Pushing test lab to its limits

Paweł Wieczorek

FOSDEM'24

COLLABORA

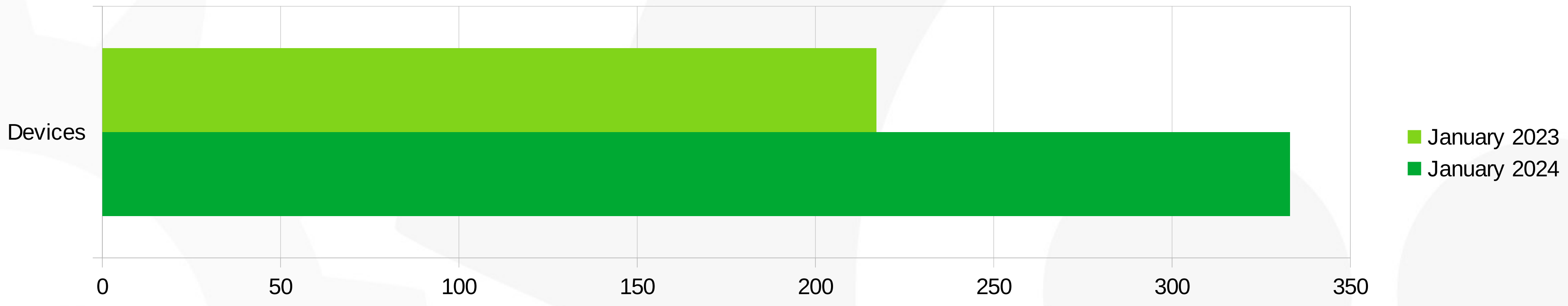
Agenda

- Background
- Interactive approaches
- Automation solutions
- Data generation

Background

Continuously growing a lab

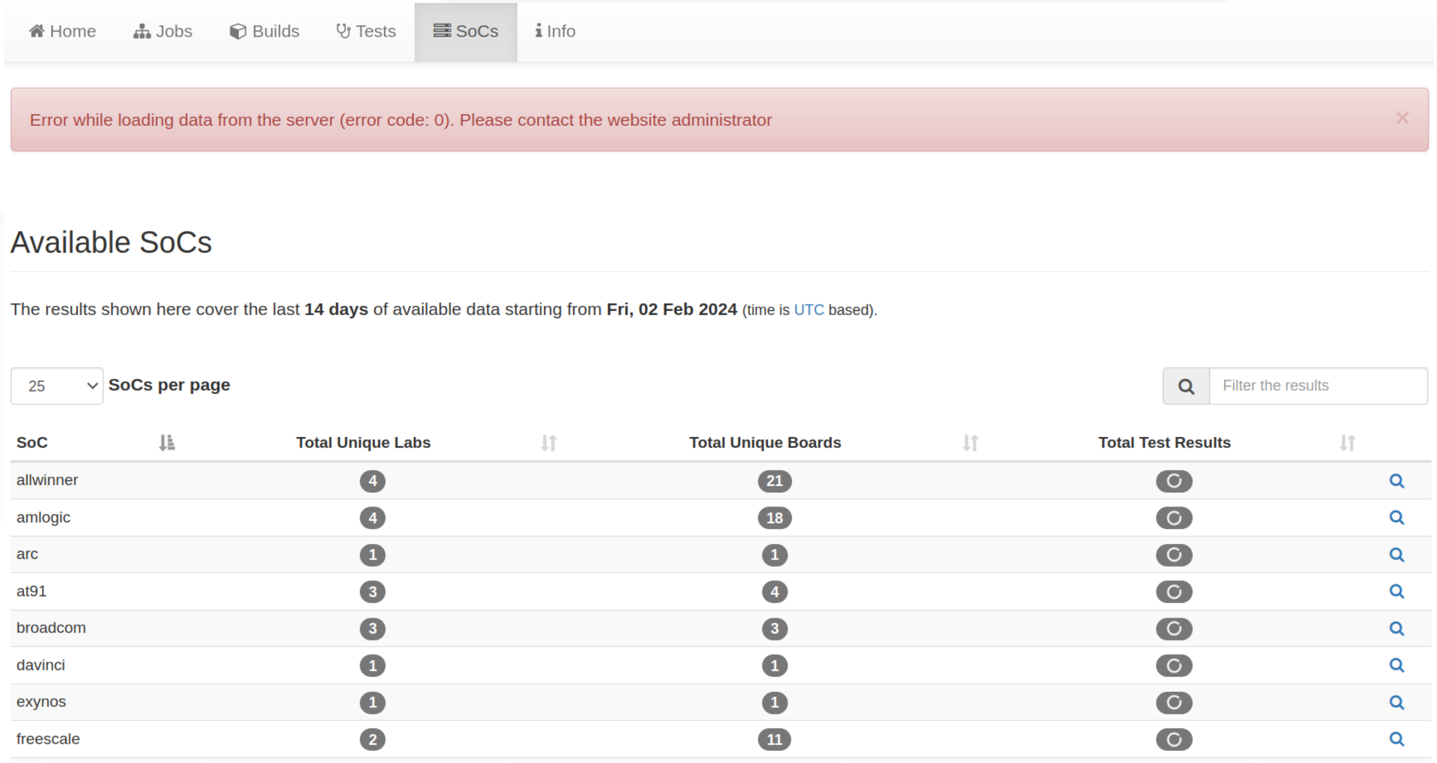
Growing a lab for automated upstream testing, Laura Nao



Unusually high load

- No reason to panic – allocated resources are in use
- Highest on the nodes running database processes

Or is it?



Error while loading data from the server (error code: 0). Please contact the website administrator

Available SoCs

The results shown here cover the last **14 days** of available data starting from **Fri, 02 Feb 2024** (time is UTC based).

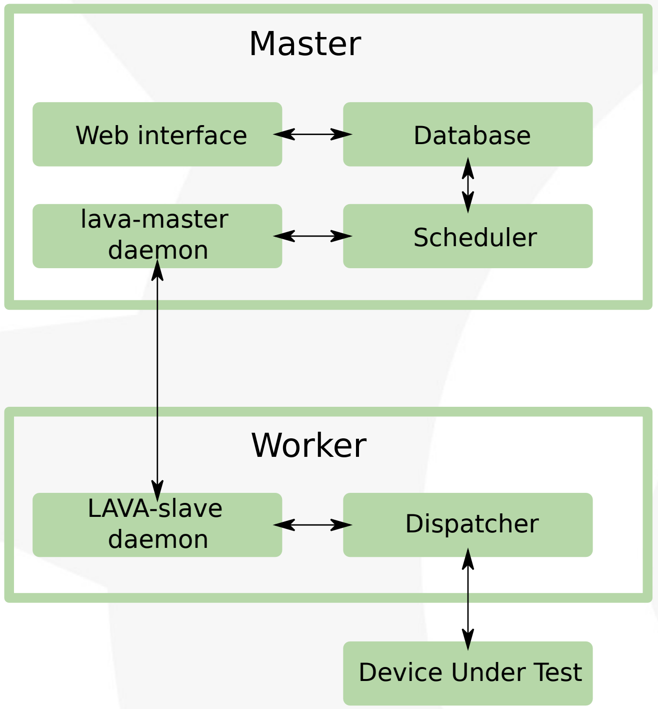
25 **SoCs per page**

Filter the results

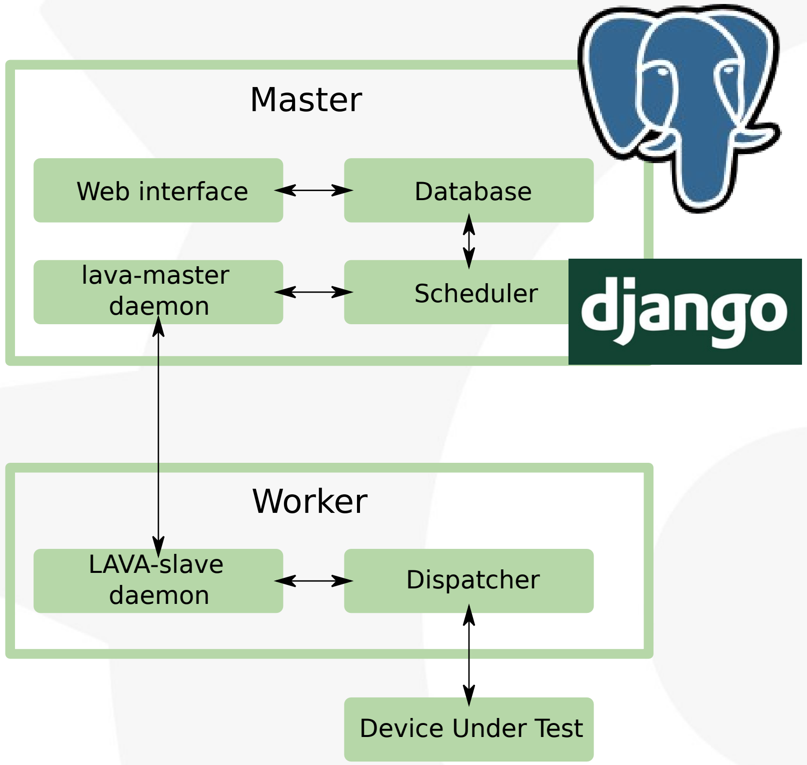
SoC	Total Unique Labs	Total Unique Boards	Total Test Results	
allwinner	4	21		
amlogic	4	18		
arc	1	1		
at91	3	4		
broadcom	3	3		
davinci	1	1		
exynos	1	1		
freescale	2	11		

Throw in more resources

LAVA architecture

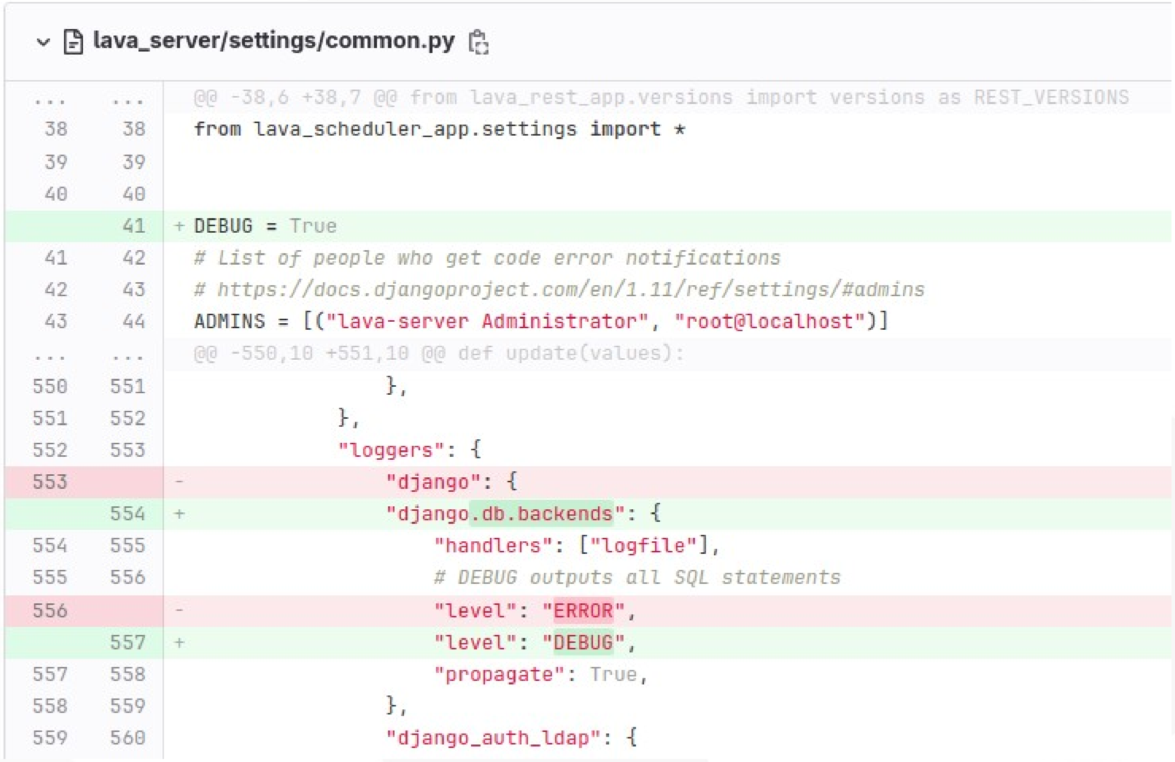


LAVA under the hood



Interactive approaches

When in doubt – check log out

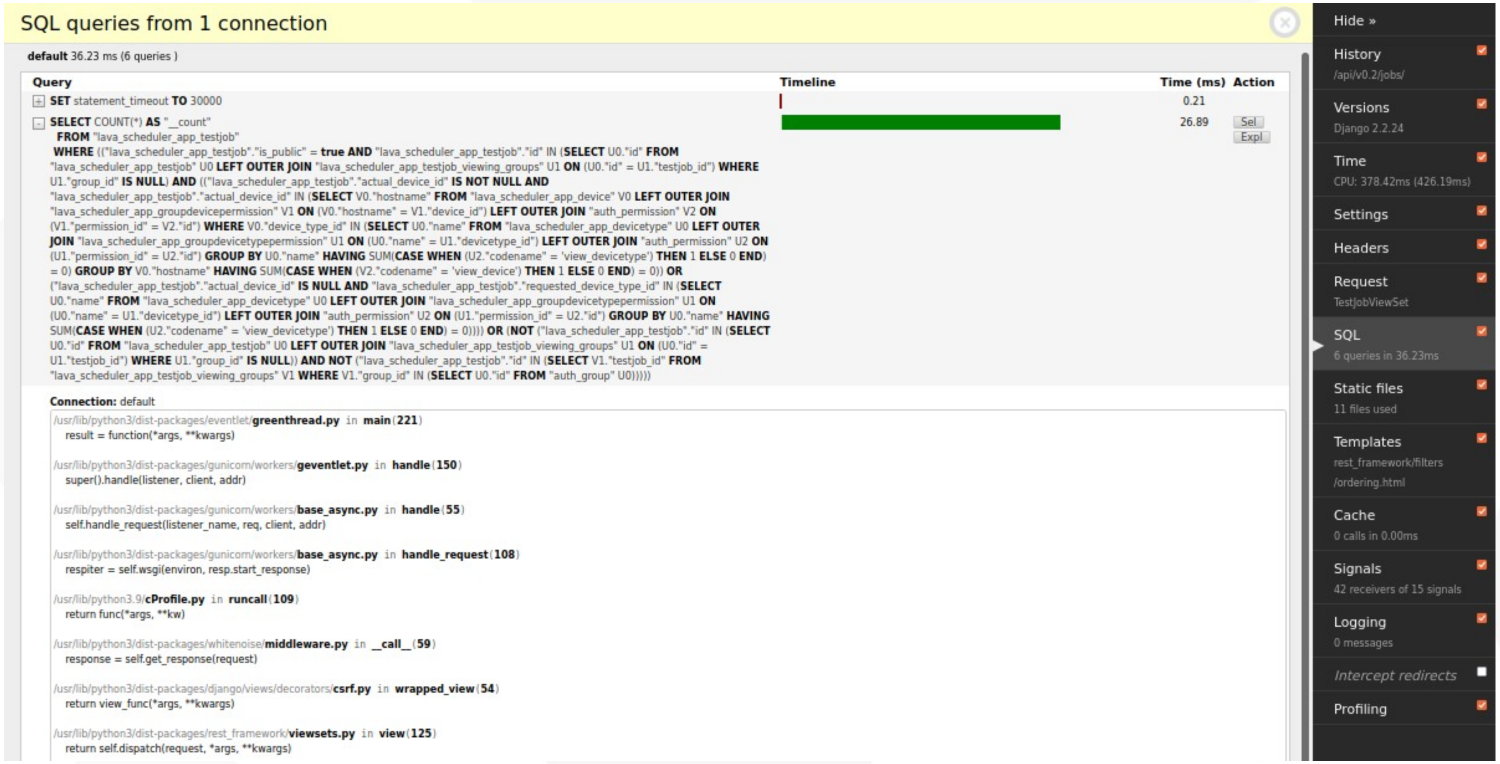


lava_server/settings/common.py

```diff
@@ -38,6 +38,7 @@ from lava_rest_app.versions import versions as REST_VERSIONS
 from lava_scheduler_app.settings import *
 
 
+DEBUG = True
 # List of people who get code error notifications
 # https://docs.djangoproject.com/en/1.11/ref/settings/#admins
 ADMINS = [("lava-server Administrator", "root@localhost")]
@@ -550,10 +551,10 @@ def update(values):
                 },
             },
             "loggers": {
-                "django": {
+                "django.db.backends": {
                     "handlers": ["logfile"],
                     # DEBUG outputs all SQL statements
-                    "level": "ERROR",
+                    "level": "DEBUG",
                     "propagate": True,
                 },
                 "django_auth_ldap": {
```

FOSDEM'24

Django Debug Toolbar



SQL queries from 1 connection

default 36.23 ms (6 queries)

Query	Timeline	Time (ms)	Action
SET statement_timeout **TO** 30000		0.21	
SELECT COUNT(*) **AS** "__count" **FROM** "lava_scheduler_app_testjob" **WHERE** (("lava_scheduler_app_testjob"."is_public" = **true AND** "lava_scheduler_app_testjob"."id" IN (**SELECT** U0."id" **FROM** "lava_scheduler_app_testjob" U0 **LEFT OUTER JOIN** "lava_scheduler_app_testjob_viewing_groups" U1 **ON** (U0."id" = U1."testjob_id") **WHERE** U1."group_id" **IS NULL**) **AND** (("lava_scheduler_app_testjob"."actual_device_id" **IS NOT NULL AND** "lava_scheduler_app_testjob"."actual_device_id" IN (**SELECT** V0."hostname" **FROM** "lava_scheduler_app_device" V0 **LEFT OUTER JOIN** "lava_scheduler_app_groupdevicepermission" V1 **ON** (V0."hostname" = V1."device_id") **LEFT OUTER JOIN** "auth_permission" V2 **ON** (V1."permission_id" = V2."id") **WHERE** V0."device_type_id" IN (**SELECT** U0."name" **FROM** "lava_scheduler_app_devicetype" U0 **LEFT OUTER JOIN** "lava_scheduler_app_groupdevicetypepermission" U1 **ON** (U0."name" = U1."devicetype_id") **LEFT OUTER JOIN** "auth_permission" U2 **ON** (U1."permission_id" = U2."id") **GROUP BY** U0."name" **HAVING** SUM(**CASE WHEN** (U2."codename" = 'view_devicetype') **THEN** 1 **ELSE** 0 **END**) = 0) **GROUP BY** V0."hostname" **HAVING** SUM(**CASE WHEN** (V2."codename" = 'view_device') **THEN** 1 **ELSE** 0 **END**) = 0)) **OR** ("lava_scheduler_app_testjob"."actual_device_id" **IS NULL AND** "lava_scheduler_app_testjob"."requested_device_type_id" IN (**SELECT** U0."name" **FROM** "lava_scheduler_app_devicetype" U0 **LEFT OUTER JOIN** "lava_scheduler_app_groupdevicetypepermission" U1 **ON** (U0."name" = U1."devicetype_id") **LEFT OUTER JOIN** "auth_permission" U2 **ON** (U1."permission_id" = U2."id") **GROUP BY** U0."name" **HAVING** SUM(**CASE WHEN** (U2."codename" = 'view_devicetype') **THEN** 1 **ELSE** 0 **END**) = 0)))) **OR** (**NOT** ("lava_scheduler_app_testjob"."id" IN (**SELECT** U0."id" **FROM** "lava_scheduler_app_testjob" U0 **LEFT OUTER JOIN** "lava_scheduler_app_testjob_viewing_groups" U1 **ON** (U0."id" = U1."testjob_id") **WHERE** U1."group_id" **IS NULL**)) **AND NOT** ("lava_scheduler_app_testjob"."id" IN (**SELECT** V1."testjob_id" **FROM** "lava_scheduler_app_testjob_viewing_groups" V1 **WHERE** V1."group_id" IN (**SELECT** U0."id" **FROM** "auth_group" U0)))))		26.89	Sel Expl

Connection: default

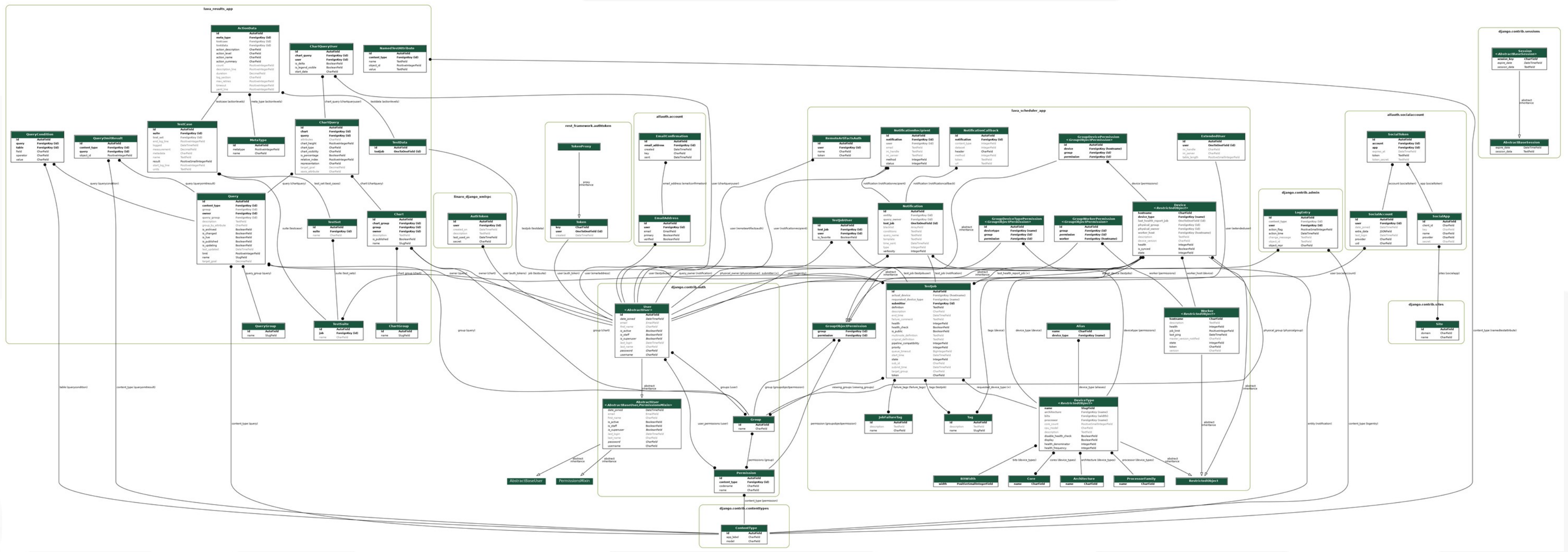
```
/usr/lib/python3/dist-packages/eventlet/greenthread.py in main(221)
  result = function(*args, **kwargs)

/usr/lib/python3/dist-packages/gunicorn/workers/geventlet.py in handle(150)
  super().handle(listener, client, addr)

/usr/lib/python3/dist-packages/gunicorn/workers/base_async.py in handle(55)
  self.handle_request(listener_name, req, client, addr)

/usr/lib/python3/dist-packages/gunicorn/workers/base_async.py in handle_request(108)
  respiter = self.wsgi(environ, resp.start_response)

/usr/lib/python3.9/cProfile.py in runcall(109)
  return func(*args, **kw)

/usr/lib/python3/dist-packages/whitenoise/middleware.py in __call__(59)
  response = self.get_response(request)

/usr/lib/python3/dist-packages/django/views/decorators/csrf.py in wrapped_view(54)
  return view_func(*args, **kwargs)

/usr/lib/python3/dist-packages/rest_framework/viewsets.py in view(125)
  return self.dispatch(request, *args, **kwargs)
```

Local instances

How to provide production workload?

- Initially often a clean slate
- Only virtual devices?
- Populating database with fixtures?

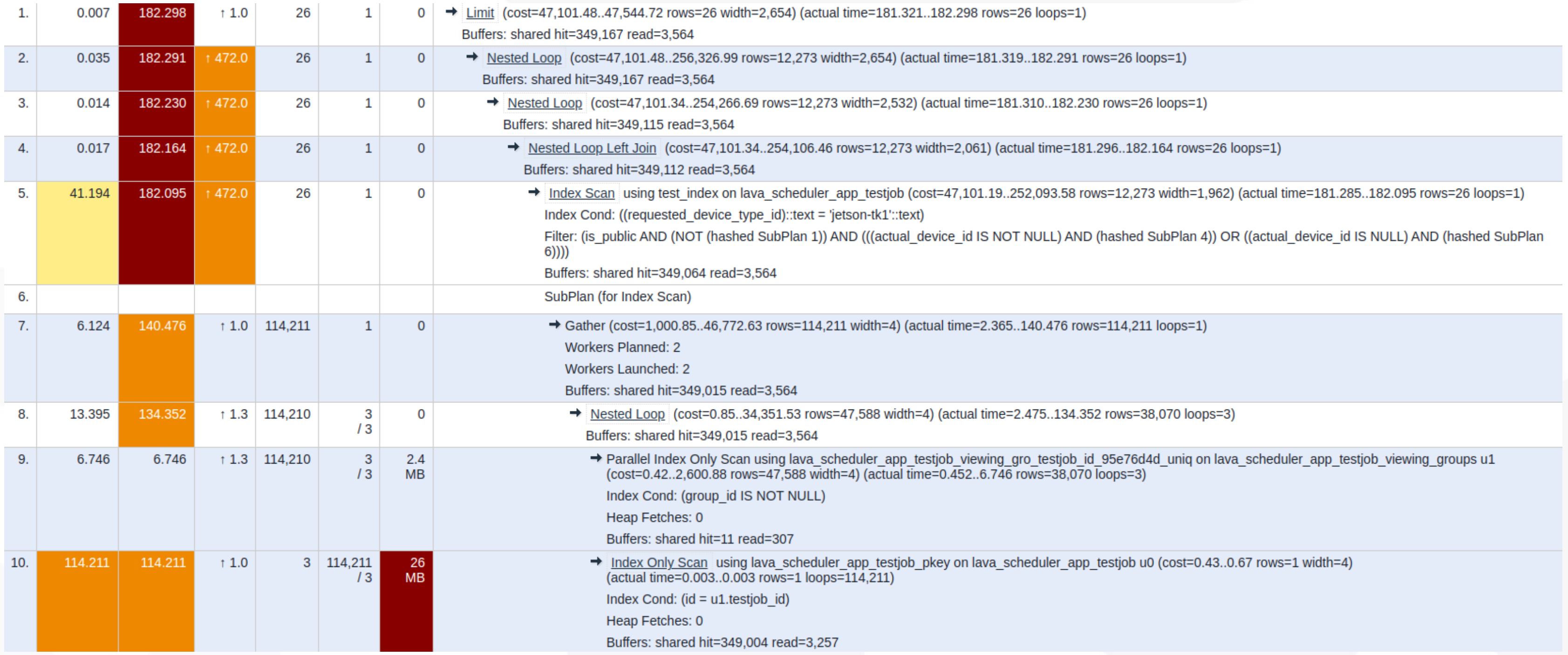
Model for lava-server



Two groups of people

```
case "$f" in
        *.sh)
                # https://github.com/docker-library/postgres/issues/450#issuecomment-393167936
                # https://github.com/docker-library/postgres/pull/452
                if [ -x "$f" ]; then
                        printf '%s: running %s\n' "$0" "$f"
                        "$f"
                else
                        printf '%s: sourcing %s\n' "$0" "$f"
                        . "$f"
                fi
                ;;
        *.sql)     printf '%s: running %s\n' "$0" "$f"; docker_process_sql -f "$f"; printf '\n' ;;
        *.sql.gz)  printf '%s: running %s\n' "$0" "$f"; gunzip -c "$f" | docker_process_sql; printf '\n' ;;
        *.sql.xz)  printf '%s: running %s\n' "$0" "$f"; xzcat "$f" | docker_process_sql; printf '\n' ;;
        *.sql.zst) printf '%s: running %s\n' "$0" "$f"; zstd -dc "$f" | docker_process_sql; printf '\n' ;;
        *)         printf '%s: ignoring %s\n' "$0" "$f" ;;
esac
```

More insights with pgAdmin



1.	0.007	182.298	↑ 1.0	26	1	0	→ Limit (cost=47,101.48..47,544.72 rows=26 width=2,654) (actual time=181.321..182.298 rows=26 loops=1) Buffers: shared hit=349,167 read=3,564
2.	0.035	182.291	↑ 472.0	26	1	0	→ Nested Loop (cost=47,101.48..256,326.99 rows=12,273 width=2,654) (actual time=181.319..182.291 rows=26 loops=1) Buffers: shared hit=349,167 read=3,564
3.	0.014	182.230	↑ 472.0	26	1	0	→ Nested Loop (cost=47,101.34..254,266.69 rows=12,273 width=2,532) (actual time=181.310..182.230 rows=26 loops=1) Buffers: shared hit=349,115 read=3,564
4.	0.017	182.164	↑ 472.0	26	1	0	→ Nested Loop Left Join (cost=47,101.34..254,106.46 rows=12,273 width=2,061) (actual time=181.296..182.164 rows=26 loops=1) Buffers: shared hit=349,112 read=3,564
5.	41.194	182.095	↑ 472.0	26	1	0	→ Index Scan using test_index on lava_scheduler_app_testjob (cost=47,101.19..252,093.58 rows=12,273 width=1,962) (actual time=181.285..182.095 rows=26 loops=1) Index Cond: ((requested_device_type_id)::text = 'jetson-tk1'::text) Filter: (is_public AND (NOT (hashed SubPlan 1)) AND (((actual_device_id IS NOT NULL) AND (hashed SubPlan 4)) OR ((actual_device_id IS NULL) AND (hashed SubPlan 6)))) Buffers: shared hit=349,064 read=3,564
6.							SubPlan (for Index Scan)
7.	6.124	140.476	↑ 1.0	114,211	1	0	→ Gather (cost=1,000.85..46,772.63 rows=114,211 width=4) (actual time=2.365..140.476 rows=114,211 loops=1) Workers Planned: 2 Workers Launched: 2 Buffers: shared hit=349,015 read=3,564
8.	13.395	134.352	↑ 1.3	114,210	3 / 3	0	→ Nested Loop (cost=0.85..34,351.53 rows=47,588 width=4) (actual time=2.475..134.352 rows=38,070 loops=3) Buffers: shared hit=349,015 read=3,564
9.	6.746	6.746	↑ 1.3	114,210	3 / 3	2.4 MB	→ Parallel Index Only Scan using lava_scheduler_app_testjob_viewing_gro_testjob_id_95e76d4d_uniq on lava_scheduler_app_testjob_viewing_groups u1 (cost=0.42..2,600.88 rows=47,588 width=4) (actual time=0.452..6.746 rows=38,070 loops=3) Index Cond: (group_id IS NOT NULL) Heap Fetches: 0 Buffers: shared hit=11 read=307
10.	114.211	114.211	↑ 1.0	3	114,211 / 3	26 MB	→ Index Only Scan using lava_scheduler_app_testjob_pkey on lava_scheduler_app_testjob u0 (cost=0.43..0.67 rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=114,211) Index Cond: (id = u1.testjob_id) Heap Fetches: 0 Buffers: shared hit=349,004 read=3,257

https://explain.depesz.com

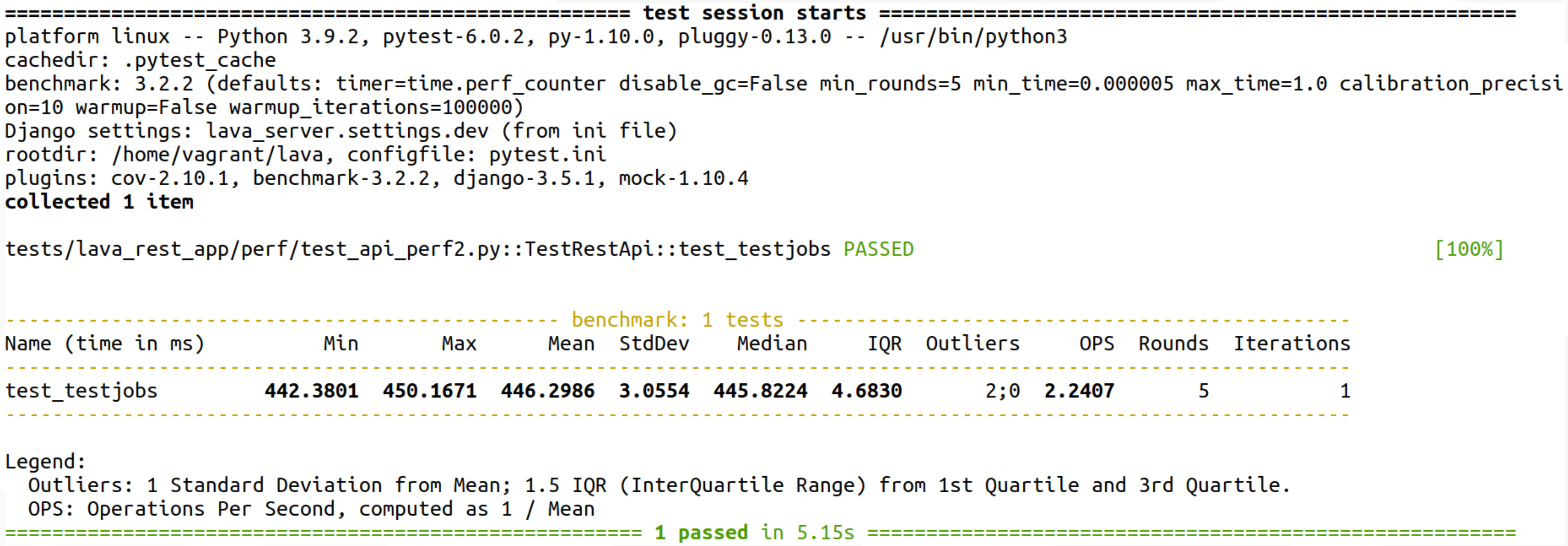
Automated tracking

Add your first benchmark to CI

How to fit benchmarks in existing tests?

- Cache warmup
- Calibration
- Result comparison
- Compatible framework

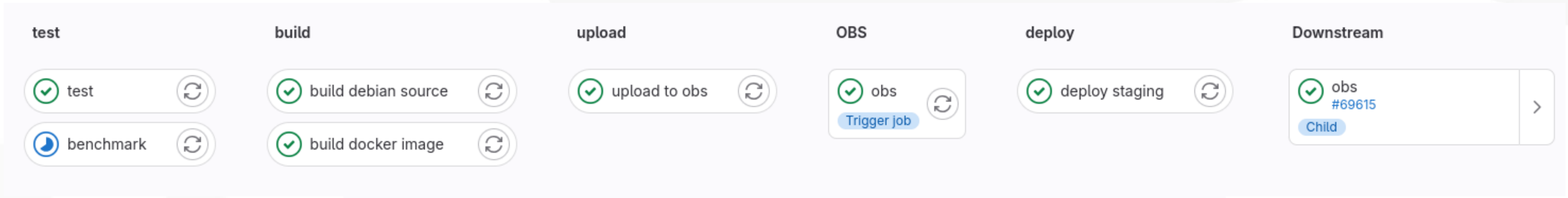
LAVA-compatible fixture



```
=================================================== test session starts ===================================================
platform linux -- Python 3.9.2, pytest-6.0.2, py-1.10.0, pluggy-0.13.0 -- /usr/bin/python3
cachedir: .pytest_cache
benchmark: 3.2.2 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005 max_time=1.0 calibration_precisi
on=10 warmup=False warmup_iterations=100000)
Django settings: lava_server.settings.dev (from ini file)
rootdir: /home/vagrant/lava, configfile: pytest.ini
plugins: cov-2.10.1, benchmark-3.2.2, django-3.5.1, mock-1.10.4
collected 1 item

tests/lava_rest_app/perf/test_api_perf2.py::TestRestApi::test_testjobs PASSED                                       [100%]


---------------------------------------------- benchmark: 1 tests ------------------------------------------------
Name (time in ms)          Min       Max      Mean  StdDev    Median     IQR  Outliers     OPS  Rounds  Iterations
--------------------------------------------------------------------------------------------------------------------
test_testjobs         442.3801  450.1671  446.2986  3.0554  445.8224  4.6830       2;0  2.2407       5           1
--------------------------------------------------------------------------------------------------------------------

Legend:
  Outliers: 1 Standard Deviation from Mean; 1.5 IQR (InterQuartile Range) from 1st Quartile and 3rd Quartile.
  OPS: Operations Per Second, computed as 1 / Mean
==================================================== 1 passed in 5.15s ====================================================
```

https://pytest-benchmark.readthedocs.io

Plug into GitLab CI pipeline


test
- ✓ test
- benchmark

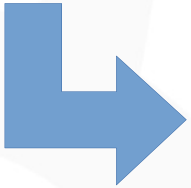
build
- ✓ build debian source
- ✓ build docker image

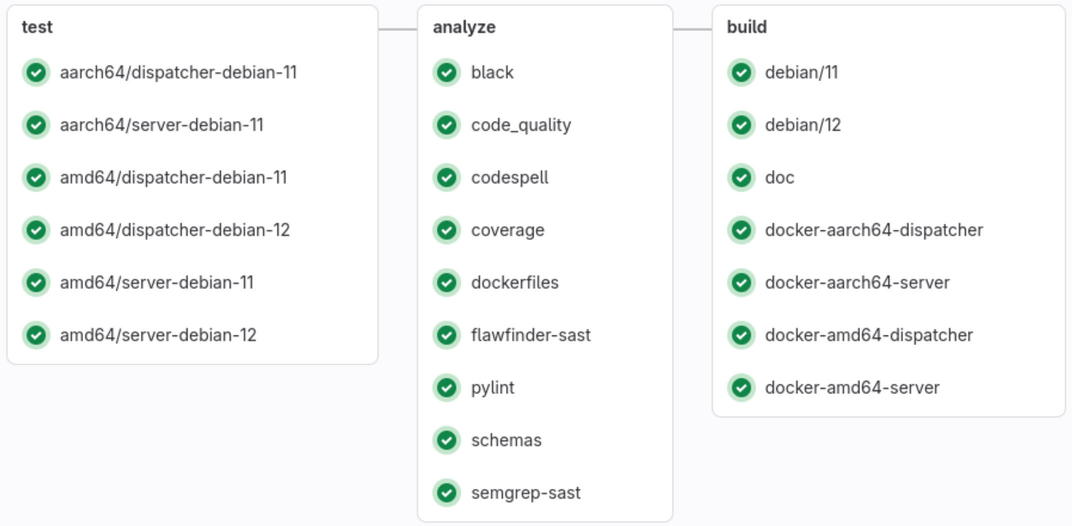
upload
- ✓ upload to obs

OBS
- ✓ obs
  Trigger job

deploy
- ✓ deploy staging

Downstream
- ✓ obs
  #69615
  Child




test
- ✓ aarch64/dispatcher-debian-11
- ✓ aarch64/server-debian-11
- ✓ amd64/dispatcher-debian-11
- ✓ amd64/dispatcher-debian-12
- ✓ amd64/server-debian-11
- ✓ amd64/server-debian-12

analyze
- ✓ black
- ✓ code_quality
- ✓ codespell
- ✓ coverage
- ✓ dockerfiles
- ✓ flawfinder-sast
- ✓ pylint
- ✓ schemas
- ✓ semgrep-sast

build
- ✓ debian/11
- ✓ debian/12
- ✓ doc
- ✓ docker-aarch64-dispatcher
- ✓ docker-aarch64-server
- ✓ docker-amd64-dispatcher
- ✓ docker-amd64-server

Dedicated GitLab CI runner

https://docs.gitlab.com/runner/install/

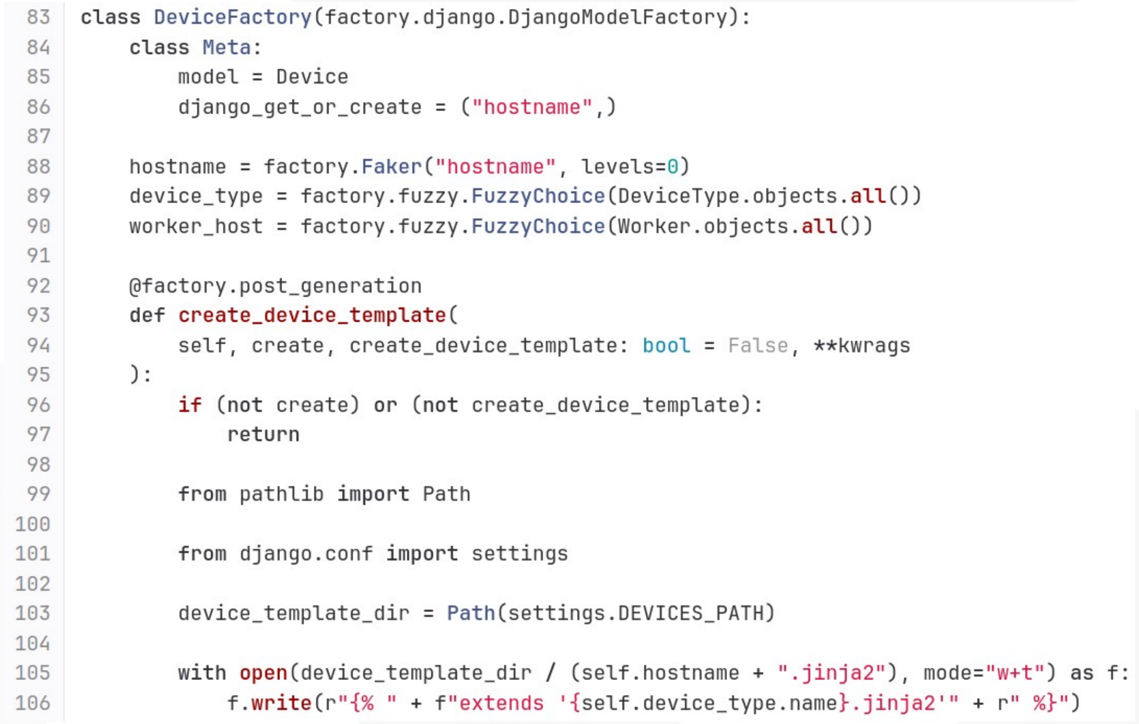
Cache CI data resources

What data could be used here?

- Quicker feedback loop
- Mechanism already in place: https://gitlab.com/lava/ci-images
- Copy interactive approach almost 1:1

Data generation

Dummy database generator



```python
class DeviceFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = Device
        django_get_or_create = ("hostname",)

    hostname = factory.Faker("hostname", levels=0)
    device_type = factory.fuzzy.FuzzyChoice(DeviceType.objects.all())
    worker_host = factory.fuzzy.FuzzyChoice(Worker.objects.all())

    @factory.post_generation
    def create_device_template(
        self, create, create_device_template: bool = False, **kwrags
    ):
        if (not create) or (not create_device_template):
            return

        from pathlib import Path

        from django.conf import settings

        device_template_dir = Path(settings.DEVICES_PATH)

        with open(device_template_dir / (self.hostname + ".jinja2"), mode="w+t") as f:
            f.write(r"{% " + f"extends '{self.device_type.name}.jinja2'" + r" %}")
```

https://gitlab.collabora.com/lava/lava/-/blob/collabora/production/lava_db_generator/

Bonus: data retention

Is all this data really necessary?

- Should LAVA archive all the jobs?
- Can it be delegated?
- Retention mechanism available upstream
- Enabled in Helm chart

Summary

Final thoughts

- Process, not a one-time action
- Frequent revisiting and adjustments
- Small changes can bring huge boosts

Thank you!

Images used

- https://www.freeimages.com/photo/burning-computer-1508147 by dknudsen
- https://pulsgdanska.pl/artykul/rzut-dyskiem-twardym/1351382
- https://docs.lavasoftware.org/lava/#architecture
- https://wiki.postgresql.org/wiki/File:PostgreSQL_logo.3colors.120x120.png
- https://www.djangoproject.com/m/img/logos/django-logo-negative.png
- https://www.servethehome.com/introducing-project-tinyminimicro-home-lab-revolution/