
>>> network .toCode()
NTC Cookie-Cutter

Overview

>>> Governance Model

Internally, the cookie cutter is managed by the dev-standards working group

- Found on Slack @ #dev-standards
- Found on Github @ https://github.com/networktocode-llc/cookiecutter-ntc
- Foundation of standards that can be re-used across multiple clients and projects
- Encourage everyone to participate whether it's issues, PRs, or conversations around decisions
- Describes our standards that should evolve
- Cookie cutter project types
  - Generic Python Project
  - Ansible Project, Collection
  - Nautobot Deployment, Plugins (Generic & ChatOps)

>>> Cookiecutter Template

Features Provided

- Linting, docstrings & code style
- Packaging & Dependency Management
- Docker Container Strategy
- CLI Application
- Doc building
- Testing
- Execution of builds, testing, etc.

>>> Tool Choices

- Cookiecutter – Deliver and re-use the project templates
- Docker & Docker Compose - Container and containers management
- Poetry – dependency management and package publishing
- Invoke – Replacement for Makefiles, used by developers
- Click – Python CLI application, consumed by users
- Mkdocs (*EM-K-Docs*) - docs hosting standard
- Linters
  - Python Black - Used for formatting
  - Bandit - Security enforcement
  - Pylint - Style enforcement
  - Flake8 - Error checking and style enforcement
  - Yamllint - Linter for yaml files
  - Pydocstyle - docstring enforcement
  - Mypy - Static typing
  - Pytest - Python testing framework

Poetry

>>> Poetry.. More Than Dependency Management

- Defines both application and development dependencies
- Defines application meta data for publishing a Python package
- Defines CLI application for package
- Virtual environment management
- Built-in security with poetry.lock and package hashes
- Uses pyproject.toml as configuration file

>>> pyproject.toml

- Replaces setup.py, setup.cfg, and requirements.txt
- PEP 518 - https://www.python.org/dev/peps/pep-0518/
- Holds other tool's configurations
- Poetry init command can be used to build initial pyproject.toml

>>> pyproject.toml – Cont'd

```
[tool.poetry]
name = "cookiecutter-ntc-python"
version = "1.0.0"
description = "NTC Development Standards"
authors = ["Network to Code, LLC <info@networktocode.com>"]
```

Package meta-data

```
[tool.poetry.dependencies]
python = "^3.6"
```

Dependencies directory related to the Python package

poetry install --no-dev

```
[tool.poetry.dev-dependencies]
pytest = "^5.4.1"
```

Development depedencies

poetry install

```
[tool.poetry.scripts]
exmple = 'example.cli:main'
```

Provides path to Python CLI for package

```
[build-system]
requires = ["poetry>=0.12"]
build-backend = "poetry.masonry.api"
```

```
[tool.pytest.ini_options]
testpaths = [
    "tests"
]
```

Other Python package configuration options that can be included in the pyproject.toml

>>> Tool.poetry.scripts

- Method to provide scripts or CLI scripts to be executable by defined name
- Can be executed by the name if poetry install is used or can be executed with poetry run example ${args}


Invoke

>>> Invoke - The Python Makefile

- Requires Invoke Python package installed
- Defined via tasks.py file
- Functions with the @task decorator to define each task (target)
- Provides CLI context help for each task using defined docstrings

>>> Invoke – Example task

```
@task
def build(context, nocache=False, forcerm=False, hide=False):
       """Build a Docker image.

       Args:
           context (obj): Used to run specific commands
           nocache (bool): Do not use cache when building the image
           forcerm (bool): Always remove intermediate containers
           hide (bool): Hide output of Docker image build
       """
       print(f"Building image {IMAGE_NAME}:{IMAGE_VER}")
       command = f"docker build --tag {IMAGE_NAME}:{IMAGE_VER} --build-arg PYTHON_VER={PYTHON_VER} -f Dockerfile ."

       if nocache:
           command += " --no-cache"
       if forcerm:
           command += " --force-rm"

       result = context.run(command, hide=hide)
       if result.exited != 0:
              print(f"Failed to build image {IMAGE_NAME}:{IMAGE_VER}\nError: {result.stderr}")
```

>>> Invoke – Example CLI - List

```
Available tasks:

  bandit           Run bandit to validate basic static code security analysis.
  black            Run black to check that Python files adhere to its style standards.
  build            Build all docker images.
  cli              Launch a bash shell inside the running Nautobot container.
  create-user      Create a new user in django (default: admin), will prompt for password.
  debug            Start Nautobot and its dependencies in debug mode.
  destroy          Destroy all containers and volumes.
  makemigrations   Run Make Migration in Django.
  nbshell          Launch a nbshell session.
  pydocstyle       Run pydocstyle to validate docstring formatting adheres to NTC defined standards.
  pylint           Run pylint code analysis.
  start            Start Nautobot and its dependencies in detached mode.
  stop             Stop Nautobot and its dependencies.
  tests            Run all tests for this plugin.
  unittest         Run Django unit tests for the plugin.
```

Docker & Docker Compose

>>> Docker

Docker container

- Docker containers are used in all projects
- Docker manages all of the dependencies (with poetry) and completely siloed
- Generally, you never run an actual docker command, you run invoke commands
- Docker is intended to be ephemeral, so should be able to destroy at any time

Docker compose

- Invoke commands heavily used, to us “Compose file inheritance” with multiple Docker compose files
- Makes use of `env` files for configuration standards


Click

>>> Click – As easy as point and Click

- Requires Click Python package installed
- Support for environment variables without extra code
- Support for prompting of custom values
- Automatic help generation

>>> Click – Key Features

- **Declarative Syntax**: Click uses a declarative syntax to define commands, arguments, and options.
- **Command Groups:** Click allows you to group related commands together, creating a structured and organized CLI application.
- **Input and Output Handling:** Click provides utilities for reading input from users and printing output to the console.
- **Options and Arguments:** Click supports both command-line options (flags) and arguments (values) that your commands can accept.
- **Automatic Type Conversion:** Click automatically converts input values to the specified data types.
- **Help Messages:** Click generates help messages for your commands and options, allowing users to learn about your CLI's functionality without extra documentation.


Cookiecutter

>>> Cookiecutter

- Reusable projects made easy with variables
- Jinja2 Templating
- Variables to be used defined in cookiecutter.json
- Internally, we save your answer to be used later
- Variables can be used for directory, file names, file content, etc.
- Pre/Post hooks
- Python API available

>>> Cookiecutter.json

```
{
    "description": "",
    "project_name": "Cookiecutter Project",
    "project_slug": "{{ cookiecutter.project_name.lower().replace(' ', '-').replace('_', '-') }}",
    "project_python_name": "{{ cookiecutter.project_name.lower().replace(' ', '_') }}",
    "version": "1.0.0"
}
```

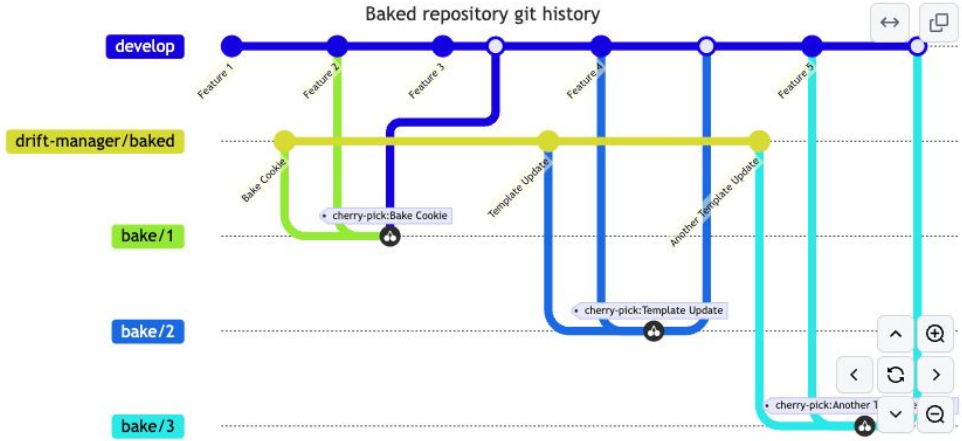
- Each key will represent a prompt when running cookiecutter
- Values are default if user chooses they're acceptable
- Jinja2 templating available within file

>>> Cookie Cutter - Coming Soon

Manage Existing Repos

How it works

- All branches are prefixed with `drift-manager/` prefix. This prefix can be customized in `.cookiecutter.json` file.
- There is `drift-manager/baked` branch in each baked repository containing:
  - baked cookie without any repository specific changes.
- Each repository with `rebake.yaml` GitHub Workflow is re-baked every Saturday at 04:00.
  - `drift-manager/baked` branch is updated by baking the cookie again using stored `.cookiecutter.json`.
    - Production Docker image is used for this.
    - New commit to `drift-manager/baked` is created containing changes since the previous baked version.
  - New feature branch (`bake/x` in the graph below) is created based on `develop`, new commits from `drift-manager/baked` are cherry-picked.
    - In case of conflicts, baked `-X theirs` version is preferred.
  - New DRAFT pull request from feature branch to `develop` is created.
  - Repository maintainer reviews and merges pull request.




Mkdocs

>>> Mkdocs

- Sleek and Contemporary Design: A modern and sleek user interface that showcases documentation with style
- Search: A robust search feature that ensures they can quickly find the information you need,
- Explicit Definitions: MkDocs provides a mkdocs.yml file for all control
- Theme support: branded to NTC's specific theme
- Plugins Support: Enhances documentation that add functionality ranging from diagrams to integrations, creating an enriched reading experience.
    - Docstrings in Action: Seamlessly incorporate docstrings directly into your documentation, ensuring your code's insight is always at hand for your users.
    - Custom plugins

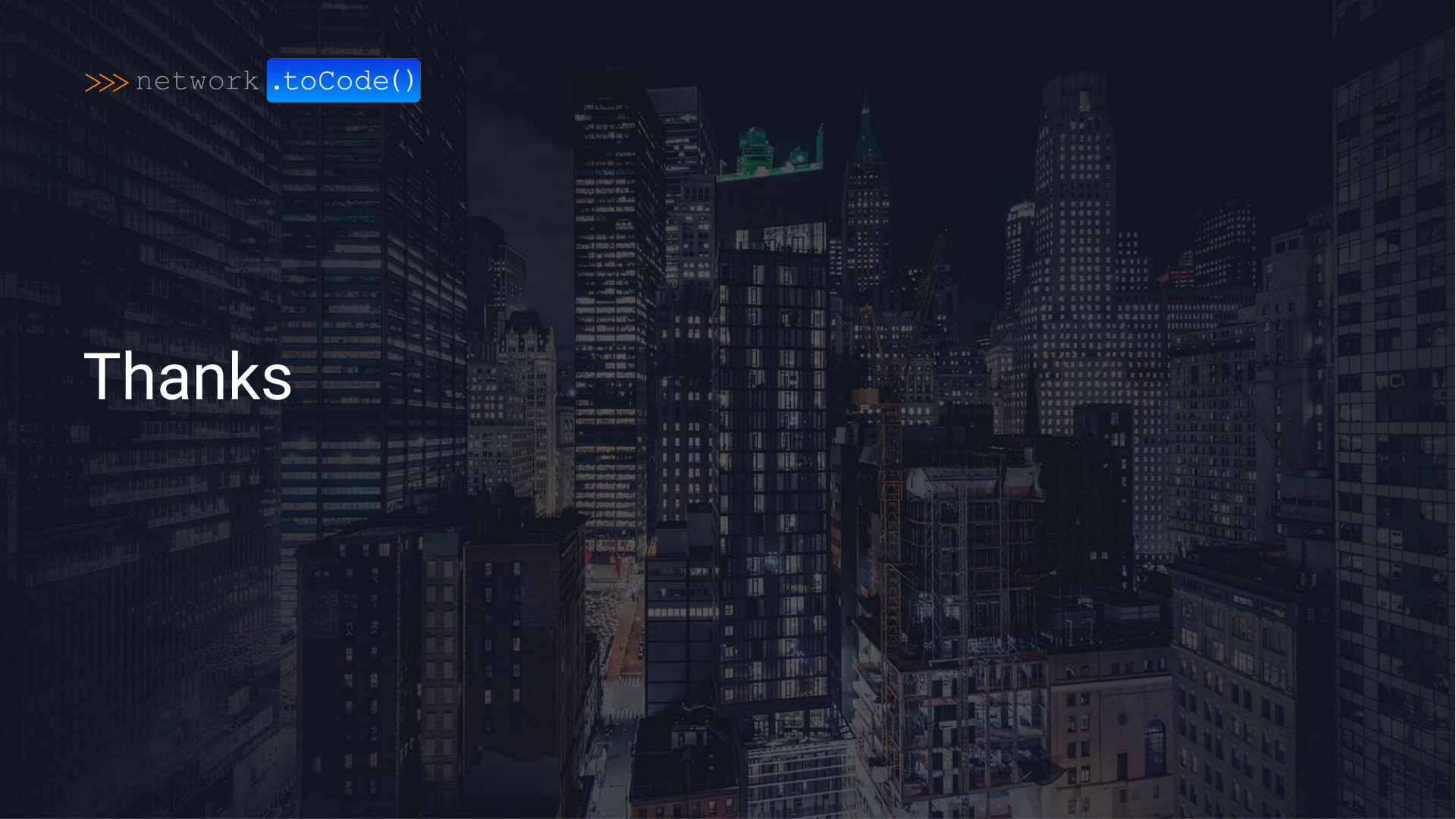

Demo

>>> Demo

```
root@a0b5c75551a0:/# cookiecutter /cookiecutter-ntc/python/
  [1/9] codeowner_github_usernames (@smith-ntc): @itdependsnetworks
  [2/9] description (): Network CLI Python Package
  [3/9] project_name (Cookiecutter Project): network_cli
  [4/9] project_slug (network-cli):
  [5/9] project_python_name (network_cli):
  [6/9] project_python_base_version (3.7): 3.8
  [7/9] Select project_with_config_settings
    1 - yes
    2 - no
    Choose from [1/2] (1):
  [8/9] Select generate_docs
    1 - no
    2 - yes
    Choose from [1/2] (1):
  [9/9] version (1.0.0):

Congratulations!  Your cookie has now been baked. It is located at /tmp/network-cli.

⚠⚠ Before you start using your cookie you must run the following commands inside your cookie:

* poetry lock
* poetry shell

root@a0b5c75551a0:/#
```

>>> Demo - Continued

```
# Update pyproject.toml to add any dependencies

root@a0b5c75551a0:/# poetry install
root@a0b5c75551a0:/# invoke build
root@a0b5c75551a0:/# invoke cli
root@a0b5c75551a0:/# invoke tests
```


>>> network .toCode()
Thanks