



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.

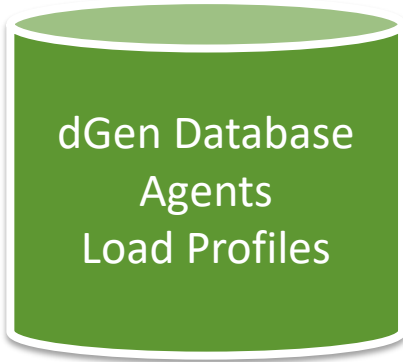


Seeking Continuous Improvement

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:

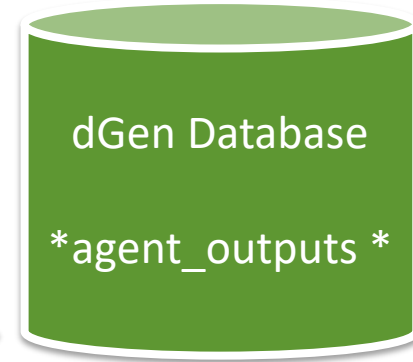
For scenario:

For year:

For agent:

- update agent attributes
- calculate technical potential
- calculate optimal system config
- calculate DER bill savings
- calculate adoption

Outputs:

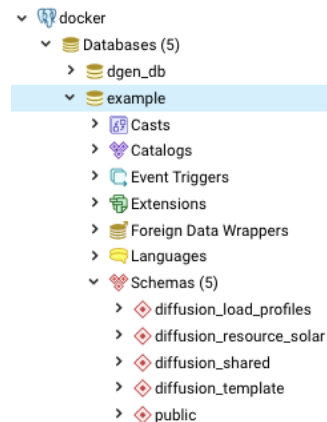


dGen Agents

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



Modularity

PostgreSQL Database within a Docker Container

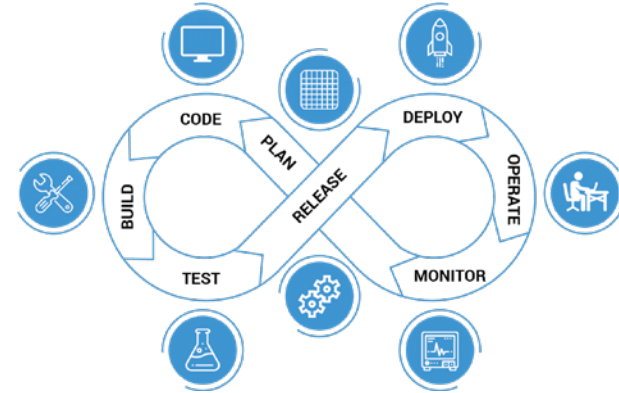
- Allows for easy isolation of postgresQL servers that can be “leanly” configured
- Portability of Docker Containers enables mounting of database on existing and diverse compute infrastructure
- Diverse and flexible configurations

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

1 Open Source Approach

2 dGen Updates for Open Source

3 dGen Setup and Demo

4 Roadmap

5 Questions

Open Source Updates (part 1)

PySAM Integration

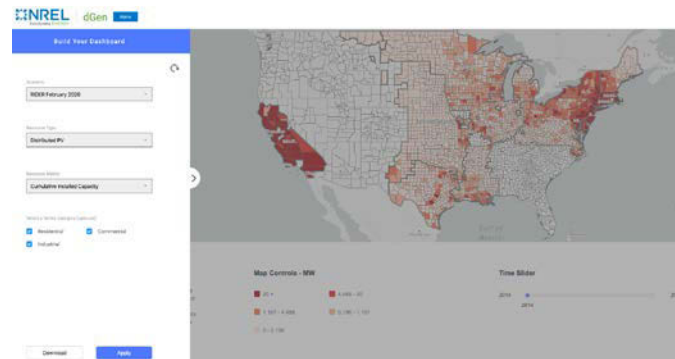
- Enables enhanced modeling of solar + storage
- Required upgrade from Python v2 to Python v3

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

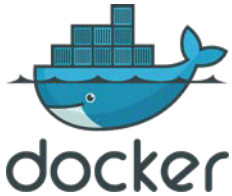
5 Questions

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

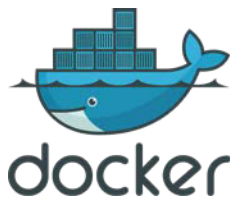
- Version control, workflow efficiency, and centralization of code. You will need to make an account.

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 +
template1	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres +
template_postgis	postgres	UTF8	en_US.utf8	en_US.utf8	postgres=Ctc/postgres

```
(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

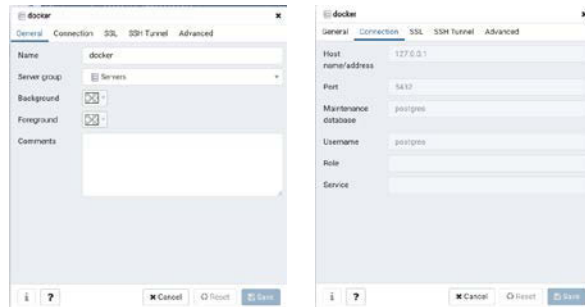
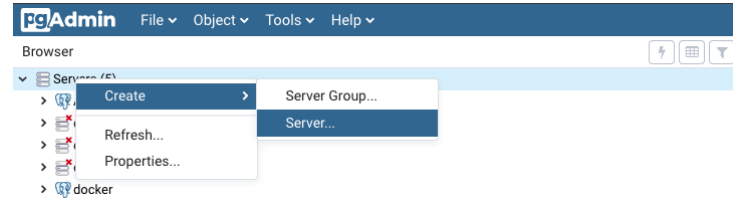
RUN: cat
/path_to_where_you_saved_data/dgen_alpha_os_db_postgres.sql | docker exec -i <container id> psql -U postgres -d dgen_db

```
tstanley -- bash -- 192x29  
(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  
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
```

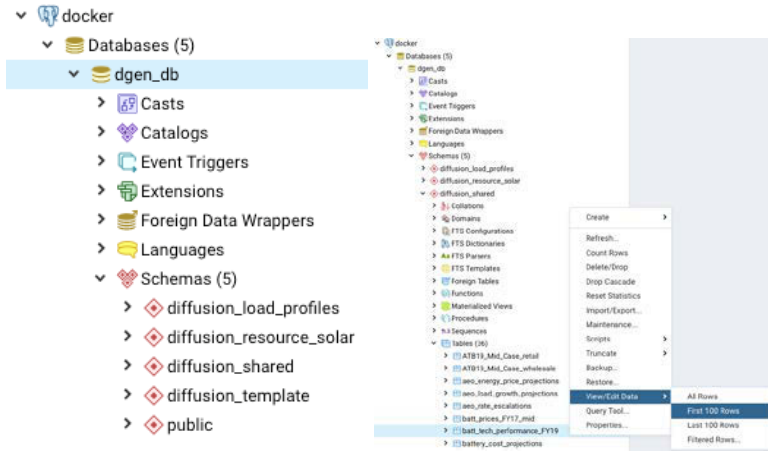
Initial Setup (reference 4)

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
- 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

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

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

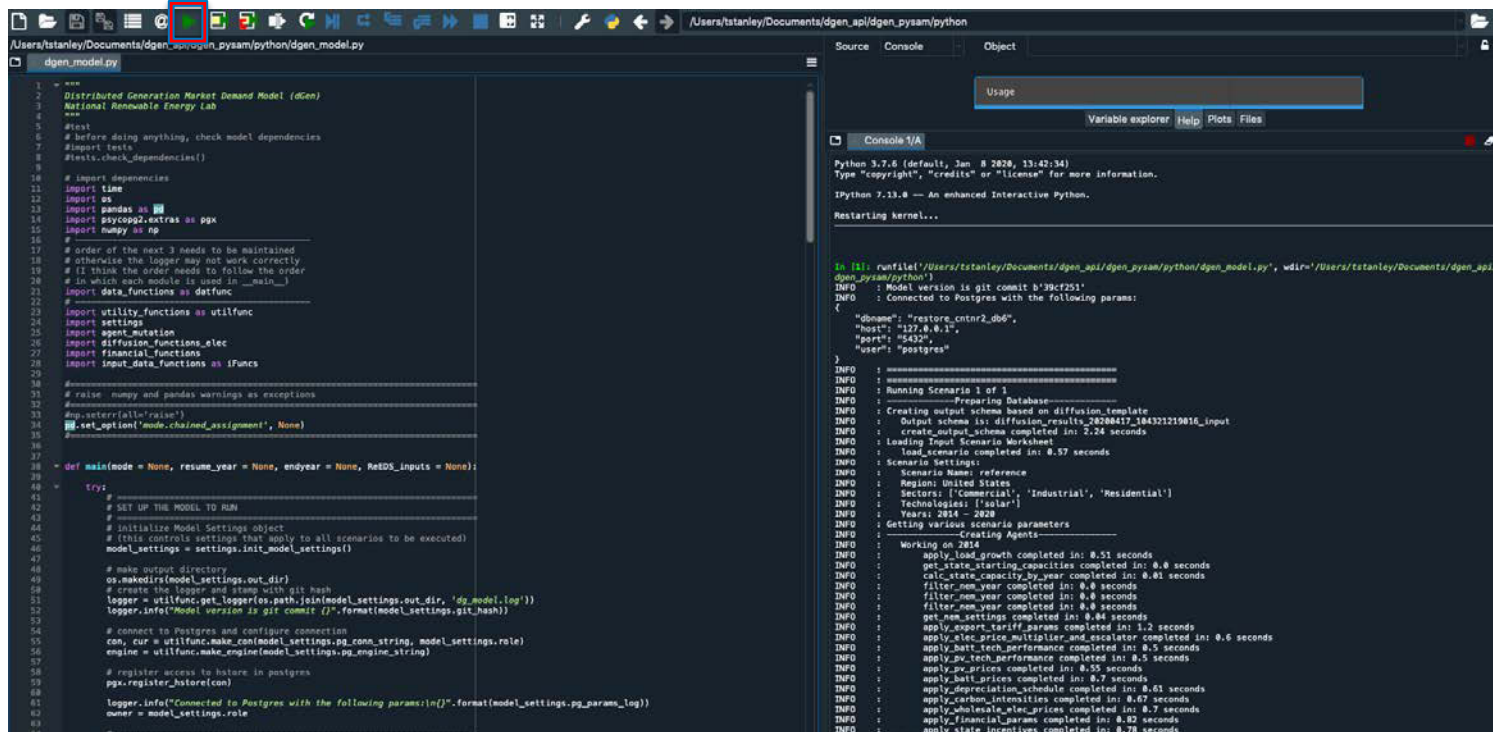
Scenario Options	Value	User Defined File
Scenario Name	reference	
Technology	Solar Only	
Agent File	Use pre-generated Agents	agent_df_base_081819_DE
Region to Analyze (If generating new agents)	Delaware	
Markets (If generating new agents)	All	
Analysis End Year	2020	
Load Growth Scenario	AEO2019 Reference	
Retail Electricity Price Escalation Scenario	AEO2019 Reference	ATB19_Mid_Case_retail
Wholesale Electricity Price Scenario	AEO2019 Reference	ATB19_Mid_Case_wholesale
PV Price Scenario	AEO2019 Low Price	pv_price_atb19_mid
PV Technical Performance Scenario	AEO2019 High Price	pv_tech_performance_defaultFY19
Storage Cost Scenario	AEO2019 Low Growth	batt_prices_FY17_mid
Storage Technical Performance Scenario	AEO2019 High Growth	batt_tech_performance_FY19
Financing Scenario	AEO2019 Low Resource	financing_atb_FY19
Depreciation Scenario	AEO2019 High Resource	deprec_sch_FY19
Carbon Intensity Scenario	AEO2018 No CPP	carbon_intensities_FY19
Random Generator Seed	User Defined	

Save Scenario

Initial Setup (reference 7)

Running dGen

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



```
1  """
2  Distributed Generation Market Demand Model (dGen)
3  National Renewable Energy Lab
4  """
5
6  # before doing anything, check model dependencies
7  #import tests
8  #tests.check_dependencies()
9
10 # import dependencies
11 import time
12 import os
13 import pandas as pd
14 import sqlalchemy as sa
15 import numpy as np
16 #
17 # order of the next 3 needs to be maintained
18 # otherwise the logger may not work correctly
19 # if I think the order needs to follow the order
20 # in which each module is used in __main__
21 import data_functions as datfunc
22 #
23 import utility_functions as utilfunc
24 import settings
25 import agent_mutation
26 import diffusion_functions_elec
27 import financial_functions
28 import input_data_functions as ifuncs
29
30 # raise numpy and pandas warnings as exceptions
31 #
32 # @pytest.raises(ValueError)
33 @pytest.raises(ValueError)
34 @pytest.raises(ValueError)
35 @pytest.raises(ValueError)
36 @pytest.raises(ValueError)
37
38 def main(mode = None, resume_year = None, endyear = None, ReDOS_inputs = None):
39     try:
40         #
41         # SET UP THE MODEL TO RUN
42         #
43         # initialize Model Settings object
44         # (this controls settings that apply to all scenarios to be executed)
45         model_settings = settings.init_model_settings()
46
47         # make output directory
48         os.makedirs(model_settings.out_dir)
49         # create the logger and stamp with git hash
50         logger = utilfunc.get_logger(os.path.join(model_settings.out_dir, 'dgen_model.log'))
51         logger.info("Model version is git commit {}".format(model_settings.git_hash))
52
53         # connect to Postgres and configure connection
54         con, cur = utilfunc.make_con(model_settings.pg_conn_string, model_settings.role)
55         engine = utilfunc.make_engine(model_settings.pg_conn_string)
56
57         # register access to history in postgres
58         pgx.register_history(con)
59
60         logger.info("Connected to Postgres with the following params:{}".format(model_settings.pg_params_log))
61         owner = model_settings.role
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type "copyright", "credits" or "license()" for more.

IPython 7.13.0 -- An enhanced Interactive Python.
Restarting kernel...

In [1]: runfile('/Users/tstanley/Documents/dgen_api/dgen_pyasm/python/dgen_model.py', wdir='/Users/tstanley/Documents/dgen_api/dgen_pyasm/python')
INFO : Model version is git commit b'39c7251'
INFO : Connected to Postgres with the following params:
{
  "dbname": "restore_ctr2_db6",
  "host": "127.0.0.1",
  "port": "5432",
  "user": "postgres"
}
INFO : =====
INFO : Running Scenario 1 of 1
INFO : Preparing Database
INFO : Creating output schema based on diffusion_template
INFO : Output schema is: diffusion_results_20200417_104321219816_input
INFO : create_output_schema completed in: 2.24 seconds
INFO : Loading Input Scenario Worksheet
INFO : load_scenario completed in: 0.57 seconds
INFO : Scenario Settings:
INFO : Scenario Name: reference
INFO : Region: United States
INFO : Sectors: ['Commercial', 'Industrial', 'Residential']
INFO : Technologies: ['Solar']
INFO : Years: 2014 - 2020
INFO : Getting various scenario parameters
INFO : Creating Agents
INFO : Working on 2014
INFO : apply_load_growth completed in: 0.51 seconds
INFO : get_state_starting_capacities completed in: 0.0 seconds
INFO : calc_state_capacity_by_year completed in: 0.01 seconds
INFO : filter_non_year completed in: 0.0 seconds
INFO : filter_non_year completed in: 0.0 seconds
INFO : filter_non_year completed in: 0.0 seconds
INFO : filter_non_year completed in: 0.0 seconds
INFO : get_new_settings completed in: 0.04 seconds
INFO : apply_export_tariff_params completed in: 1.2 seconds
INFO : apply_elec_price_multiplier_and_escalator completed in: 0.6 seconds
INFO : apply_bit_tech_performance completed in: 0.5 seconds
INFO : apply_pv_tech_performance completed in: 0.5 seconds
INFO : apply_pv_prices completed in: 0.35 seconds
INFO : apply_bit_prices completed in: 0.7 seconds
INFO : apply_depreciation_schedule completed in: 0.61 seconds
INFO : apply_carbon_intensities completed in: 0.67 seconds
INFO : apply_wholesale_elec_prices completed in: 0.7 seconds
INFO : apply_financial_params completed in: 0.82 seconds
INFO : apply_state_incentives completed in: 0.78 seconds
```

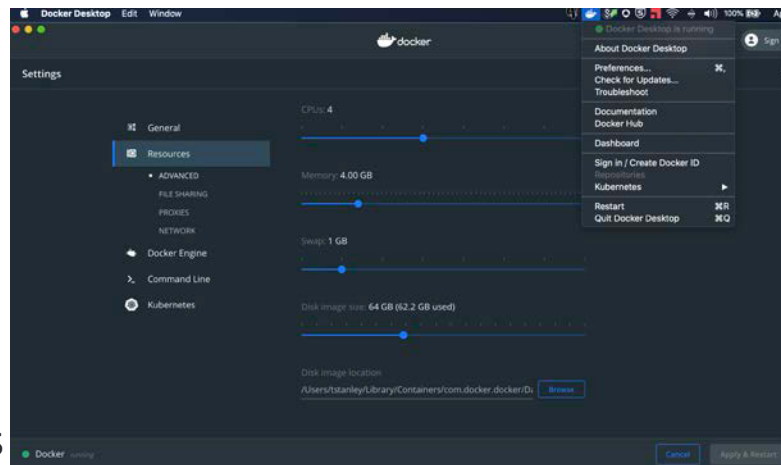
Things To Highlight & Best Practices

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

1 Open Source Approach

2 Updates for Open Source

3 dGen Setup and Demo

4 Roadmap

5 Questions

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.

