Deploying a TOM: Making your system available to others

Current deployments

- Who deploys your current applications?
  - Scientists
  - IT
- What tools are used to deploy your current applications?
  - Package manager (e.g. DNF, apt-get, YUM)
  - **Only** containers
  - Container orchestration

Objective

- Allow others to access your TOM
- Don’t let your data get stolen
- Don't leave you whole institution open to get hacked through your TOM

Be Safe!

Developing your TOM

- **Always** use version control
  - Allows other to help contribute to your TOM
  - Makes it easier to identify changes that introduce bugs and unexpected behavior
  - Use git (https://git-scm.com/) with a web service to make development easier such as:
    - GitHub (https://github.com/)
    - GitLab (https://about.gitlab.com/)
- Keep secrets *secret*
  - Don't store your passwords or access keys in git!!
  - Instead, use environment variables or a local settings file

Using Environment Variables in your settings.py

settings.py

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ['MY_TOM_SECRET_KEY']
```

One way to load values into an environment variable, is with your ~/.bashrc

```
export MY_TOM_SECRET_KEY='03s19c@m#huycxb@pzs8c5z%e=^*b!zupt+ds&amp;ml+cj7xif%6f'
```

IMPORTANT!

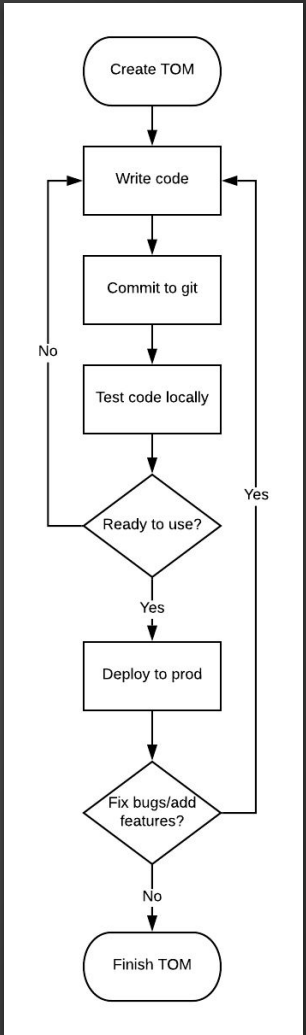
Since the settings.py file gets committed to git, it will be available to the public when pushed to GitHub. **Never** commit keys or tokens to git.

Environments

- Development
  - You will be the only one to use this
  - Rapidly changing
  - Insecure
  - Low overhead because only **one** user needs to be served
- Production
  - Intended to be used by more than just you
  - Changes less quickly than a development environment
  - Secure
  - Scales to serve multiple users

Software Development Lifecycle (SDLC)

- Just a fancy name for developing and deploying software
- This is roughly how your TOM development workflow should look like



Django Deployment Checklist

- Checklist can be found here
- Can be setup to use git hooks: https://githooks.com

Deployment checklist

The Internet is a hostile environment. Before deploying your Django project, you should take some time to review your settings, with security, performance, and operations in mind.

Django includes many security features. Some are built-in and always enabled. Others are optional because they aren't always appropriate, or because they're inconvenient for development. For example, forcing HTTPS may not be suitable for all websites, and it's impractical for local development.

Performance optimizations are another category of trade-offs with convenience. For instance, caching is useful in production, less so for local development. Error reporting needs are also widely different.

The following checklist includes settings that:

- must be set properly for Django to provide the expected level of security;
- are expected to be different in each environment;
- enable optional security features;
- enable performance optimizations;
- provide error reporting.

Many of these settings are sensitive and should be treated as confidential. If you're releasing the source code for your project, a common practice is to publish suitable settings for development, and to use a private settings module for production.

Run `manage.py check --deploy`

Some of the checks described below can be automated using the `check --deploy` option. Be sure to run it against your production settings file as described in the option's documentation.

Contents

How do users access the data in your TOM?

User

The Internet

Web Server

Some considerations

- How does your user get to your host?
  - IP address; difficult to remember
  - Setup DNS (e.g. google.com instead of 172.217.15.78)
- How do you protect your machine from the internet?
  - Reverse proxy
  - Encrypt network traffic with TLS over HTTP (i.e. HTTPS) (https://letsencrypt.org/)
- How do you make a TOM reproducible?
  - Specify dependency versions in requirements.txt (a.k.a "pinning" dependencies)

Necessary Components: The Software Stack

- Users accessing your TOM
- Improve security and performance
- Access your TOM over the network
- Your TOM

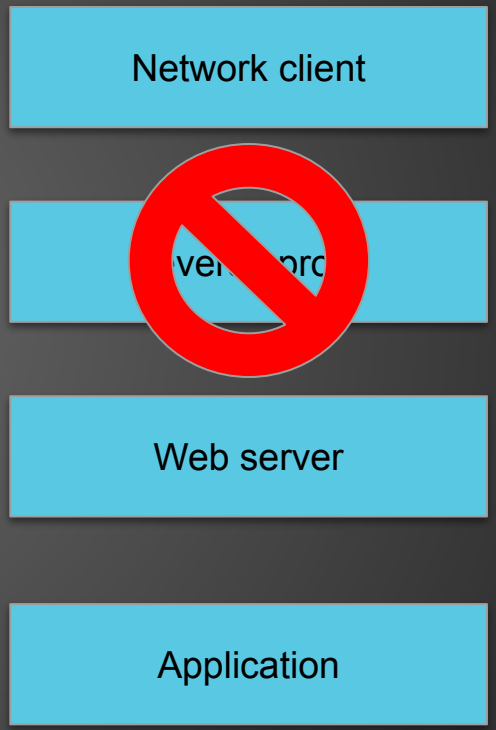
Network client

Reverse proxy

Web server

Application

A Warning: Do not develop in production!



- Limited by design, i.e. lower overhead for single user
- Insecure
- Not scalable
- **Cannot** be used in production

A common production setup:

Application	Web server	Reverse proxy	Network client

- Only limited by hardware
- Scalable
- Secure

Deployment

We'll cover 4 different deployment methods from **most** common to **least** common:

1. Platform as a service - **most common**
2. Container - **common**
3. Server - **uncommon**
4. Container orchestration - **most uncommon**

Deployment option 1

Deployment: Platform as a service (PaaS)

Heroku is a popular PaaS https://www.heroku.com/ (requires a GitHub account).

Heroku TOM example

Pros

- Don't have to manage a machine
- Mostly batteries included
- No domain name registration

Cons

- Free tier is very limited

Deployment option 2

Deployment: Container

Docker is the most popular container format: https://www.docker.com/

Pros

- Isolated environment
- Secure parameters can be defined as environment variables
- Platform independent
- Reproducible

Cons

- May not be initially intuitive

Deployment option 3

Deployment: Server

```
sudo apt-get -y install nginx
pip install gunicorn
<edit /etc/nginx/nginx.conf>
Add the following:
location / { /path/to/tom/static }
sudo systemctl start nginx
gunicorn --workers=2 mytom:myapp
```

Pros

- Familiar concepts

Cons

- Difficult to set up and reproduce
- Not very portable

Deployment: Containers

Example can be found at:
https://github.com/TOMToolkit/dockertom

Dockerfile

```
FROM python:3.7

WORKDIR /tom

COPY requirements.txt .

RUN pip install \
    --no-cache \
    --disable-pip-version-check \
    --requirement requirements.txt

COPY . .

CMD [ \
    "gunicorn", \
    "--bind=0.0.0.0:8080", \
    "--worker-class=gevent", \
    "--workers=4", \
    "--timeout=300", \
    "--access-logfile=-", \
    "--error-logfile=-", \
    "dockertom.wsgi:application" \
    ]
```

Deployment option 4

Deployment: Container orchestration

Should only be used when many applications are being managed or computational throughput cannot be achieved with a single machine.

Kubernetes is the most popular platform: https://kubernetes.io/

Pros

- Manage container interaction easier
- Horizontally scalable (i.e. not limited by the performance of a single machine)

Cons

- **Very** steep learning curve
- Requires a huge amount of infrastructure setup and management

docker-compose.yml

Running multiple containers together probably locally

```yaml
version: '2'
services:
  tom-nginx:
    image: nginx:1.17.4
    container_name: tom-nginx
    network_mode: "tom"
    volumes:
    - tomvolume:/tom
    command: ["nginx", "-g", "daemon off;", "location / { /tom/static }"]
  tom:
    image: tomtoolkit/tom
    network_mode: "tom"
    container_name: tom
    mem_limit: '1g'
    volumes:
      - tomvolume:/tom
    command: ["gunicorn", "--workers=2", "mytom:myapp"]

volumes:
  tomvolume:
    driver: local
```

Kubernetes (k8s)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: tom
  namespace: prod
  labels:
    app.kubernetes.io/name: tom
spec:
  # Create some empty directories to be mounted within the Pod
  volumes:
    - name: tom-data
      emptyDir:
        sizeLimit: 10Gi

  containers:
    - name: tom-nginx
      image: nginx:1.17.4
      imagePullPolicy: IfNotPresent
      resources:
        requests:
          cpu: 0.1
          memory: 256M
        limits:
          cpu: 1
          memory: 1Gi
      command:
        - "nginx"
        - "-g"
        - "daemon"
        - "off;"
        - "location"
        -  "/"
        - "{"
        - "/tom/static"
        - "}"
    - name: tom
      image: tomtoolkit/tom
      imagePullPolicy: IfNotPresent
      resources:
        requests:
          cpu: 0.1
          memory: 512M
        limits:
          cpu: 1
          memory: 1Gi
      command:
        - "gunicorn"
        - "--workers=2"
        - "mytom:myapp"
```