# I Have a Web Framework. Now What?


## Pradeep Gowda


**IndyPy Web Conf**
**Aug 23, 2019**

# Who am i?

- Programming python since ~1998
- Member of IndyPy since 2008
- Zope/Plone-> Django -> Flask -> Pyramid -> Scala/Java | **Django**
- Recently developed a web service that is capable of collecting millions of events a day
- Built on Django, Ansible, Prometheus, Grafana ...

# Themes of this talk

- Growing your app beyond "Hey, I have a web app!"
- Confidence
- Reduce toil
- Observability

# 12 factor app https://12factor.net

I. **Codebase**
II. **Dependencies**
III. **Config**
IV.
V. **Build, release, run**
VI.
VII.
VIII.
IX.

## The Twelve Factors

**I. Codebase**
One codebase tracked in revision control, many deploys

**II. Dependencies**
Explicitly declare and isolate dependencies

**III. Config**
Store config in the environment

**IV. Backing services**
Treat backing services as attached resources

**V. Build, release, run**
Strictly separate build and run stages

**VI. Processes**
Execute the app as one or more stateless processes

**VII. Port binding**
Export services via port binding

**VIII. Concurrency**
Scale out via the process model

**IX. Disposability**
Maximize robustness with fast startup and graceful shutdown

**X. Dev/prod parity**
Keep development, staging, and production as similar as possible

**XI. Logs**
Treat logs as event streams

**XII. Admin processes**
Run admin/management tasks as one-off processes

# Devops

> *If a human operator needs to touch your system during normal operations, you have a bug. The definition of normal changes as your systems grow.*
>
> Carla Geisser, Google SRE

# What is Toil

Work tied to running a production service that tends to be:

- Manual
- Repetitive
- Automatable and not requiring human judgement
- Interrupt-driven and reactive
- Of no enduring value

# Further reading on "Toil"

- Eliminating Toil
- Invent more, toil less [USENIX 2016 paper]

# Confidence

being able to say:

- something is done
- something works
- works [not] only on my machine
- works for a team, not just an individual or a subset of team(s)

# What gives confidence?

- Repeatable
- Reproducable
- Changing with ease
- Handling change in external circumstances with ease
- Incorporating learning into future scenarios

# Reproducible

- reproducible, not Reproducible (see Nix etc.,)
  - pinning dependencies
  - operating systems
  - environments

# Pinning dependencies

- `requirements.txt`
- `Pipenv`
- `poetry`

# pipenv

```
$ cat Pipfile
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
django = "*"
redis = "*"
django-role-permissions = "*"
django-extensions = "*"
bpython = "*"
coverage = "*"
django-debug-toolbar = "*"
python-decouple = "*"
django-rest-swagger = "*"
django-docs = "*"
django-redis = "*"
hiredis = "*"
"psycopg2-binary" = "*"
python-language-server = {version = "*",extras = ["all"]}
django-prometheus = "*"

[dev-packages]
pyre-check = "*"
pylint = "*"
yapf = "*"

[requires]
python_version = "3.7"
```

# Packaging

- deployment ready.
- use CI and build systems.

# Versioning

- the same build should travel through various environments.
- dev/qa/staging/prod

```
# myapp/__version__.py
VERSION = (1, 1, 19)

__version__ = '.'.join(map(str, VERSION))
```

- use your build system to increment the version

# Testing

- Confidence
- Testing as documentation
- Especially true for APIs

# Testing

```
from django.test import Client
from myapi.tests import MyTestCase
import json

class TestSensor(MyTestCase):
    def test_get_sensor_info_as_a_sensor(self):
        """as a sensor, fetch info about itself"""
        # ... snip ...
        j = json.loads(response.content)
        self.assertEqual(response.status_code, 201)
        sensor_id = j['sensorId']
        sensor_token = j['token']
        header = "Bearer %s" % (sensor_token, )
        response = c.get(
            '/api/v1/sensor',
            {
                'sensorId': sensor_id,
            },
            HTTP_AUTHORIZATION=header,
        )
        j = json.loads(response.content)
        self.assertEqual(response.status_code, 200)
```

# Environments

*One codebase, many deployments*

- tie back to "reproducibility"
- dev -> qa -> staging -> prod
- drive application behaviour through configuration, not code change

# Configuration

- Use python-decouple

```
# settings.ini
[settings]
DEBUG=True
TEMPLATE_DEBUG=%(DEBUG)s
SECRET_KEY=ARANDOMSECRETKEY
```

```
# coding: utf-8
from decouple import config

DEBUG = config('DEBUG', default=False, cast=bool)
```

# Configuration

- use a deployment tool to install environment specific configuration

# Deployment

**Use** an automation tool

- **ansible**, puppet, chef, kubernetes
- repeatable
- reproducible
- self documenting
- start automation along with code

# Contextual memory

context-dependent memory is the improved recall of specific episodes or information when the context present at encoding and retrieval are the same. One particularly common example of context-dependence at work occurs when an individual has lost an item (e.g. lost car keys) in an unknown location. Typically, people try to systematically **"retrace their steps"** to determine all of the possible places where the item might be located.

# Make

## Use `Makefile` as your contextual memory helper

```
# Makefile from a real project

test:
    python manage.py test --settings=myproject.test_settings

package:
    python3 setup.py sdist

cppackage: package
    cp dist/myproject*.tar.gz ../myproject-deployments/dist/

.PHONY:
clean:
    rm -rf dist
```

# Why Make

*Makefiles are machine-readable documentation that make your workflow reproducible.*

– Mike Bostok

See Why I use Make https://bost.ocks.org/mike/make/

# Observability

Gain visibility into the behavior of applications and infrastructure

- the multi-dimensional, everchanging aspects of production environment
- unpredictable inputs
- dependence on upstream and downstream dependencies

Ref: Logs and Metrics

# Logging

- log is an immutable record of discrete events that happened over time.
- what to log?
- DEBUG
- INFO
- WARN
- ERROR

# Logging

- avoid print statements
- convert `prints` to `logging.DEBUG`
- catch exceptions in logs with error information

```
try:
with io.open(os.path.join(here, 'README.md'),
encoding='utf-8') as f:
    long_description = '\n' + f.read()
```

# Logging

- Request ID
- Identify client requests within non-sequential logs
- Adds a unique ID to each request
- Fronting web server / load-balancers might also provide this
- request-id to add unique id to WSGI app

# Logging

- Plain-text
- Structured
- Binary

# structured logging

- you can capture just about any data
- high dimensionality
- Can do things like
  - exploratory analysis
  - auditing
  - analytics (user engagement)

# Structured logging


```
{
 "method":"GET",
 "path":"/users",
 "format":"html",
 "controller":"users",
 "action":"index",
 "status":200,
 "duration":189.35,
 "view":186.35,
 "db":0.92,
 "@timestamp":"2015-12-11T13:35:47.062+00:00",
 "@version":"1",
 "message":"[200] GET /users (users#index)",
 "severity":"INFO",
 "host":"app1-web1",
 "type":"apps"
}
```

# Log analysis tools

- Splunk
- ELK stack

# Structured Log library

structlog library for python

```
import logging
import uuid
import structlog

logger = structlog.get_logger()
app = flask.Flask(__name__)


@app.route("/login", methods=["POST", "GET"])
def some_route():
    log = logger.new(request_id=str(uuid.uuid4()))
    # do something
    # ...
    log.info("user logged in", user="test-user")
    # gives you:
    # event='user logged in' request_id='ffcdc44f-b952-4b5f-95e6-0f1f3a9ee5fd' user='test-user'
```

# JSON structured logging

```
>>> import datetime, logging, sys
>>> from structlog import wrap_logger
>>> from structlog.processors import JSONRenderer
>>> from structlog.stdlib import filter_by_level
>>> logging.basicConfig(stream=sys.stdout, format="%(message)s")
>>> def add_timestamp(_, __, event_dict):
...     event_dict["timestamp"] = datetime.datetime.utcnow()
...     return event_dict
>>> def censor_password(_, __, event_dict):
...     pw = event_dict.get("password")
...     if pw:
...         event_dict["password"] = "*CENSORED*"
...     return event_dict
>>> log = wrap_logger(
...     logging.getLogger(__name__),
...     processors=[
...         filter_by_level,
...         add_timestamp,
...         censor_password,
...         JSONRenderer(indent=1, sort_keys=True)
...     ]
... )
>>> log.info("something.filtered")
>>> log.warning("something.not_filtered", password="secret")
{
 "event": "something.not_filtered",
 "password": "*CENSORED*",
 "timestamp": "datetime.datetime(..., ..., ..., ..., ...)"
}
```

# Metrics

*a set of numbers that give information about a particular process or activity.*

# Metrics

- measure of success and failure
- rate of growth
- patterns of behavior

# prometheus

- time series are represented using key/value pairs "labels"
- a metric => name, label

```
<metric name>{<label name>=<label value>, ...}
```

```
api_http_requests_total{method="POST", handler="/messages"} 3582
```

# Prometheus

- increase in traffic does not mean increase in disk use, complexity
- disk use increases only when you add **new** metrics (and/or more hosts)
- Push vs Pull

# Prometheus Pull

Part of your application

```
from django_prometheus import exports

urlpatterns = [
    path("api/v1/", include("sensorapi.urls")),
    url(r'^metrics$',
        exports.ExportToDjangoView,
        name='prometheus-django-metrics'),
]
```

# Prometheus Metrics

Counter:

```
from prometheus_client import Counter

logger = logging.getLogger(__name__)
state_transition_counter = Counter('state_transition_counter',
                                   'Number of State Transitions')
```

# Prometheus Push

- push to "gateway"
- used for cronjobs and "one-off" processes

# Django prometheus

```
# HELP django_http_requests_total_by_view_transport_method_total Count of requests by view, transport, method.
# TYPE django_http_requests_total_by_view_transport_method_total counter
django_http_requests_total_by_view_transport_method_total{method="GET",transport="http",view="prometheus-django-metrics"} 358280.0
django_http_requests_total_by_view_transport_method_total{method="GET",transport="http",view="homepage"} 2.410289e+06
django_http_requests_total_by_view_transport_method_total{method="HEAD",transport="http",view="sensor"} 423094.0
```

# What metrics to collect?

| **RED method** | |
|---|---|
| How *busy* is my service? | **R**equest rate |
| Are there any *errors* in my service? | **E**rror rate |
| What is the *latency* in my service? | **D**uration of requests |

- use these for 95% for monitoring and alerting. Combine with Utilisation, Saturation, Error metrics (Brendan Gregg) plus other metrics for fault finding -- @tom_wilkie

# What metrics to collect?

- resource: all physical server functional components (CPUs, disks, and software resourcess)
- **U**tilization: the average time that the resource was busy servicing work ["one disk is running at 90% utilization"]
- **S**aturation: the degree to which the resource has extra work which it can't service, often queued ["the CPUs have an average run queue length of four"]

# Logs vs Metrics

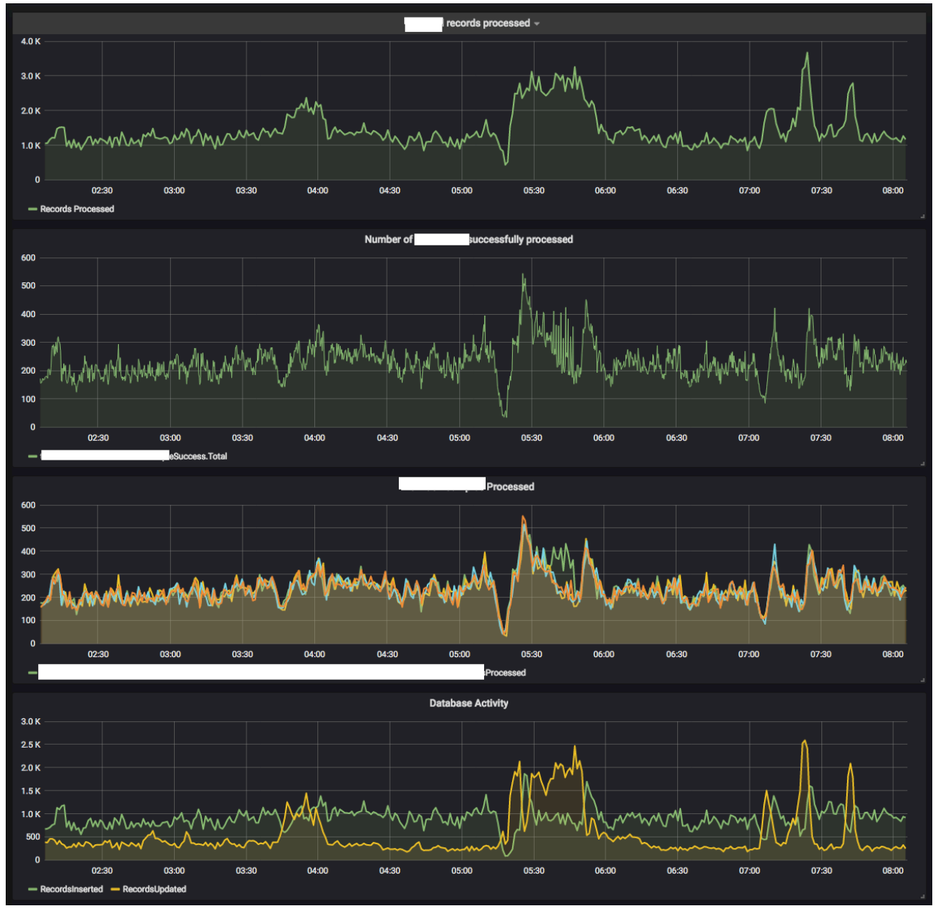
- events -- aggregate of events
- high dimensionality -- low dimensionality
- unstructured -- structured
- analysis -- dashboards & alerting
- vary in volume -- fixed volume
- high volume -- low volume

-- \@pmech42

# grafana dashboard

- grafana is a software for time series analytics
- and dashboards

# grafana


records processed
4.0 K
3.0 K
2.0 K
1.0 K
0
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
Records Processed
Number of successfully processed
600
500
400
300
200
100
0
Success.Total
Processed
Processed
Database Activity
3.0 K
2.5 K
2.0 K
1.5 K
1.0 K
500
RecordsInserted
RecordsUpdated

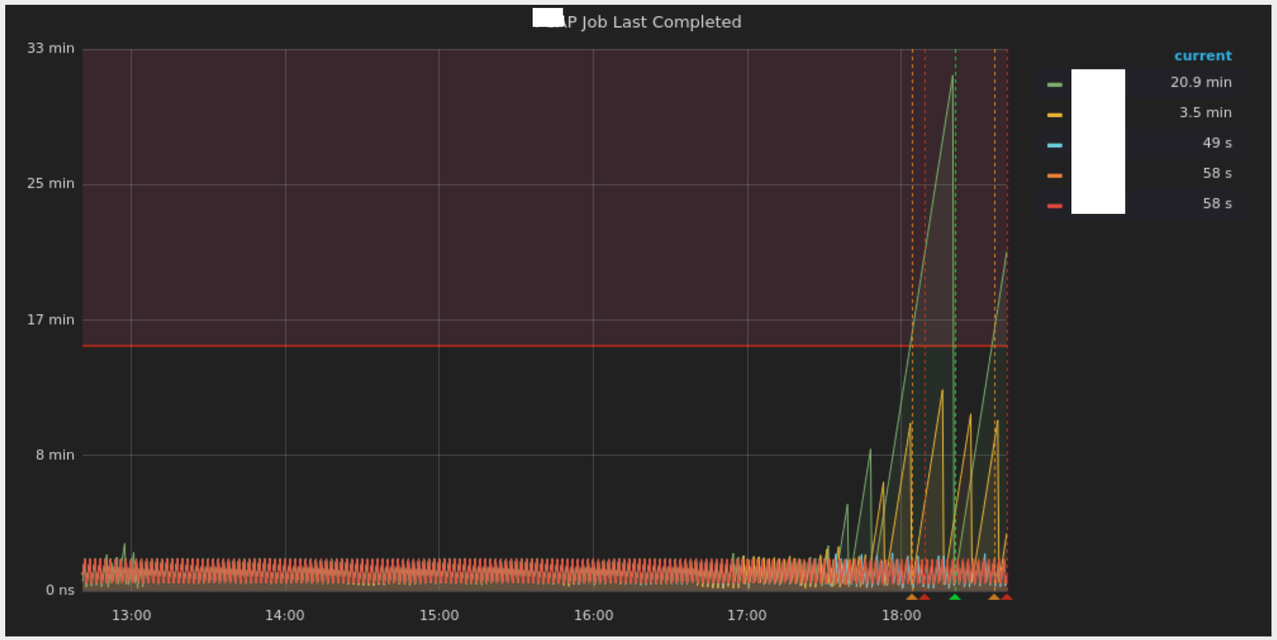
# Alerting

- nagios alerting
- grafana alerting

# Grafana alerting

*[Alerting] M[redacted]P Job Down*

| Metric name | Value |
| --- | --- |
| W[redacted] | 1253.535 |



# Themes of this talk

- Growing your app beyond "Hey, I have a web app!"
- Confidence
- Reduce toil
- Observability

# Questions/Comments?

- pradeep@btbytes.com
- @btbytes on twitter