# Open Source dGen: Beta Release

Paritosh Das, Ben Sigrin, Trevor Stanley

dGen Beta Release Webinar

August 25th, 2020

# Discussion

1. Open Source Approach
2. Updates for Open Source
3. dGen Setup and Demo
4. Roadmap
5. Questions

# Overview

1. Open Source Approach
2. Updates for Open Source
3. dGen Setup and Demo
4. Roadmap
5. Questions

# Why Open Source?

## Transparency

- Access to underlying code and methods

## Flexibility

- A diversity of users with unique uses/research questions
- Ability to modify or change specific inputs and configurations

## Collaboration

- Development of new methods, technology models, datasets, etc.
- Feedback on the code and desired improvements

# dGen is Licensed Under BSD-3

## BSD 3-Clause License

- Allows for widespread use by any user
- Allows for modification
- Absolves NREL from liabilities
- https://github.com/NREL/dgen/blob/master/LICENSE.txt

### Balancing Public and Private Interest

Form partnerships that serve the public interest and advance U.S. Department of Energy goals. Demonstrate appropriate stewardship of publicly funded assets, yielding national benefits. Provide value to the commercial partner.

### Focusing on Outcomes

Develop mutually beneficial collaborations and align actions with business outcomes.

### Reflecting Core Values

Conduct technology partnership processes through professional practices, action, and a respect for duty. Align with the fundamental values of honesty, integrity, fairness, stewardship, and quality.

### Creating Transparency

Keep partners informed of goals, processes, decisions, and the status of actions as agreements are developed.

### Ensuring Confidentiality

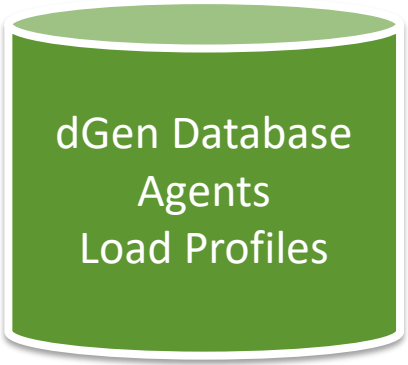
Maintain deep respect for proprietary business information and data.

### Seeking Continuous Improvement

Measure, monitor, and seek feedback about processes and outcomes. Use this information to improve processes and practices.

# dGen Architecture

**Inputs:**

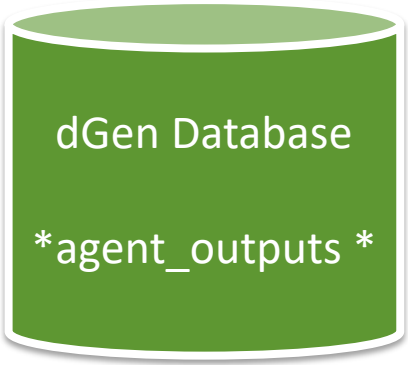



**dGen Scenarios:**
../input_scenarios/
../input_data/

**Code:**
Github.com/NREL/dgen
dgen_model.py Pseudocode:

```
For scenario:
  For year:
    For agent:
      - update agent attributes
      - calculate technical potential
      - calculate optimal system config
      - calculate DER bill savings
      - calculate adoption
```

**Outputs:**



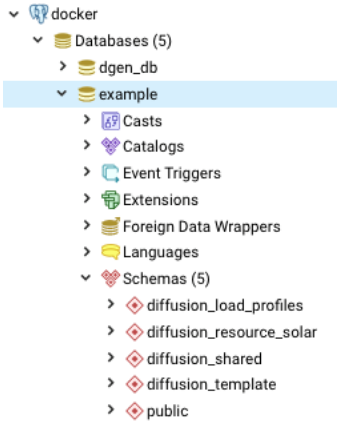
# dGen Agents

dGen agents are pre-generated and downloaded with the model database. Due to size restrictions, data is not stored on GitHub and is downloaded separately. Currently the national agent and database file are approximately 22 GB

dGen agents are *statistical*, meaning they are intended to statistically represent the underlying population. Each agent has a *weight* and are interpreted to represent N identical consumers. Agents are primarily segmented by sector and county. The agent location is modeled as the centroid of a Census block, which is population-weighted. More populous counties have corresponding more agents, with identical weights within the county.

New to the Beta release, dGen agents now use residential and commercial load profiles simulated using the NREL ResStock and ComStock tools, which use a physics-based engine (OpenStudio) to simulate energy consumption

ResStock + ComStock



# Modularity

## PostgreSQL Database within a Docker Container

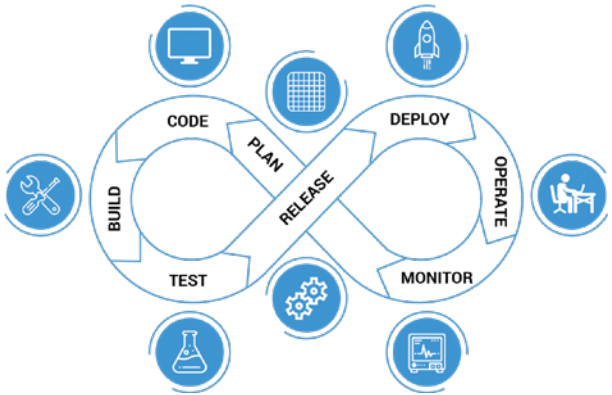
- Allows for easy isolation of postgreSQL servers that can be “leanly” configured
- Portability of Docker Containers enables mounting of database on existing and diverse compute infrastructure
- Diverse and flexible configurations

## Anaconda

- Easy package management and environment creation
- Suite of development, debugging, and analysis tools

## Github

- Branching, version control, submitting issues, and more enables contiguous and efficient workflows for specific projects



# Improvements

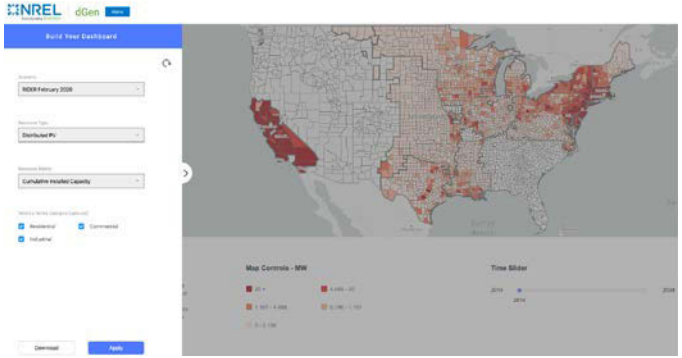
# Open Source Updates (part 1)

## PySAM Integration

- Enables enhanced modeling of solar + storage
- Required upgrade from Python v2 to Python v3

## Developed Visualizer Web Application

- To be launched in September
- Scenario building and visualization




## Model Calibration

- To be launched in September
- Automated process to calibrate agent decision-making using historic data

# Updates (part 2)

## PySAM Integration

- Enables enhanced modeling of DER performance and financials
  - E.g. Simulation of optimal combinations of solar and/or storage system configurations based on maximizing customer bill savings
  - Storage can be dispatched by heuristic, e.g revenue maximization, peak-shaving

- Speeds up certain financial and optimization functions within dGen
  - Upgrade to PySAM + Python 3 allows dGen to utilize faster multi-processing techniques
  - PySAM integration reduced redundancy between dGen and SAM

- PySAM improves user's ability to simulate multiple new technology and financial model modules.

# Workflow

1 Open Source Approach

2 Updates for Open Source

3 **dGen Setup and Demo**

4 Roadmap

5 Questions

# Primary Tools

## Anaconda Python 3.7

- Easy install and management of packages

## Docker

- Simplicity and portability
- PostgreSQL server and database (DB) will be housed in a portable container
- Portability will enable agnostic cloud services use in future

## PgAdmin

- Used for easier interaction with the PostgreSQL DB and interaction with results

## Github

- Version control, workflow efficiency, and centralization of code. You will need to make an account.

# Primary Tools

## Anaconda Python 3.7

- Install for your operating system: https://www.anaconda.com/distribution/

## Docker

- Install for Mac: https://docs.docker.com/docker-for-mac/install/
- Install for Windows: https://docs.docker.com/docker-for-windows/install/

## PgAdmin

- Version 4 is recommended: https://www.pgadmin.org/download/

## Github

- dGen Code: https://github.nrel.com/dgen

  [For now, must email NREL for access]
- dGen Input Data (22 GB): https://app.box.com/s/9zx58ojj0hhwr3b59xhanvmzimp06bgt

  [To be updated for final release]

# Initial Setup (part 1)

## Download Code:

- Install git/bash if not installed: https://www.atlassian.com/git/tutorials/install-git
- Fork the repository to your own Github account
- $ git clone https://github.nrel.com/ <github_username_here/dgen.git

## Step A: Create Environment

- Navigate to the python folder in the cloned repository and run this command:

```
(base) tstanley-33754s:python tstanley$ conda env create -f dg3n.yml
```

- Activate the environment & run Spyder

```
(base) tstanley-33754s:~ tstanley$ conda activate dg3n
(dg3n) tstanley-33754s:~ tstanley$ spyder
```

## Step B: Data Download

- https://app.box.com/s/9zx58ojj0hhwr3b59xhanvmzimp06bgt)
- Unzip files once downloaded

# Initial Setup (part 2)

## Docker: PostgreSQL Database

- Creating the container

RUN: docker run --name postgis_2_2 -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -d mdillon/postgis

```
(base) tstanley-33754s:~ tstanley$ docker run --name postgis_2_2 -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -d mdillon/postgis
452b560df7e8ba7e7d679e4bddefb29e377693748a5c49812a92cac6a2181232
(base) tstanley-33754s:~ tstanley$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
452b560df7e8        mdillon/postgis     "docker-entrypoint.s…"   6 seconds ago       Up 5 seconds        0.0.0.0:5432->5432/tcp   postgis_2_2
```

- Execute and enter the container

RUN: docker exec -it 452b560df7e8 psql -U postgres

```
(base) tstanley-33754s:~ tstanley$ docker exec -it 452b560df7e8 psql -U postgres
psql (11.2 (Debian 11.2-1.pgdg90+1))
Type "help" for help.

postgres=# \l
                                  List of databases
       Name       |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
------------------+----------+----------+------------+------------+-----------------------
 postgres         | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0        | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                  |          |          |            |            | postgres=CTc/postgres
 template1        | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                  |          |          |            |            | postgres=CTc/postgres
 template_postgis | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
(4 rows)
```

# Initial Setup (reference 3)

## Docker: PostgreSQL Database Continued

- Create & connect to a new database

  RUN: CREATE DATABASE restore_cntnr2_db6;
  RUN: \c restore_cntnr2_db6

```
postgres=# CREATE DATABASE restore_cntnr2_db6;
CREATE DATABASE
postgres=# \l
                                  List of databases
        Name         |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
---------------------+----------+----------+------------+------------+-----------------------
 postgres            | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 restore_cntnr2_db6  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0           | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                     |          |          |            |            | postgres=CTc/postgres
 template1           | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                     |          |          |            |            | postgres=CTc/postgres
 template_postgis    | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
(5 rows)

postgres=# \c restore_cntnr2_db6
You are now connected to database "restore_cntnr2_db6" as user "postgres".
```

- Restoring the DB

RUN: cat /path_to_where_you_saved_data/dgen_alpha_os_db_postgres.sql | docker exec -i <container id> psql -U postgres -d dgen_db

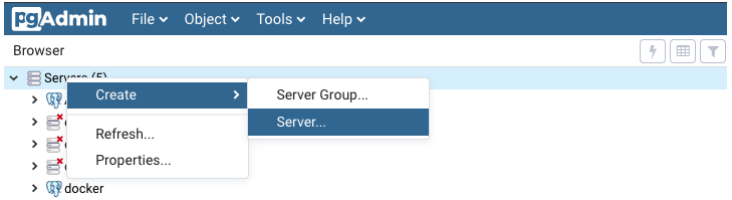
```
tstanley — -bash — 192×29
(base) tstanley-33754s:~ tstanley$ cat /Users/tstanley/Documents/dGen_Materials/OpenSourcing/dgen_os_db_postgres4.sql | docker exec -i 452b560df7e8 psql -U postgres -d restore_cntnr2_db6
SET
SET
SET
SET
SET
 set_config
------------
(1 row)

SET
SET
SET
SET
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
CREATE SCHEMA
ALTER SCHEMA
```

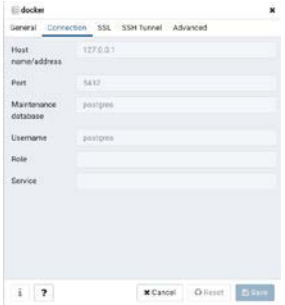
# Initial Setup (reference 4)

## PgAdmin

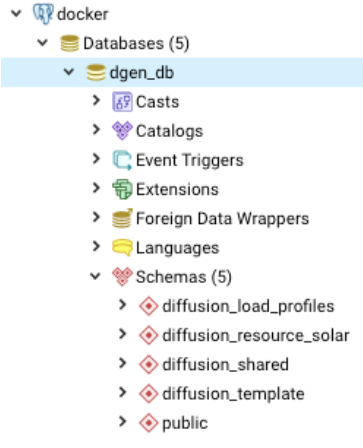
- Launch pgadmin & create a new server

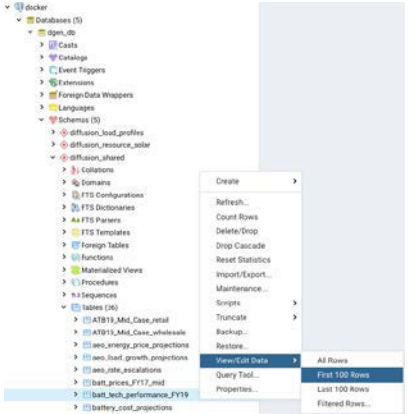



- Configure the server as shown and save


- Examining the data



# Initial Setup (reference 5)

## Configuring dGen Code

- Pg_params_atlas.json
  - Open pg_params_atlas.json in the python folder
  - Change ”dbname” to the name of your database

```
1  {
2      "dbname": "dgen_db",
3      "host": "127.0.0.1",
4      "port": "5432",
5      "user": "postgres",
6      "password": "postgres"
7  }
```

- Config.py
  - Can change the start year, number of agents per region, number of cores to run on, and the number of parallel processes

- Settings.py and data_functions.py
  - Default ”role” variable set to postgres (same as the default owner of the database)
  - If you change the ownership of the database, make sure the “role” variable is changed in kind

# Initial Setup (reference 6)

## Input Data

- Save agent file to the input_data folder
  - In this example ”agent_df_base_081819_DE.pkl” is used

## Input Sheet

- Configuring the drop downs and agent pkl file name
- Save to the ‘input_scenarios’ folder

**dGen Model Input**

| Scenario Options | Value | User Defined File |
|---|---|---|
| Scenario Name | reference | |
| Technology | Solar Only | |
| Agent File | Use pre-generated Agents | agent_df_base_081819_DE |
| Region to Analyze (If generating new agents) | Delaware | |
| Markets (If generating new agents) | All | |
| Analysis End Year | 2020 | |
| Load Growth Scenario | AEO2019 Reference | |
| Retail Electricity Price Escalation Scenario | User Defined | ATB19_Mid_Case_retail |
| Wholesale Electricity Price Scenario | User Defined | ATB19_Mid_Case_wholesale |
| PV Price Scenario | User Defined | pv_price_atb19_mid |
| PV Technical Performance Scenario | User Defined | pv_tech_performance_defaultFY19 |
| Storage Cost Scenario | User Defined | batt_prices_FY17_mid |
| Storage Technical Performance Scenario | User Defined | batt_tech_performance_FY19 |
| Financing Scenario | User Defined | financing_atb_FY19 |
| Depreciation Scenario | User Defined | deprec_sch_FY19 |
| Carbon Intensity Scenario | User Defined | carbon_intensities_FY19 |
| Random Generator Seed | 1 | |

Save Scenario

**dGen Model Input**

| Scenario Options | Value | User Defined File |
|---|---|---|
| Scenario Name | reference | |
| Technology | Solar Only | |
| Agent File | Use pre-generated Agents | agent_df_base_081819_DE |
| Region to Analyze (If generating new agents) | Delaware | |
| Markets (If generating new agents) | All | |
| Analysis End Year | 2020 | |
| Load Growth Scenario | AEO2019 Reference | |
| Retail Electricity Price Escalation Scenario | | ATB19_Mid_Case_retail |
| Wholesale Electricity Price Scenario | | ATB19_Mid_Case_wholesale |
| PV Price Scenario | | pv_price_atb19_mid |
| PV Technical Performance Scenario | | pv_tech_performance_defaultFY19 |
| Storage Cost Scenario | | batt_prices_FY17_mid |
| Storage Technical Performance Scenario | | batt_tech_performance_FY19 |
| Financing Scenario | | financing_atb_FY19 |
| Depreciation Scenario | | deprec_sch_FY19 |
| Carbon Intensity Scenario | | carbon_intensities_FY19 |
| Random Generator Seed | | |

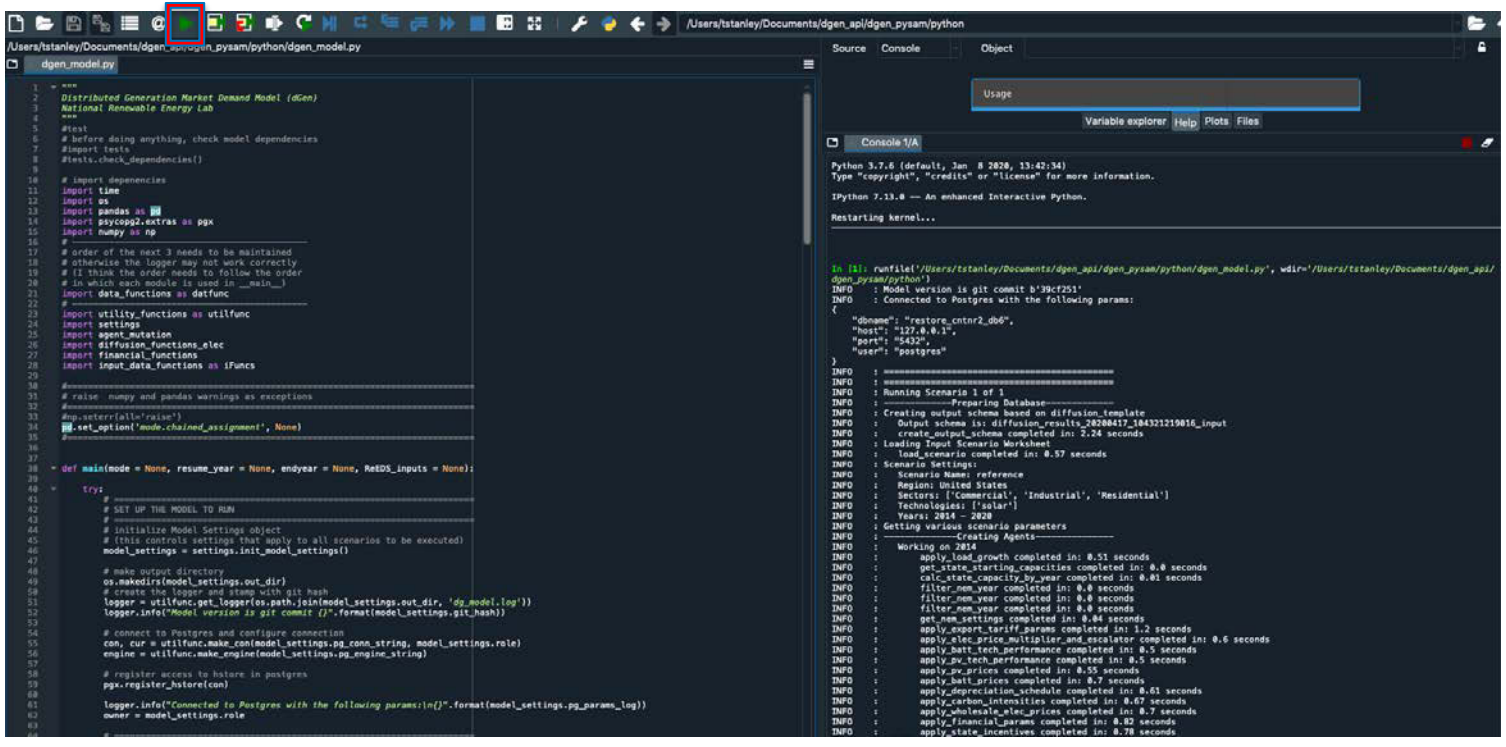
AEO2019 Reference
AEO2019 Low Price
AEO2019 High Price
AEO2019 Low Growth
AEO2019 High Growth
AEO2019 Low Resource
AEO2019 High Resource
AEO2018 No CPP
User Defined

Save Scenario

# Initial Setup (reference 7)

## Running dGen

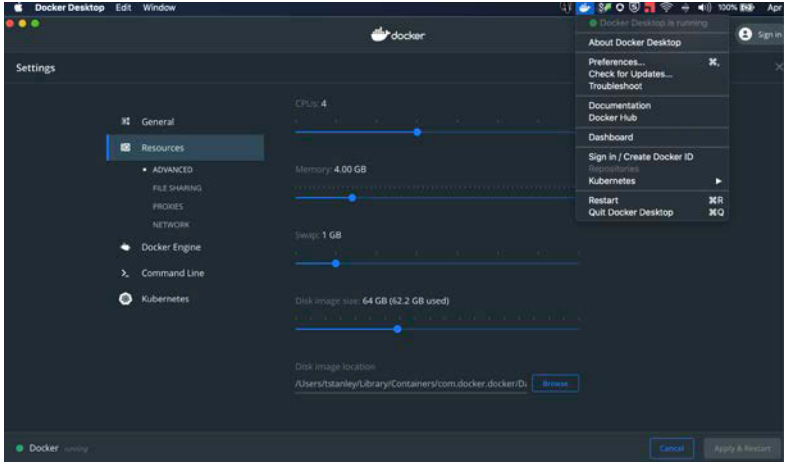
- Open ’dgen_model.py’ in spyder and press play



# Things To Highlight & Best Practices

Increase/Clear the memory allocated to “disk image size” in Docker

- If getting a memory issue when trying to restore the database, run the following commands in terminal:
  - $ docker image prune
  - $ docker container prune
  - $ docker system prune –volumes




## PostgreSQL Tips & Tricks

- $ \l will list all of the databases in your server
- $ \c <database> connects to the database
- $ CREATE DATABASE <database_name>; creates done prior to restoring the database backup file
- $ docker stop <container_id> to pause the DB
- $docker start <container_id> to start the DB again

# Roadmap

# Roadmap

## September Full Release:

- Patch issues identified in Beta
- Integrate new model calibration module
- Launch dGen Scenario Viewer

## Upcoming reports:

- Load profiles methodology
- Model calibration methodology

## Cloud Access:

- Goal: ability to run portable docker images of the model with high performance compute (HPC) capabilities

Annual cycle for model versioning and agent datasets

# We're Looking for You

## How you can help:

- Report bugs
- Suggest improvements for usability
- Recommend colleagues to join our mailing list
- Let us know how you used dGen in your work or research

## Want to join our developer community?

We're looking for individuals and institutions interested in building out and improving the core dGen code, algorithms, and data. If you're interested in advancing the state-of-art in DER modeling and are committed to the open-source approach, email us at dgen@NREL.gov. The procedure for submitting GitHub issues and pull requests is on the dgen repo.

**NREL can provide technical assistance and training on a case-by-case basis**

# Consideration

1 Open Source Approach

2 Updates for Open Source

3 dGen Setup and Demo

4 Roadmap

5 **Questions**

# Thank you!

**www.nrel.gov**

NREL/PR-6A20-77705

dGen@NREL.gov

Trevor Stanley - Trevor.Stanley@NREL.gov

Ben Sigrin – Benjamin.Sigrin@NREL.gov

Paritosh Das – Paritosh.Das@NREL.gov

Website: nrel.gov/analysis/dgen/

Documentation: nrel.gov/docs/fy16osti/65231.pdf

Publications: nrel.gov/analysis/dgen/publications.html

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Solar Technologies Program. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

NREL Transforming ENERGY