd` file. One csv file, appropriately called “output_variables.csv”, will be downloaded alongside the “enduse_timeseries.csv” file.

Utility Bill Calculations

This measure is currently under construction. Do not include it in your PAT analysis.

1.3 Run the Project on Amazon Web Services

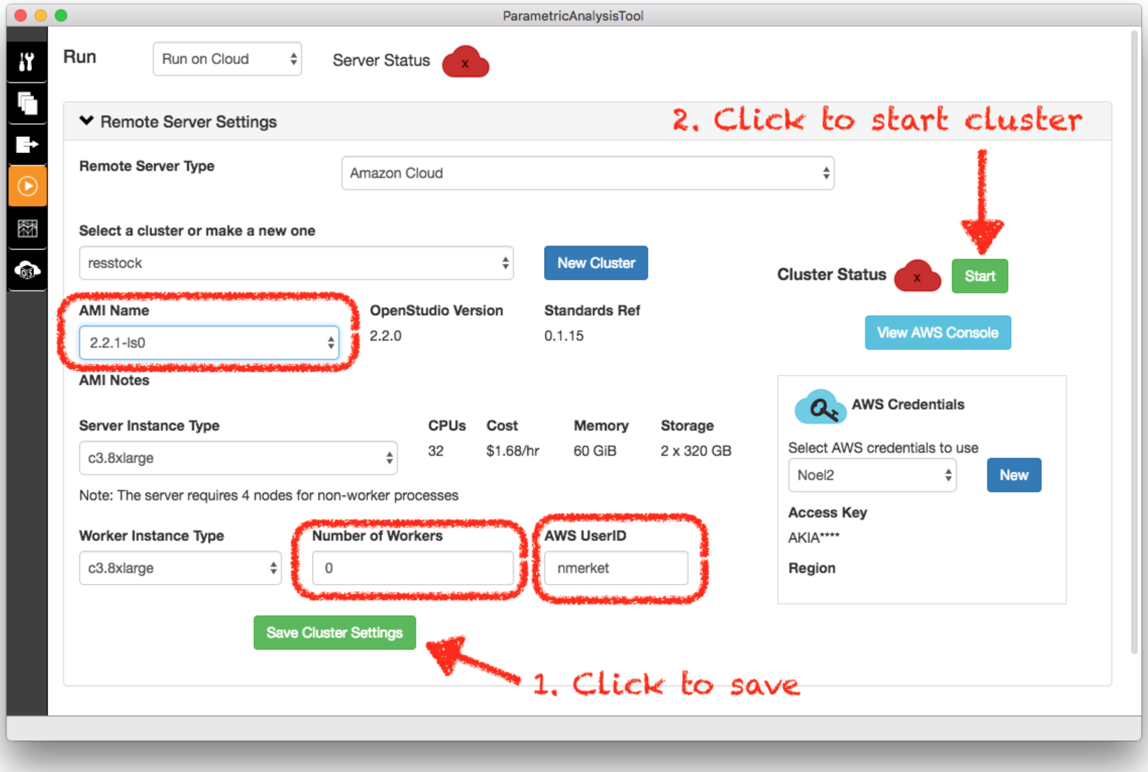
Switch to the Run tab. The selection will initially be on “Run Locally” with an error message stating that you cannot run algorithmic analyses locally. Select “Run on Cloud” and change **Remote Server Type** to “Amazon Cloud” to set up your run environment.

1.3.1 AWS Credentials

First, you will need some AWS credentials to allow PAT to start compute instances in the cloud. Go to https://aws.amazon.com and click the button to create an AWS Account and add a payment method for billing. You will also need to create access keys for your AWS account. When you have your AWS Access and Secret keys, click on the **New** button in the **AWS Credentials** box in PAT and enter your keys. Also, make sure to enter *your* **AWS UserID** on the main run screen.

1.3.2 Cluster Settings and Starting the Cluster

Ensure that your project's AMI selection matches "2.8.0" (this should also be the version of OpenStudio/PAT that you are using). We will leave most of the rest of the settings at their defaults, but because we're doing a small analysis here, we're going to set the number of worker nodes to zero. For guidance on cluster settings for your analysis including instance selection and worker nodes see *AWS Cluster Configuration*.



Run: Run on Cloud

Remote Server Type: Amazon Cloud

Select a cluster or make a new one: resstock

AMI Name: 2.2.1-ls0

OpenStudio Version: 2.2.0

Standards Ref: 0.1.15

AMI Notes

Server Instance Type	CPUs	Cost	Memory	Storage
c3.8xlarge	32	$1.68/hr	60 GiB	2 x 320 GB

Note: The server requires 4 nodes for non-worker processes

Worker Instance Type: c3.8xlarge

Number of Workers: 0

AWS UserID: nmerket

Select AWS credentials to use: Noel2

Access Key: AKIA****

Region

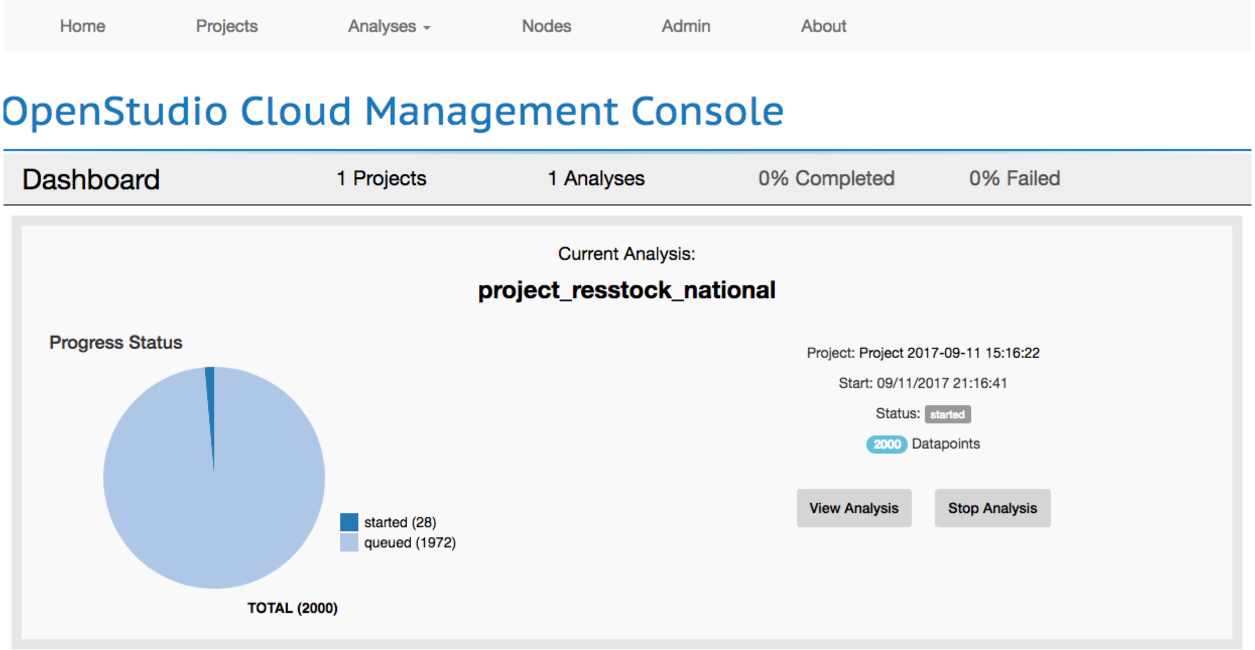
2. Click to start cluster

1. Click to save

Click **Save Cluster Settings** and the **Start** button next to the **Cluster Status** label. Wait for the cluster to start. The cloud icon will turn green when it is ready. It can take up to 10 minutes.

1.3.3 Run Analysis and Monitor Status

When the cluster is running, start the analysis by clicking the **Run Entire Workflow** button below the server settings. You will see a status bar and messages. Once it says "Analysis Started" you can click the **View Server** button to see the status of your analysis on the OpenStudio Server.



Home Projects Analyses Nodes Admin About

OpenStudio Cloud Management Console

Dashboard 1 Projects 1 Analyses 0% Completed 0% Failed

Current Analysis:

project_resstock_national

Progress Status

Project: Project 2017-09-11 15:16:22

Start: 09/11/2017 21:16:41

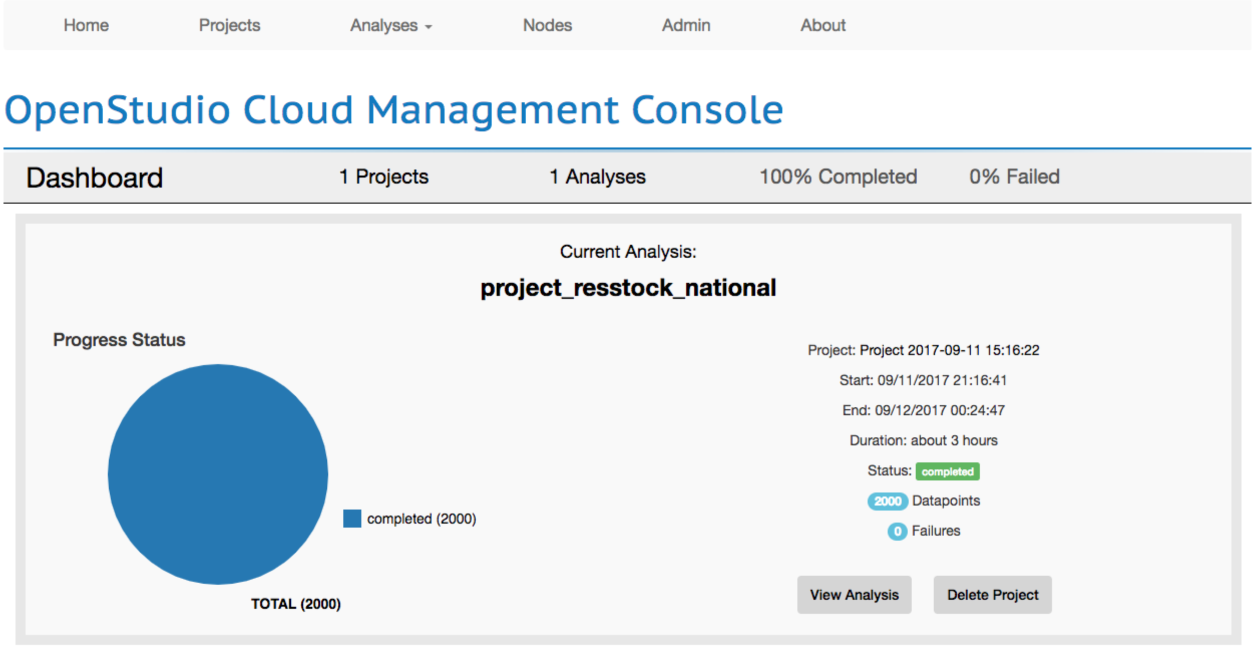
Status: started

2000 Datapoints

Leave the PAT application open while your analysis runs. It could take a while.

1.3.4 Download results

Eventually PAT will show in the status bar "Analysis completed". And the OpenStudio Server console will show the same.



Home Projects Analyses Nodes Admin About

OpenStudio Cloud Management Console

Dashboard 1 Projects 1 Analyses 100% Completed 0% Failed

Current Analysis:

project_resstock_national

Progress Status

Project: Project 2017-09-11 15:16:22

Start: 09/11/2017 21:16:41

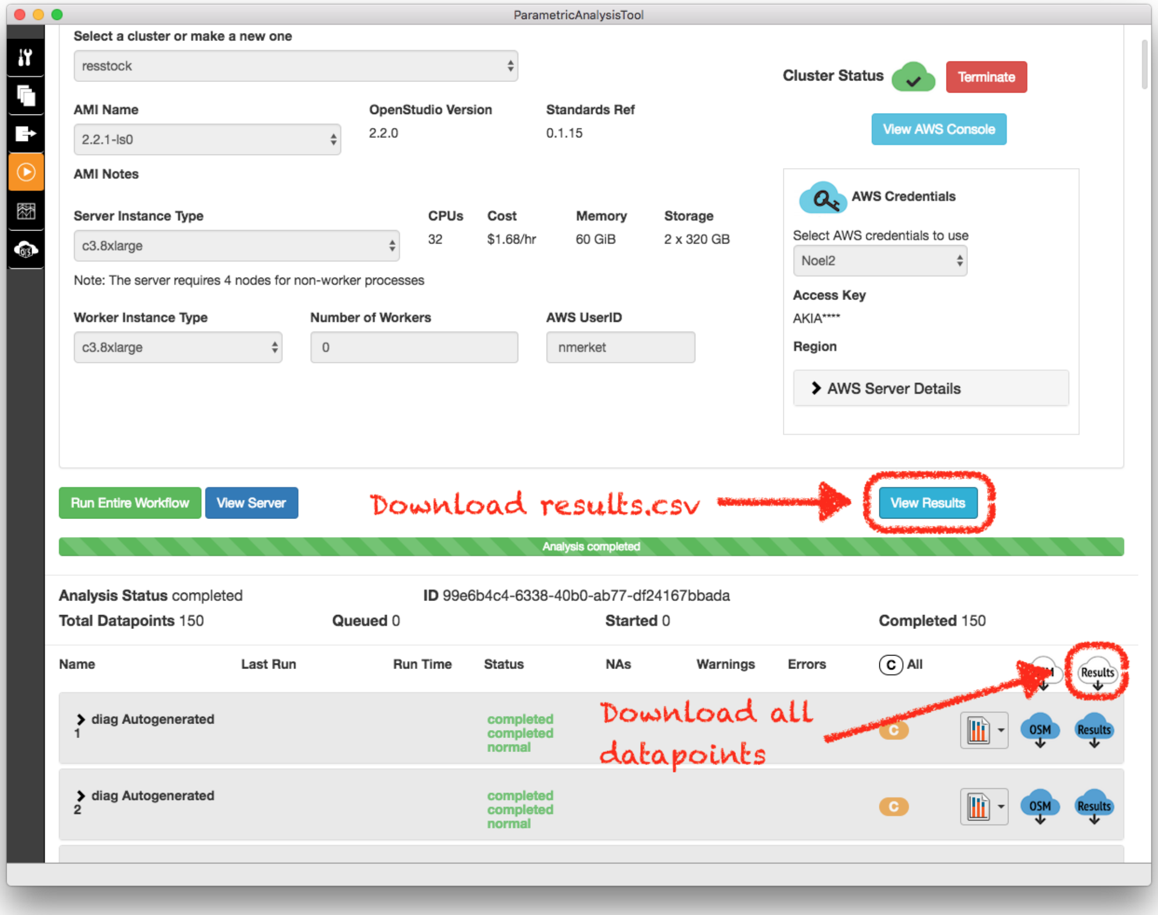
End: 09/12/2017 00:24:47

Duration: about 3 hours

Status: completed

2000 Datapoints

0 Failures



Select a cluster or make a new one: resstock

AMI Name	OpenStudio Version	Standards Ref
2.2.1-ls0	2.2.0	0.1.15

AMI Notes

Server Instance Type	CPUs	Cost	Memory	Storage
c3.8xlarge	32	$1.68/hr	60 GiB	2 x 320 GB

Note: The server requires 4 nodes for non-worker processes

Worker Instance Type	Number of Workers	AWS UserID
c3.8xlarge	0	nmerket

Select AWS credentials to use: Noel2

Access Key: AKIA****

Region

Analysis Status completed ID 99e6b4c4-6338-40b0-ab77-df24167bbada

Total Datapoints 150 Queued 0 Started 0 Completed 150

Name	Last Run	Run Time	Status	NAs	Warnings	Errors
diag Autogenerated 1			completed completed normal			
diag Autogenerated 2			completed completed normal			

Clicking the **View Results** button in PAT will open the results.csv file for your analysis. It contains a row for every sampled building including all options selected for that building and annual energy simulation results. Often this is the only results you will need. That file is saved in your project in `localResults/results.csv`.

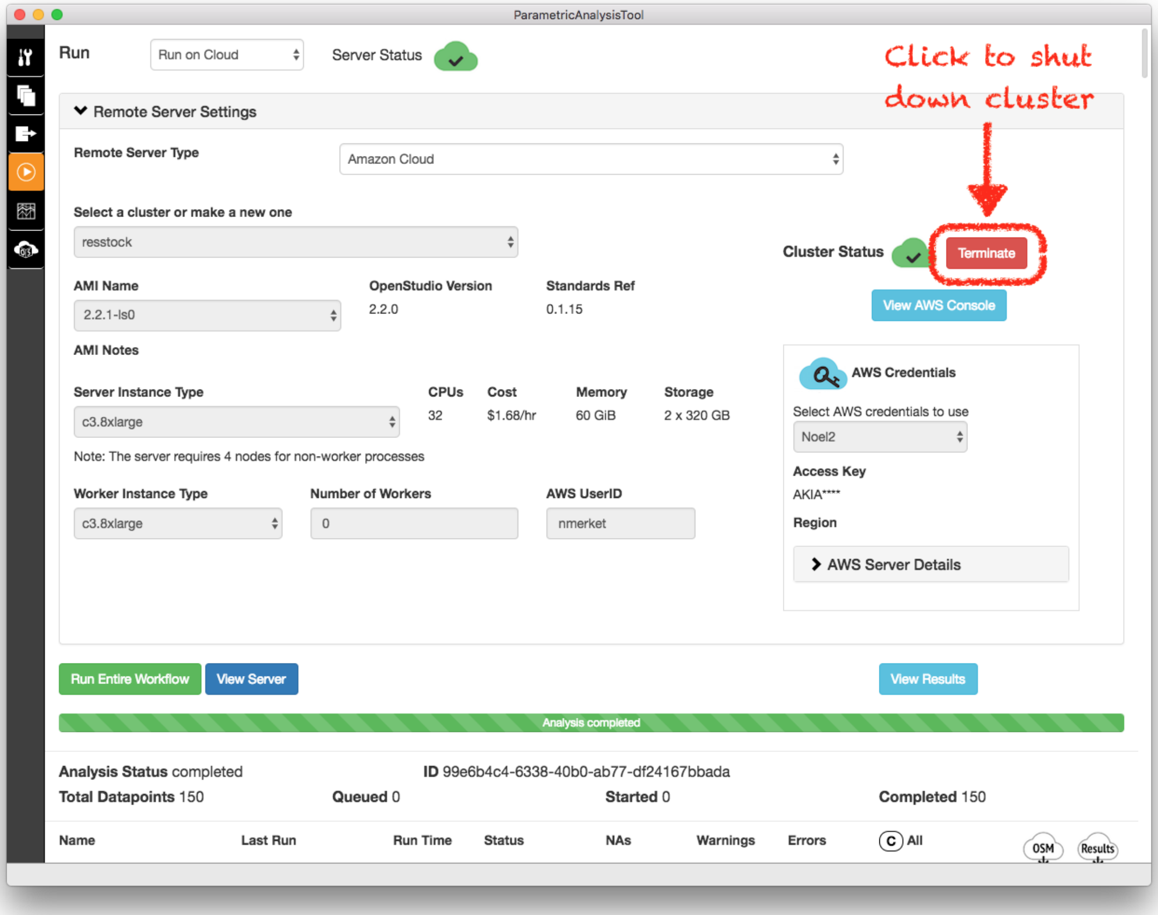
Sometimes you will need *all* the simulation results including timeseries results if you requested them. Clicking the **Results (cloud, down arrow)** button will pull down all of the simulation results from the server and save them to your project. Each result datapoint will be stored in a `localResults/[GUID]` folder in your project.

Warning: Downloading all simulation results can be a lot of data. Make sure you're on a good connection and have enough room on your local machine. Be prepared for the download to take a while.

Note: To download all datapoints, including the timeseries csv output for each simulation, run the `scripts/download_datapoints.rb` script using the OpenStudio CLI. The script requires that the *Timeseries CSV Export* reporting measure be included in your PAT project. Supply all required arguments to the script, including project directory, server DNS, and analysis ID. A usage example is given as follows: `$ /c/openstudio-2.8.0/bin/openstudio.exe scripts/download_datapoints.rb -p project_resstock_national -s http://ec2-107-23-165-146.compute-1.amazonaws.com -a 706c3b4a-9685-4924-bb13-c6bec77aa397` Additionally, the script has an optional argument to unzip each datapoint zip file on the fly.

1.3.5 Shutting Down the OpenStudio Server Cluster

Once you have retrieved all the data you need from your analysis, it is a good idea to shut down the OpenStudio Server Cluster to stop incurring AWS costs. The most straightforward way to do that is to click the **Terminate** button on the Run tab of PAT.



Click to shut down cluster

Run: Run on Cloud

Server Status ✓

Remote Server Settings

Remote Server Type: Amazon Cloud

Select a cluster or make a new one: resstock

Cluster Status ✓

AMI Name	OpenStudio Version	Standards Ref
2.2.1-ls0	2.2.0	0.1.15

AMI Notes

Server Instance Type	CPUs	Cost	Memory	Storage
c3.8xlarge	32	$1.68/hr	60 GiB	2 x 320 GB

Note: The server requires 4 nodes for non-worker processes

Worker Instance Type	Number of Workers	AWS UserID
c3.8xlarge	0	nmerket

AWS Credentials

Select AWS credentials to use: Noel2

Access Key: AKIA****

Region

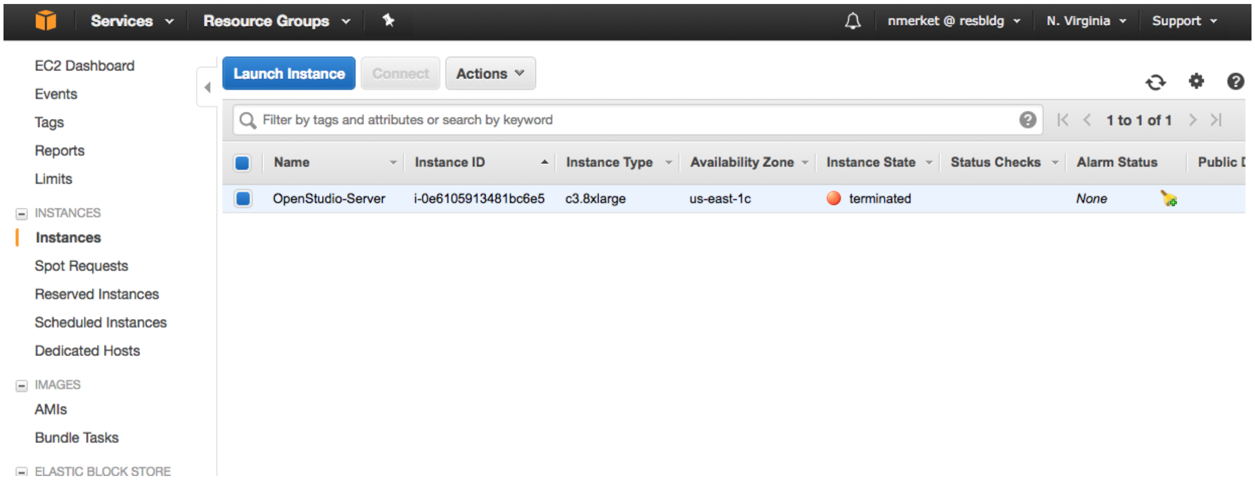
Analysis Status completed

ID 99e6b4c4-6338-40b0-ab77-df24167bbada

Total Datapoints 150 **Queued** 0 **Started** 0 **Completed** 150

Name	Last Run	Run Time	Status	NAs	Warnings	Errors	All

The PAT interface will indicate that the cluster has been shut down after a moment. Just to be sure, it's best to open the AWS cloud console either by clicking the **View AWS Console** button in PAT or visiting https://console.aws.amazon.com. Select "Services" > "EC2". Select "N. Virginia" in the region menu (upper right). You should see a terminated instance of OpenStudio-Server and potentially some workers if you chose to use workers above.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public D
OpenStudio-Server	i-0e6105913481bc6e5	c3.8xlarge	us-east-1c	● terminated		*None*	

If it has been long enough the list will be empty. If for some reason the instances are still running, you can terminate them by right-clicking and selecting “Terminate”. If PAT has been closed or crashed, this is how you will have to shut down the cluster.

1.4 Conclusion

Congratulations, you have now completed your first ResStock analysis. See the other sections in this documentation for more advanced topics.

CHAPTER 2

Advanced Tutorial

This advanced tutorial describes the process for developing residential measures and testing residential building models. Reasons for wanting to develop residential measures include: customizing any of the existing residential modeling algorithms or adding new technology models.

At this point in the tutorial, it is assumed that you have checked out a new branch that is up-to-date with the **master branch** of the OpenStudio-BuildStock repository. Optionally, you may have created a new PAT project folder (i.e., copied an existing project folder) and modified the set of tsv files in its `housing_characteristics` folder.

If your changes are intended to be merged into the master branch of the OpenStudio-BuildStock repository, a pull request review is required.

2.1 Modifying Probability Distributions

This section provides a description of the housing characteristics and their dependencies and options.

A particular building within the building stock has a set of characteristics (e.g., level of wall insulation, type of lighting, vintage, and a variety of different schedules). Each housing characteristic corresponds to a tab-separated value (tsv) file with the extension *.tsv*. These housing characteristics files are found in the `<project_folder>/housing_characteristics` directory. A housing characteristic defines the probability mass function (PMF) of that characteristic in the building stock.

$Pr(X = A_i) = P(A_i) > 0 \quad and \quad \sum_{A_i \in S_A} P(A_i) = 1 \quad for \quad i = 1:n$

When sampling a discrete random variable X to create a representative building, X takes a particular **Option** A_i. All possible options are collected in the set $S_A = \{A_0, A_1, ..., A_n\}$ and is size n. Since these are probabilities, the entries $P(A_i)$ must be greater than 0 and the probability of all possible options must sum to 1.

For example, a set of options for a building's vintage (when the building was built) may be the following:

$S_A =< 1950, 1950s, 1960s, 1970s, 1980s, 1990s, 2000s.$

Then the probability mass function may look like the following:

A_i	<1950	1950s	1960s	1970s	1980s	1990s	2000s
$P(X = A_i)$	0.020	0.060	0.090	0.230	0.370	0.130	0.090

Where the probability of a building having a given vintage in this example is

- 2% built before 1950,
- 6% in the 1950s,
- 9% in the 1960s,
- 23% in the 1970s,
- 37% in the 1980s,
- 13% in the 1900s, and
- 9% in the 2000s.

However, housing characteristics can have a **Dependency**, B_i, to another housing characteristic. All possible values of the dependency are collected in the set $S_B = B_0, B_1, ...B_m$ which is size m. If the **Option** of interest A_j and the **Dependency** B_i is known to have occurred when sampling X in the creation of a representative building, then conditional probability of A_j given B_i is usually written $P(A_j|B_i) = P_{B_i}(A_j)$.

Using the example from before, the PMF of the vintage depends on location of the particular building stock (which is represented by EPW weather files). In this example the vintage housing characteristic is examined. The first three lines in the `<project_folder>/housing_characteristics/Vintage.tsv` are shown in the table below.

	Location EPW (S_B)	<1950	1950s	1960s	1970s	1980s	1990s	2000s
$P(B_0\|A_j)$	USA_FL_Key.West.Intl.AP.722010_TMY3.epw	0.02	0.06	0.09	0.23	0.37	0.13	0.09
$P(B_1\|A_j)$	USA_FL_Miami.Intl.AP.722020_TMY3.epw	0.05	0.13	0.13	0.18	0.17	0.18	0.16

The vintage is dependent on the EPW location. The vintage discrete PMF that uses the Key West International Airport weather file, B_0, is defined by the following distribution:

- 2% built before 1950,
- 6% in the 1950s,
- 9% in the 1960s,
- 23% in the 1970s,
- 37% in the 1980s,
- 13% in the 1900s, and
- 9% in the 2000s.

While the vintage PMF that uses the Miami International Airport weather file, B_1 is defined by the following distribution:

- 5% built before 1950,
- 13% in the 1950s,
- 9% in the 1960s,
- 13% in the 1970s,
- 18% in the 1980s,
- 17% in the 1900s, and

- 18% in the 2000s.

The **Options** can correspond to a Measure in OpenStudio or can be used as a **Dependency** for other housing characteristics. For the list of available options for a given housing characteristic, see the `resources/options_lookup.tsv` file. In this file the "Parameter Name" corresponds to the housing characteristic, the "Option Name" corresponds to an available option for the housing characteristic, the "Measure Dir" corresponds to the OpenStudio Measure being used, and the following columns correspond to different arguments needed by the OpenStudio Measure. Each option used in the housing characteristics tsv files must be in this `resources/options_lookup.tsv`. These options can be modified by the user to model their particular building stock.

If adding or renaming any housing characteristics tsv files, refer to the *Refresh Outputs* section for instructions on how to get the sampled options to show up in results files.

2.2 Installer Setup

After you have downloaded the OpenStudio installer, you will want to install Ruby. This will allow you to execute rake tasks contained in the Rakefile. Follow the instructions below for *Windows Setup* or *Mac Setup*.

2.2.1 Windows Setup

1. Install Ruby. Follow the installation instructions here ("Optional - Install Ruby").
2. Run `gem install bundler`. (If you get an error, you may have to issue the following: `gem sources -r https://rubygems.org/` followed by `gem sources -a http://rubygems.org/`.)
3. Download the DevKit at http://rubyinstaller.org/downloads/. Choose either the 32-bit or 64-bit version depending on which version of Ruby you installed. Run the installer and extract to a directory (e.g., C:RubyDevKit). Go to this directory, run `ruby dk.rb init`, modify the config.yml file as needed, and finally run `ruby dk.rb install`.
4. Run `bundler` from the OpenStudio-BuildStock directory. (If you get an error, the problem may be that `git` is not in your `PATH`.)

2.2.2 Mac Setup

Install Homebrew if you don't have it already.

Run `brew doctor`. It should give you, among other issues, a list of unexpected dylibs that you'll need to move for this to work such as:

```
Unexpected dylibs:
  /usr/local/lib/libcrypto.0.9.8.dylib
  /usr/local/lib/libcrypto.1.0.0.dylib
  /usr/local/lib/libcrypto.dylib
  /usr/local/lib/libklcsagt.dylib
  /usr/local/lib/libklcskca.dylib
  /usr/local/lib/libklcsnagt.dylib
  /usr/local/lib/libklcsrt.dylib
  /usr/local/lib/libklcsstd.dylib
  /usr/local/lib/libklcstr.dylib
  /usr/local/lib/libklmspack.0.1.0.dylib
  /usr/local/lib/libklmspack.0.dylib
  /usr/local/lib/libklmspack.dylib
  /usr/local/lib/libssl.0.9.8.dylib
```

(continues on next page)

(continued from previous page)

```
  /usr/local/lib/libssl.1.0.0.dylib
  /usr/local/lib/libssl.dylib
  /usr/local/lib/libz.1.2.5.dylib
  /usr/local/lib/libz.1.2.6.dylib
  /usr/local/lib/libz.1.dylib
  /usr/local/lib/libz.dylib
```

Highlight and copy the list (without the header "Unexpected dylibs:"). Run the following commands to move them to another location where they won't interfere.

```
mkdir ~/unused_dylibs
pbpaste | xargs -t -I % mv % ~/unused_dylibs
```

Install `rbenv` and required dependencies.

```
brew install openssl libyaml libffi rbenv
```

Initialize `rbenv` by running the command below and following the instructions to add the appropriate things to your `~/.bash_profile`.

```
rbenv init
```

Install the appropriate ruby version.

```
cd path/to/repo
rbenv install `cat .ruby-version`
```

Add the path to the install ruby libraries top the bottom of your `~/.bash_profile`

```
echo "export RUBYLIB=/Applications/OpenStudio-2.0.5/Ruby" >> ~/.bash_profile
echo "export ENERGYPLUS_EXE_PATH=\"/Applications/OpenStudio-2.1.0/EnergyPlus/
↪energyplus-8.7.0\""
```

Install bundler and the libraries that bundler installs.

```
gem install bundler
bundle install
```

2.3 Rake Tasks

Once you have completed instructions found in *Installer Setup*, you can then *use the Rakefile* contained at the top level of this repository (Rakefile). First you will run a rake task for copying measures and resource files from the OpenStudio-BEopt) repository into the top-level `resources/measures` folder. Then you will run rake task(s) for *performing integrity checks on project inputs*.

2.3.1 Using the Rakefile

Run `rake -T` to see the list of possible rake tasks. The `-T` is replaced with the chosen task.

```
$ rake -T
rake integrity_check_all                    # Perform integrity check on inputs...
rake integrity_check_resstock_national      # Perform integrity check on inputs...
```

(continues on next page)

(continued from previous page)

```
rake integrity_check_resstock_multifamily # Perform integrity check on inputs...
rake integrity_check_resstock_testing     # Perform integrity check on inputs...
rake test:all                             # Run tests for all
rake test:regenerate_osms                 # Run tests for regenerate_osms
rake update_measures                      # update all measures
rake update_tariffs                       # update urdb tariffs
```

2.3.2 Integrity Checks

Run `rake integrity_check_resstock_<project_name>`, where `<project_name>` matches the project you are working with. If no rake task exists for the project you are working with, extend the list of integrity check rake tasks to accommodate your project by copy-pasting and renaming the `integrity_check_resstock_national` rake task found in the Rakefile. An example for running a project's integrity checks is given below:

```
$ rake integrity_check_resstock_national
Checking for issues with project_resstock_national/Location Region...
Checking for issues with project_resstock_national/Location EPW...
Checking for issues with project_resstock_national/Vintage...
Checking for issues with project_resstock_national/Heating Fuel...
Checking for issues with project_resstock_national/Usage Level...
...
```

If the integrity check for a given project fails, you will need to update either your tsv files and/or the `resources/options_lookup.tsv` file. See *Options Lookup* for information about the `options_lookup.tsv` file.

2.4 Options Lookup

The `options_lookup.tsv` file, found in the `resources` folder, specifies mappings from sampled options into measure arguments. For example, if the distribution of cooling system types in `HVAC System Cooling.tsv` has `Option=AC, SEER 13` and `Option=AC, SEER 15`, but you want to include a `Option=AC, SEER 17` option, you would add that option as a column in `HVAC System Cooling.tsv` and then create a corresponding row in `options_lookup.tsv`. Updates to this file will allow you to avoid hitting the following types of integrity check errors:

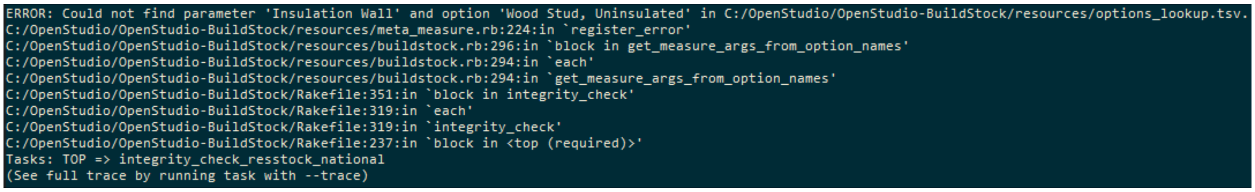
- *Could not find parameter and option*
- *Required argument not provided*

Once you have updated your `options_lookup.tsv` file and all integrity checks are passing, you can move on to *Updating Projects*.

2.4.1 Could not find parameter and option

You do not have a row in `options_lookup.tsv` for a particular option that is sampled.

An example of this error is given below:

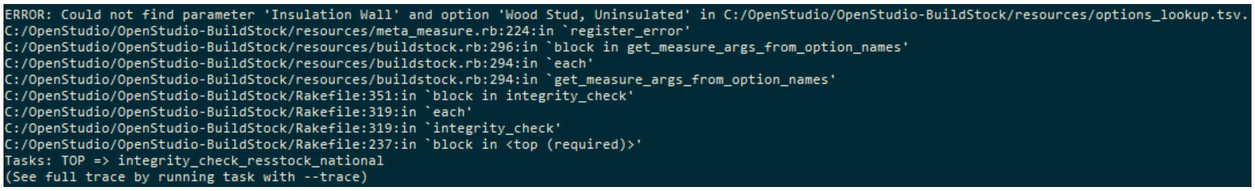


```
ERROR: Could not find parameter 'Insulation Wall' and option 'Wood Stud, Uninsulated' in C:/OpenStudio/OpenStudio-BuildStock/resources/options_lookup.tsv.
C:/OpenStudio/OpenStudio-BuildStock/resources/meta_measure.rb:224:in `register_error'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:296:in `block in get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `each'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:351:in `block in integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `each'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:237:in `block in <top (required)>'
Tasks: TOP => integrity_check_resstock_national
(See full trace by running task with --trace)
```

2.4.2 Required argument not provided

For the particular option that is sampled, your corresponding measure is missing an argument value assignment.

An example of this error is given below:



```
ERROR: Could not find parameter 'Insulation Wall' and option 'Wood Stud, Uninsulated' in C:/OpenStudio/OpenStudio-BuildStock/resources/options_lookup.tsv.
C:/OpenStudio/OpenStudio-BuildStock/resources/meta_measure.rb:224:in `register_error'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:296:in `block in get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `each'
C:/OpenStudio/OpenStudio-BuildStock/resources/buildstock.rb:294:in `get_measure_args_from_option_names'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:351:in `block in integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `each'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:319:in `integrity_check'
C:/OpenStudio/OpenStudio-BuildStock/Rakefile:237:in `block in <top (required)>'
Tasks: TOP => integrity_check_resstock_national
(See full trace by running task with --trace)
```

2.5 Updating Projects

Once you have successfully run the rake tests described in *Rake Tasks*, you are ready to update your PAT project(s) and run ResStock analyses.

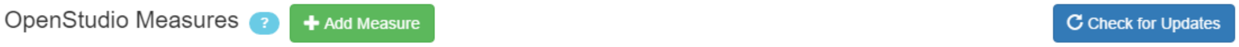
Discrepancies may exist between top-level measures and project-level measures, you will need to:

- *Check for Updates*
- *Refresh Outputs*

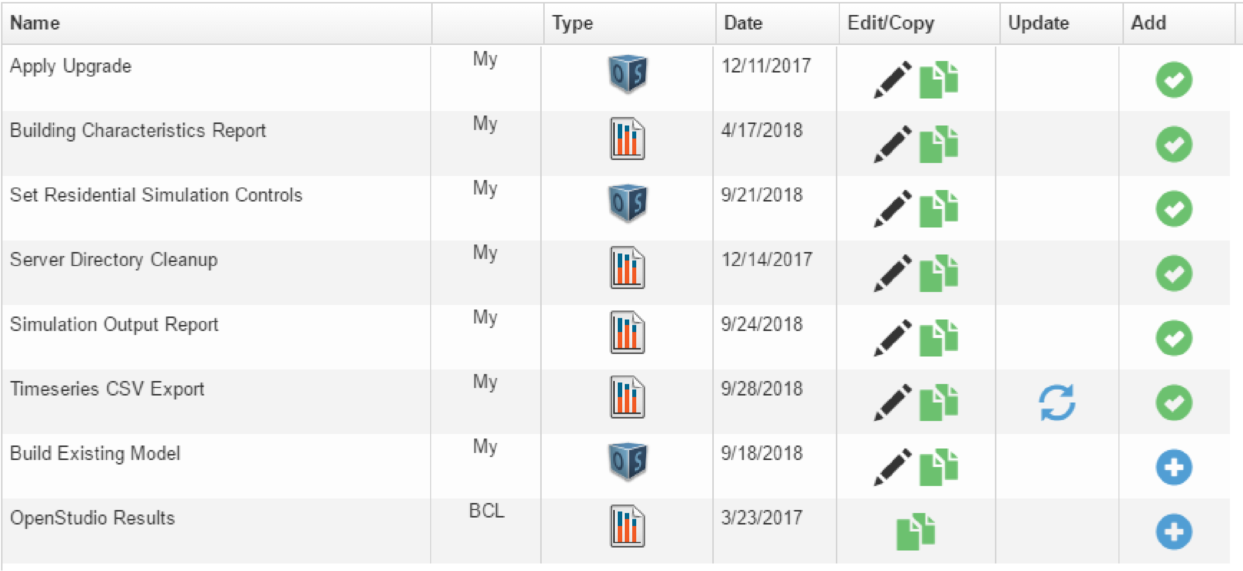
Once these two items have been completed, you are ready to start an AWS server and run your project. (See *Run the Project on Amazon Web Services* for more information.) Datapoint failures may be dealt with using instructions found in *Debugging*.

2.5.1 Check for Updates

You want project-level measures to be up-to-date with top-level measures (measures found in `<project_name>/measures` vs. those found in the top-level `measures` folder). To ensure this, click the blue "Check for Updates" button on the Measures Selection tab:



Then select any icons that appear in the Update column, as in the following example:



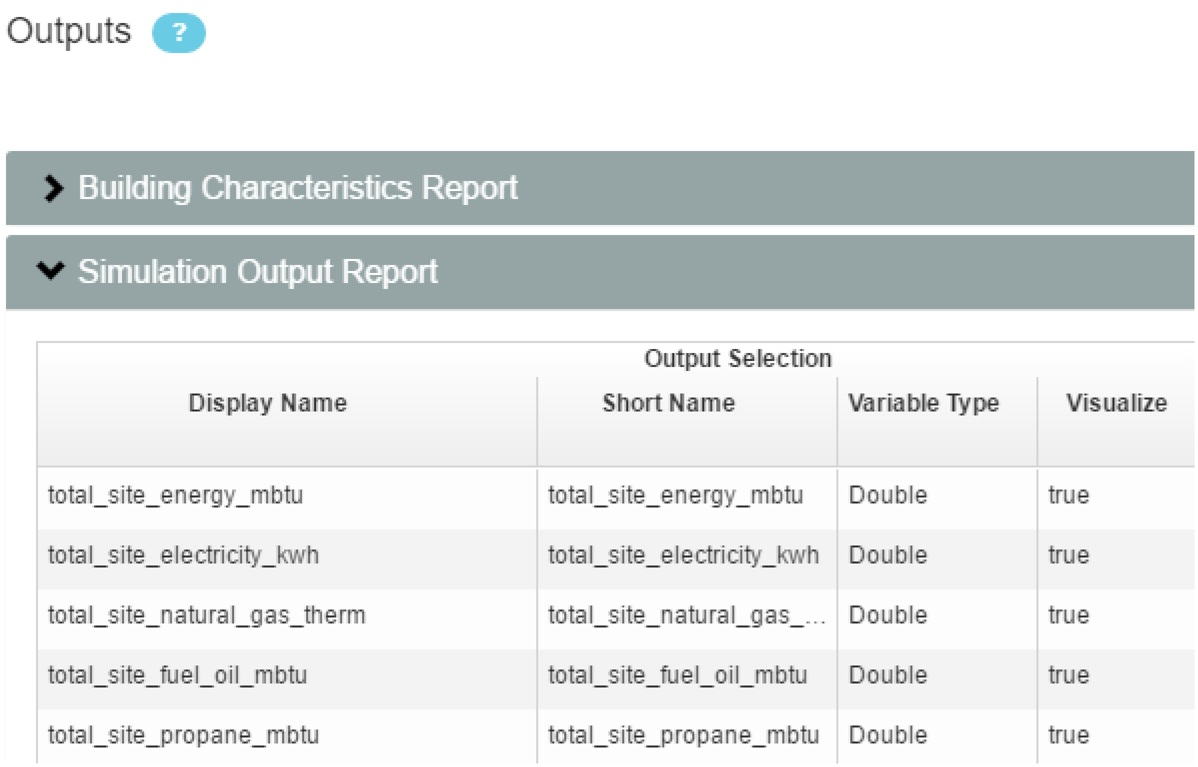
Name		Type	Date	Edit/Copy	Update	Add
Apply Upgrade	My	OS	12/11/2017	✎ 📄		✓
Building Characteristics Report	My	📊	4/17/2018	✎ 📄		✓
Set Residential Simulation Controls	My	OS	9/21/2018	✎ 📄		✓
Server Directory Cleanup	My	📊	12/14/2017	✎ 📄		✓
Simulation Output Report	My	📊	9/24/2018	✎ 📄		✓
Timeseries CSV Export	My	📊	9/28/2018	✎ 📄	↻	✓
Build Existing Model	My	OS	9/18/2018	✎ 📄		⊕
OpenStudio Results	BCL	📊	3/23/2017	📄		⊕

Click “Update Project” after each icon click. Click “OK” when done.

2.5.2 Refresh Outputs

If you made changes to either the Building Characteristics Report or Simulation Output Report measures, you will need to make updates on the Outputs tab. To do this remove the measure(s) from the project and re-add them, making sure they occupy the original position in the workflow order.

If you added or renamed any `housing_characteristics` tsv files, you will additionally need to make a change (e.g., add a whitepace) to `measure.rb` of Building Characteristics Report. Then navigate to the Outputs tab, select the measure(s) you’ve re-added, and then choose the outputs to include. Click “OK” when done. These outputs will be included in the summary results csv file.



	Output Selection		
Display Name	Short Name	Variable Type	Visualize
total_site_energy_mbtu	total_site_energy_mbtu	Double	true
total_site_electricity_kwh	total_site_electricity_kwh	Double	true
total_site_natural_gas_therm	total_site_natural_gas_...	Double	true
total_site_fuel_oil_mbtu	total_site_fuel_oil_mbtu	Double	true
total_site_propane_mbtu	total_site_propane_mbtu	Double	true

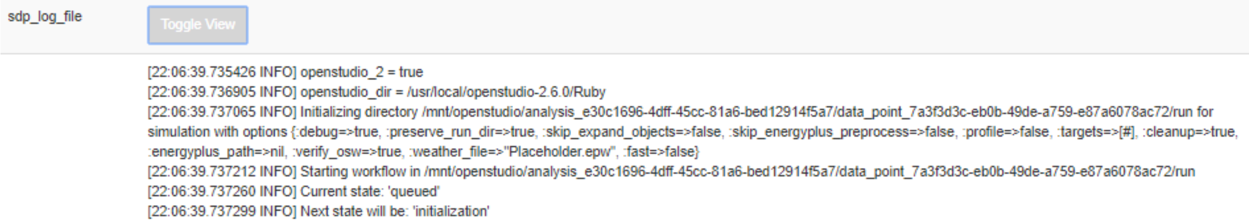
2.6 Debugging

Simulations may not always run successfully (even if the integrity checks pass). Reasons may include bad weather files, invalid measure arguments, untested measure combinations, etc. The OpenStudio-server GUI will indicate unsuccessful datapoints by showing “datapoint failure” under the “Status Message” column when you click on the homepage’s “View Analysis” button. Options for investigating causes of datapoint failures include:

- *Simulation Datapoint Log File*
- *Run Simulations Locally*

2.6.1 Simulation Datapoint Log File

To investigate the issues behind failed datapoints, click “View” for any datapoint failure rows. Scroll down and select the “sdp_log_file” button:



```
[22:06:39.735426 INFO] openstudio_2 = true
[22:06:39.736905 INFO] openstudio_dir = /usr/local/openstudio-2.6.0/Ruby
[22:06:39.737065 INFO] Initializing directory /mnt/openstudio/analysis_e30c1696-4dff-45cc-81a6-bed12914f5a7/data_point_7a3f3d3c-eb0b-49de-a759-e87a6078ac72/run for
simulation with options {:debug=>true, :preserve_run_dir=>true, :skip_expand_objects=>false, :skip_energyplus_preprocess=>false, :profile=>false, :targets=>[#], :cleanup=>true,
:energyplus_path=>nil, :verify_osw=>true, :weather_file=>"Placeholder.epw", :fast=>false}
[22:06:39.737212 INFO] Starting workflow in /mnt/openstudio/analysis_e30c1696-4dff-45cc-81a6-bed12914f5a7/data_point_7a3f3d3c-eb0b-49de-a759-e87a6078ac72/run
[22:06:39.737260 INFO] Current state: 'queued'
[22:06:39.737299 INFO] Next state will be: 'initialization'
```

Search the log’s text for the error. Diagnose the problem and fix it. Depending on where the issue originates, you may need to address problems upstream in the OpenStudio-BEopt’s `measures` directory. If that’s the case, you’ll probably want to start this entire development process over beginning with *Rake Tasks*. Bummer.

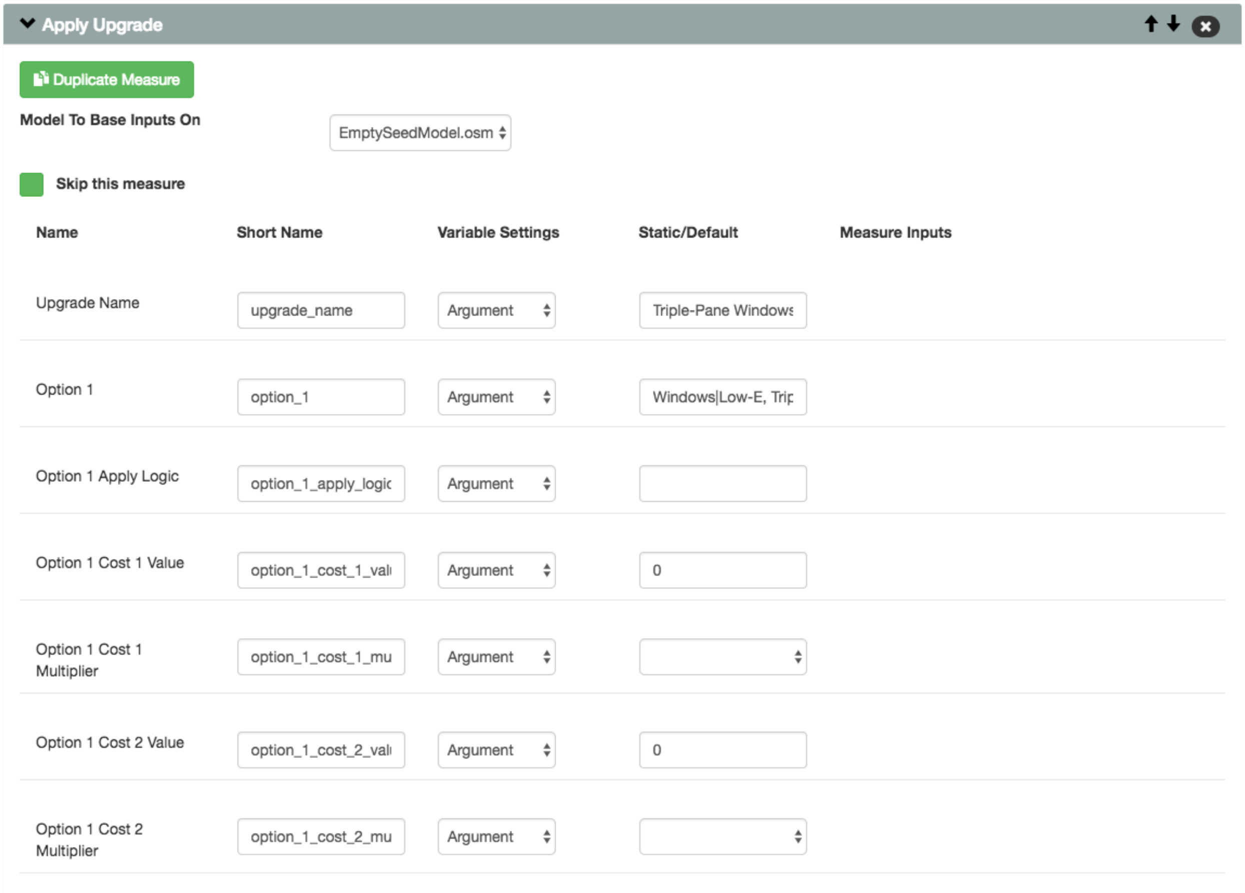
2.6.2 Run Simulations Locally

Each datapoint will come with a `measures.osw` file. You can pull these down from the server and run them locally.

CHAPTER 3

Upgrade Scenario Configuration

There is quite a bit more flexibility and capability in defining an upgrade scenario than was discussed in the *tutorial*. Here we will go through each field in the **Apply Upgrade** measure and discuss how it can be used to build more complicated real-life scenarios for upgrades.



Model To Base Inputs On: EmptySeedModel.osm

🟩 Skip this measure

Name	Short Name	Variable Settings	Static/Default	Measure Inputs
Upgrade Name	upgrade_name	Argument	Triple-Pane Windows	
Option 1	option_1	Argument	Windows|Low-E, Trip	
Option 1 Apply Logic	option_1_apply_logic	Argument		
Option 1 Cost 1 Value	option_1_cost_1_val	Argument	0	
Option 1 Cost 1 Multiplier	option_1_cost_1_mu	Argument		
Option 1 Cost 2 Value	option_1_cost_2_val	Argument	0	
Option 1 Cost 2 Multiplier	option_1_cost_2_mu	Argument		

3.1 Upgrade Name

This is a human readable name for the upgrade scenario. Something like, “Replace electric furnaces with Energy Star heat pumps” or “Insulate attics to R-49”.

3.2 Option <#>

In this field we enter the parameter and option combination to be applied. In the upgrade scenario simulations, this option will replace the option for the corresponding parameter in the baseline run. These can be found and referenced in the `resources/options_lookup.tsv` file in your local git repository. (You can see the most updated version on github here, but it’s recommended to use your local version as it will be synchronized with your project.) The file can be opened in a spreadsheet editor like Excel for viewing.

The text to enter in the field will be the Parameter Name followed by the Option Name separated by a pipe character.

```
Insulation Wall|Wood Stud, R-36
```

3.3 Option <#> Apply Logic

The apply logic field specifies the conditions under which the option will apply based on the baseline building’s options. To specify the condition(s) include one or more `parameter|option` pairs from `options_lookup.tsv`. Multiple option conditions can be joined using the following logical operators. Parentheses may be used as necessary as well.

`||`	logical OR
`&&`	logical AND
`!`	logical NOT

A few examples will illustrate. First, lets say we want the apply the option `Water Heater|Gas Tankless`, but only for water heaters that are worse and also use gas. We would use the following apply logic:

```
Water Heater|Gas Standard||Water Heater|Gas Benchmark
```

Or say we want to apply the upgrade only to houses with 3 car garages that aren’t in New England.

```
(!Location Census Division|New England)&&(Geometry Garage|3 Car)
```

Todo: Come up with some better examples here.

3.4 Option <#> Cost <#>

This is the cost of the upgrade. Multiple costs can be entered and each is multiplied by a cost multiplier, described below.

3.5 Option <#> Cost <#> Multiplier

The cost above is multiplied by this value, which is a function of the building. Since there can be multiple costs, this permits both fixed and variable costs for upgrades that depend on the properties of the baseline house.

- Fixed (1)
- Conditioned Floor Area (ft^2)
- Conditioned Foundation Slab Area (ft^2)
- Lighting Floor Area (ft^2)
- Above-Grade Conditioned Wall Area (ft^2)
- Above-Grade Total Wall Area (ft^2)
- Below-Grade Conditioned Wall Area (ft^2)
- Below-Grade Total Wall Area (ft^2)
- Window Area (ft^2)
- Roof Area (ft^2)
- Door Area (ft^2)
- Water Heater Tank Size (gal)
- HVAC Cooling Capacity (kBtuh)
- HVAC Heating Capacity (kBtuh)

3.6 Package Apply Logic

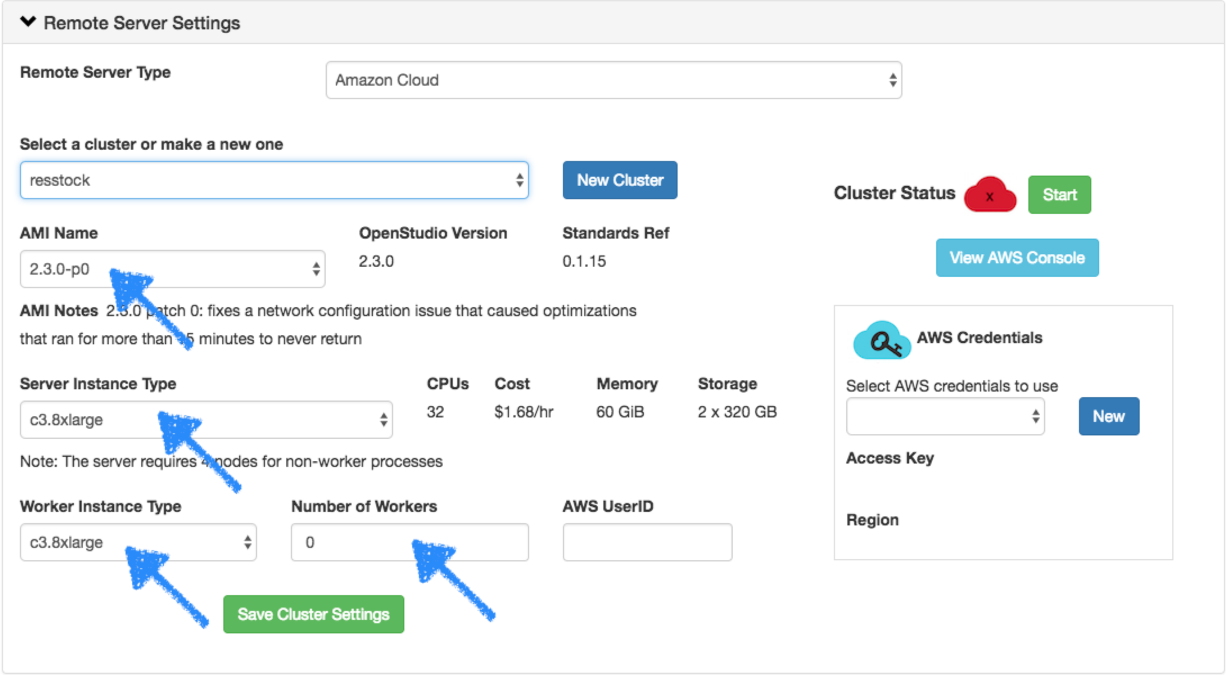
This is where to specifiy logic to determine whether the whole package of upgrades is applied (all of the options together). It uses the same format as *Option <#> Apply Logic*.

Todo: An example of when this might be useful would be nice.

CHAPTER 4

AWS Cluster Configuration

Depending on the size of your analysis, you can adjust selections for **Server Instance Type**, **Worker Instance Type**, and/or **Number of Workers**. Large analyses with many simulations may require more computing power. Keep in mind that more computing power may lead to faster analysis runtimes, but generally will cost more money.



Remote Server Settings

Remote Server Type: Amazon Cloud

Select a cluster or make a new one: resstock

AMI Name: 2.3.0-p0

OpenStudio Version: 2.3.0

Standards Ref: 0.1.15

AMI Notes 2.3.0 patch 0: fixes a network configuration issue that caused optimizations that ran for more than 45 minutes to never return

Server Instance Type	CPUs	Cost	Memory	Storage
c3.8xlarge	32	$1.68/hr	60 GiB	2 x 320 GB

Note: The server requires 4 nodes for non-worker processes

Worker Instance Type: c3.8xlarge

Number of Workers: 0

AWS UserID:

Cluster Status

AWS Credentials

Select AWS credentials to use

Access Key

Region

You can use the following guidance to decide what combination of settings makes sense for your analysis:

- For smaller analyses where the number of simulations is between 1 and 10,000, use the c3.8xlarge server and worker instance type. This instance type should be selected by default. You can also leave the number of workers at its default value of zero.
- For larger analyses where the number of simulations is between 1,000 and 100,000, use the d2.4xlarge server

instance type, c3.8xlarge worker instance type, and up to 10 workers. Using the d2.4xlarge server with more memory is necessary to manage larger analyses.