heist-salt Documentation

None

Jul 26, 2023

CONTENTS

CHAPTER

ONE

HEIST-SALT

1.1 Getting Started

1.1.1 Prerequisites

- Python 3.7+
- git *(if installing from source, or contributing to the project)*

1.1.2 Installation

Note: If wanting to contribute to the project, and setup your local development environment, see the `CONTRIBUTING.rst` document in the source repository for this project.

If wanting to use `heist-salt`, you can do so by either installing from PyPI or from source.

Install from PyPI

To install the latest version from PyPI:

```
# Requires Python 3.7+
pip install heist-salt
```

Install from PyPi when using Salt's onedir packages

```
salt-pip install heist-salt
```

Note: Due to this issue: https://github.com/saltstack/salt/issues/64192 you cannot run Heist-Salt alongside Salt's 3006.0 onedir packages. This was resolved in Salt's 3006.1 onedir packages.

After heist-salt is installed into the Salt's onedir packages, you need to ensure you add the Heist binary to your path. If you installed the linux 3006.1 Salt onedir packages, by default this path would be `/opt/saltstack/salt/extras-3.10/bin/`. This is where the heist binary is installed when running `salt-pip install heist-salt`. You could also directly call the full binary path instead of adding it to your path. For example:

```
# /opt/saltstack/salt/extras-3.10/bin/heist salt.minion -R /etc/heist/roster
```

Install from source

`heist-salt` can also be installed from source:

```
# Requires git and Python 3.6+
git clone https://gitlab.com/saltstack/pop/heist-salt.git
cd heist-salt
pip install -e .
```

1.1.3 Setting up a Salt master

Don't worry, this is a snap! Once Heist is installed you will need a Salt master to connect to if you are using the `salt.minion` manager. If you have an existing Salt master running or you are using a different manager such as `salt.master` you can skip this section, just run `heist` on your Salt master.

1.1.4 Using onedir binary

Download the all-in-one Salt binary for Linux x86_64.

```
wget https://repo.saltproject.io/salt/py3/onedir/latest/salt-3006.1-onedir-linux-x86_64.
↪tar.xz
```

This is to install the 3006.1 version of Salt. You can view the directory listing here: https://repo.saltproject.io/salt/py3/onedir/ to see all of the Salt versions available for download.

Extract the tarball:

```
tar -xvf salt-3006.1-onedir-linux-x86_64.tar.xz
```

This will extract a single file named *salt*. You can now use this single binary to run the Salt master.

```
./salt/salt-master
```

Now you have a running Salt master to control your minions!

1.1.5 Using onedir system packages

You can also install the onedir system packages. Please see the Salt install guide for instructions on how to install and run the Salt master.

1.1.6 Pip install Salt

If you want to pip install Salt you only need to run:

```
pip install salt
```

Now you can run the salt-master in the background:

```
salt-master -d
```

1.1.7 Making your roster

A Roster is a file used by Heist to map login information to the systems in your environment. This file can be very simple and just needs to tell Heist where your systems are and how to log into them via ssh. Open a file called `roster.cfg` and add the data needed to connect to a remote system via ssh:

```
192.168.4.4:
  username: fred
  password: freds_password
```

The roster files typically all live inside of a roster directory. But to get started will execute a single roster file with `heist`:

```
heist <heist-manager> -R roster.cfg
```

Please see heist's roster documentation for more information on rosters.

To use the `salt.minion` manager to deploy and manage a Salt Minion artifact, run the following:

```
heist salt.minion -R roster.cfg
```

To use the `salt.master` manager to deploy and manage a Salt Master artifact, run the following:

```
heist salt.master -R roster.cfg
```

Assuming your roster is correct, heist will now connect to the remote system, deploy a Salt minion, and connect it to your running master! Now you can use the same binary that you started the master with to accept your new minion's keys:

```
./salt/salt-key -A
```

Then give your minion a few seconds to authenticate and then run your first `salt` command on the newly set up minion:

```
./salt/salt \* test.version
```

That's it! Now that the minion is up you can run `salt` commands on it at breakneck speed, the full power of Salt is at your fingertips!!

1.2 What is Heist-Salt

Salt requires an agent be installed across all devices a user desires to manage. The agent normally requires a manual step to install across all devices, before being able to get started with Salt. Heist-Salt is a tool that removes this manual installation step and manages the installation for the user. Heist-Salt uses the heist tool to help manage the artifacts.

Heist Salt is a tool that can manage the deployment and installation of Salt or be used as an agentless solution for Salt. Heist-Salt uses Heist, which is a tool that helps manage any artifacts. Heist-Salt uses Heist to help deploy and install the Salt artifact. Heist-Salt can currently manage the following artifacts:

- *Heist Minion*
- *Heist Master*
- *Heist Proxy Minion*

There are two different use for Heist-Salt:

- ***Agentless Minions*: Provides an agentless solution for the user over SSH. This removes** the requirement to for the user to install an agent across each device they want to manage with Salt.
- *Bootstrapping Minions to Existing Masters*: A solution to automate the installation of the Salt onedir across all devices they want to manage. This solution can also manage upgrades of Salt, manage the configurations and manage the service.

1.3 Running Heist-Salt as unprivileged user

If you are running Heist-Salt as a non root user and using the agentless mode of Heist-Salt you need to ensure the user running Heist also has the correct permissions on Salt. Please see the running Salt as unprivileged user instructions on what permissions need to be set for your user.

CHAPTER

TWO

ARTIFACTS

2.1 Building Custom Salt Bins

To build a custom Salt binary, follow the instructions in the Salt packaging documentation.

2.1.1 salt_repo_url

By default, heist-salt will query https://repo.saltproject.io/salt/py3/onedir/repo.json to return data about the artifacts. The returned json data will include the artifact name, version and hashes of the artifact.

You can set a custom repo by setting *salt_repo_url* to a url that points to your custom repo. The custom repo needs to include a *repo.json* file and follow the directory structure of https://repo.saltproject.io/salt/py3/onedir/

2.2 Salt Artifact

Heist-salt uses Heist to deploy and manage the Salt artifact. The Salt binary Heist-Salt deploys is built using relenv. Relenv creates a reproducable and re-locatable python builds.

Heist automatically downloads artifacts from repo.saltproject.io and uses them to deploy agents. Heist will automatically download the latest artifact from the repo, unless it already exists in the *artifacts_dir* or a specific Salt version is set via the *artifact_version* option. Heist will automatically detect the target OS and download the appropriate binary. If *artifact_version* is set, heist will download the *Salt* binary version from the repo if it exists.

Note: Starting in Heist-Salt version v6.0.0, Heist will only support managing and deploying the Salt artifact 3006.0 and above.

You can deploy a custom version of Salt onedir package that includes a different version of python, or more dependency libraries. See the *Building Custom Salt Bins* documentation for more information.

When the artifacts are downloaded from the remote repo they are placed in your *artifacts_dir*. By default this location is `/var/tmp/heist/artifacts`

The downloaded executables are in a tarball and are versioned with a version number following the dash right after the name of the binary. It also includes the OS and architecture. In the case of *salt* the file looks like this: *salt-3006.0-onedir-linux-aarch64.tar.xz*.

2.3 Heist Master

You can use the `salt.master` Heist plugin to deploy and manage a Salt master artifact.

```
heist salt.master -R roster.cfg
```

This will automatically download and deploy a Salt master to the defined targets. This Heist manager will also handle the Salt master upgrades, and managing the service.

If you want to define Salt master config options to add to the master config file, you would define them in your roster file like so:

```
system_name:
  host: 192.168.1.2
  username: root
  password: "rootpassword"
  master_opts:
    log_level_logfile: debug
```

2.4 Heist Minion

There are two ways to differentiate between a Heist-Salt minion and a regular minion. A Heist-Salt minion communicates with the master over SSH. Also, a Heist-Salt minion includes a *minion_type: heist* grain. If you want to target only heist minions you can with grains targeting on a Salt Master.

```
salt -G 'minion_type:heist' test.version
```

Since Heist-Salt minions communicate over SSH if your Salt Master is attempting to check for connected minions it will not work by default for Heist Minions. You will need to set *detect_remote_minions* to be True in your Salt Master configuration. This will check for connections the Master is connected to over port 22 by default. If you are running SSH on a different port you can change the port with *remote_minions_port*. These settings will allow presence events, the manage runner and any other features that detect connected minions in to a Salt Master to work properly.

2.5 Heist Proxy Minion

You can use the `salt.proxy` Heist plugin to deploy and manage a Salt proxy minion artifact.

```
heist salt.proxy -R roster.cfg
```

This will automatically download and deploy a Salt proxy minion to the defined targets. This Heist manager will also handle the Salt proxy minion upgrades and managing the service.

If you want to define Salt proxy config options to add to the proxy config file, you would define them in your roster file like so:

```
system_name:
  host: 192.168.1.2
  username: root
  password: "rootpassword"
  proxy_opts:
    log_level_logfile: debug
```

To define the proxy type and any other needed settings for the Salt proxy minion you will need to use the `pillar` arg in the roster file. For example:

```
system_name:
  host: 192.168.1.2
  username: root
  password: "rootpassword"
  pillar:
    proxy:
      proxytype: dummy
```

This setting ensures the pillar data is set for this minion to configure the Salt proxy minion type to dummy. This will also edit your top file to ensure this pillar data is added to your pillar environment. It will use the `base` pillar environment by default or look at your Salt opts for `pillarenv`.

2.5.1 Heist Proxy Minion Grains

There are two ways to differentiate between a Heist-Salt proxy minion and a regular proxy minion. A Heist-Salt minion communicates with the master over SSH. Also, a Heist-Salt minion includes a *minion_type: heist_proxy* grain. If you want to target only heist proxy minions, you can with grains targeting on a Salt Master.

```
salt -G 'minion_type:heist_proxy' test.version
```

Since Heist-Salt proxy minions communicate over SSH, if your Salt Master is attempting to check for connected minions, it will not work by default for Heist Minions. You will need to set *detect_remote_minions* to True in your Salt Master configuration. This will check for connections the Master is connected to over port 22 by default. If you are running SSH on a different port, you can change the port with *remote_minions_port*. These settings will allow presence events, the manage runner and any other features that detect connected minions to a Salt Master to work properly.

CHAPTER

THREE

RELEASES

3.1 v3.0.0 (2021-07-15)

3.1.1 Changelog

Removed

- Remove *accept_keys* in favore of *generate_keys*. (#23)

Fixed

- Fixed setup.py to include the correct directory when building package. (#24)

Added

- Add towncrier tool to the heist-salt project to help manage CHANGELOG.md file. (#22)

3.2 v4.0.0 (2021-08-18)

3.2.1 Changelog

Deprecated

- Deprecate artifacts.salt.fetch and artifacts.salt.verify_hash in favor of heist.artifacts.init.{fetch,verify}. This will be removed in Heist-Salt version v5.0.0. (#28)

Added

- Allow users to "bootstrap" their minions. This feature will allow a user to deploy a salt minion and point it to a different master. Heist will no longer manage this minion after deployment. (#5)
- Migrate artifact calls to heist/artifact/init.py (#27)

3.3 v5.0.0 (2021-10-25)

3.3.1 Changelog

Removed

- Removed salt.artifacts.salt.fetch in favor of heist.artifacts.init.fetch Removed salt.artifacts.salt.verify_hash in favor of heist.artifacts.init.verify (#28)

Fixed

- Allow heist to work with pkg version. For example 3004-1. It will also continue to allow use of old versioning (3004). (#30)

3.4 v5.1.0 (2021-11-04)

3.4.1 Changelog

Fixed

- Fix using heist-salt with Salt versions < 3003 when cleaning the connections. (#32)
- Set minion_type heist grain for bootstrap heist minions. (#33)
- Do not traceback when heist cannot connect to the target host. (#34)

Added

- Add offline_mode option to skip downloading artifact step. (#31)

3.5 v5.2.0 (2021-12-07)

3.5.1 Changelog

Fixed

- Do not create files outside of /var/tmp/heist_<user>. This ensures the files /var/log/salt/minion and /etc/salt/minion.d do not get created. (#35)

Added

- Add –onedir option to allow user to use onedir Salt packages. (#36)

3.6 v5.3.0 (2022-07-12)

3.6.1 Changelog

Deprecated

- The support for singlebin packages will be removed in v6.0.0. Please use *–onedir* going forward. In the v6.0.0 release, onedir will be the default option and the *–onedir* argument will be removed.

Fixed

- Fix permission denied error when using sudo. (#40)

Added

- Allow users to use the new Heist raw service. (#37)
- Add Support for win_service Plugin for Windows (#38)
- Add Windows Support for SingleBin (#39)
- Added support for OneDir on Windows (#43)

3.7 v5.3.1 (2022-07-13)

3.7.1 Changelog

Fixed

- Add packaging as a base requirement for heist-salt. (#50)

3.8 6.0.0 (2023-05-03)

3.8.1 Changelog

Changed

- Removing the single bin artifact in favor of the onedir. (#49)
- Heist-Salt will not require salt as a pip dependency. It now checks on startup if salt is installed or not. This allows a user to have Salt installed without pip. (#61)

Fixed

- Return success if a machine is bootstrapped successfully. (#46)
- Ensure raw_service restarts on upgrade of artifact. Do not re-deploy alias files on upgrade. (#62)

Added

- Add ability to only manage the service of an already deployed artifact. Add ability to clean the previously deployed artifact before deploying a new artifact. (#25)
- added some better error handling. it isn't prfect yet. but at least should catch the big stuff. (#47)
- Add aliases for salt-call and salt-minion binaries. (#48)
- Add ability to detect there is a new Salt package and upgrade it. (#51)
- Add ability to deploy Salt proxy (#52)
- Add ability for Heist-Salt to deploy Salt Master. (#53)
- Check if the artifact has already been deployed and verify the artifact. Also, start the service if not started. (#60)
- Added support in Heist-Salt for the new Salt 3006.0 packages. This is a breaking change as Heist-Salt will only support Salt packages 3006.0 and above. (#66)
- https://gitlab.com/saltstack/pop/heist/-/issues/120 - started with heist-salt roster instead of going through the heist way first. (#120)

3.9 6.0.1 (2023-05-08)

3.9.1 Changelog

Fixed

- Ensure we create the pki directory on windows targets when using Salt Minion Heist manager. (#67)

3.10 6.0.2 (2023-05-18)

3.10.1 Changelog

Fixed

- Daemonize the Salt master, minion and proxy (-d). (#69)

CHAPTER

FOUR

BOOTSTRAP

4.1 Bootstrapping Minions to Existing Masters

Heist can be used to bootstrap minions to connect to existing masters and not be connected back through the ssh tunnel. This allows for minions to be deployed and managed to many system in a very streamlined way.

The only change that needs to take place is that *bootstrap: True* and the master that the target system needs to connect to is defined in minion_opts inside the roster:

```
hostname:
  username: frank
  password: horsebatterystaple
  bootstrap: True
  minion_opts:
    master: salt
```

This now tells the minion to deploy the salt minion artifact and connect to the specific master located at the host named *salt*. When heist is stopped it will not delete anything and will immediately kill the connection. With this approach you will need to manually accept the key on the master that the minions are now communicating with.

CHAPTER

FIVE

AGENTLESS

5.1 Agentless Minions

Heist-Salt agentless is the default use case for Heist-Salt. Heist-Salt will automatically deploy and install the Salt minion for the user and automatically generate and accept the Salt key. Heist-Salt will also establish a SSH tunnel connection between the Salt Master and the Salt Minion. Once this connection is established and the Salt Minion is deployed a user can then use Salt to manage the minion, with all of the communication over SSH. When the Heist-Salt process is killed it cleans everything up, including removing the Salt Minion agent, the Salt keys, and the SSH tunnel.

CHAPTER

SIX

TRANSPORT

6.1 Pluggable Tunnel System

The tunnel system in Heist and Heist Salt is pluggable, but currently the only supported tunnel is asyncssh. The tunnel plugin system is used to create connections from heist to the minions. This connection is used to do the following:

6.1.1 1. Manage files and binaries

The tunnel helps copy binaries and files over to the minion. The asyncssh tunnel, for example, uses sftp to copy over these files via ssh.

6.1.2 2. Manage a tunnel

Establish an ssh tunnel from the Heist Salt minions to the Salt Master's publish and master ports. The asyncssh tunnel, for example creates an ssh tunnel on the minions on ports *44505* and *44506* back to the Salt Master's *master_port* and *publish_port* ports, by default (*4505* and *4506*)

Salt minion's set the *publish_port* to *44505* and the *master_port* to *44506* by default. You can change the minion's *publish_port* and *master_port* by editing the minion_opts in your roster.

```
system_name:
  host: 192.168.1.2
  username: root
  password: "rootpassword"
  minion_opts:
    master_port: 54506
    publish_port: 54505
```

This will edit the minion config to point the *master_port* to 54506 and the *publish_port* to 54505. The SSH tunnel will also use these values to setup the tunnel on these ports on the minion side.

6.1.3 3. Handling Disconnects

The tunnel detects if there is a disconnect and will attemt to re-connect automatically. The *checkin_time* configuration is used to determine in seconds how often to check if the connection is established.

CHAPTER

SEVEN

DEVELOPMENT

7.1 Changelog

The Heist-Salt project uses the towncrier tool to manage the CHANGELOG.md file. Theis tool helps manage the changelog and help prevent merge conflicts when managing one file. This tool adds changelog entries into separate files and before a release we simply need to run `towncrier --version=<version>` for it to compile the changelog correctly.

7.1.1 How do I add a changelog entry

To add a changelog entry you will need to add a file in the *changelog* directory. The file name should follow the syntax `<issue #>.<type>`.

The types are in alignment with keepachangelog:

removed:
: any features that have been removed

deprecated:
: any features that will soon be removed

changed:
: any changes in current existing features

fixed:
: any bug fixes

added:
: any new features added

For example if you are fixing a bug for issue number #1234 your filename would look like this: changelog/1234.fixed. The contents of the file should contain a summary of what you are fixing. If there is a legitimate reason to not include an issue number with a given contribution you can add the MR number as the file name (`<MR #>.<type>`).

If your MR does not align with any of the types, then you do not need to add a changelog entry.

7.1.2 How to generate the changelog

This step is only used when we need to generate the changelog right before releasing. You should NOT run towncrier on your MR, unless you are preparing the final MR to update the changelog before a release.

You can run the *towncrier* tool directly or you can use nox to help run the command and ensure towncrier is installed in a virtual environment. The instructions below will detail both approaches.

If you want to see what output towncrier will produce before generating the change log you can run towncrier in draft mode:

```
towncrier --draft --version=3001
```

```
nox -e 'changelog(draft=True)' -- 3000.1
```

Version will need to be set to whichever version we are about to release. Once you are confident the draft output looks correct you can now generate the changelog by running:

```
towncrier --version=3001
```

```
nox -e 'changelog(draft=False)' -- 3000.1
```

After this is run towncrier will automatically remove all the files in the changelog directory.

CHAPTER

EIGHT

GENERATING THE SALT MINION KEY

By default, when a Salt heist minion is deployed the minion will generate the key and copy back over to the master. On the master, it is copied into the accepted keys folder in the master's *pki_dir* directory. If you do not want to generate the key by default, you can set the *generate_keys* config option to *False*. You will need to manually accept the minion's key on the master if this is turned off.

Note: Windows targets do not currently support this feature, but it will added when Salt releases 3006.1

CHAPTER

NINE

GET INVOLVED

9.1 Contributing Guide

Contributions are what make the open source community such an amazing place to learn, inspire, and create in. Any contributions you make are **greatly appreciated!**

9.1.1 TL;DR Quickstart

1. Have pre-requisites completed:
   - `git`
   - `nox`
   - `pre-commit`
   - Python 3.6+
2. Fork the project
3. `git clone` your fork locally
4. Create your feature branch (ex. `git checkout -b amazing-feature`)
5. Setup your local development environment

```
# setup venv
python3 -m venv .venv
source .venv/bin/activate
pip install -U pip setuptools wheel pre-commit nox

# pre-commit configuration
pre-commit install
```

6. Hack away!
7. Commit your changes (ex. `git commit -m 'Add some amazing-feature'`)
8. Push to the branch (ex. `git push origin amazing-feature`)
9. Open a pull request

For the full details, see below.

9.1.2 Ways to contribute

We value all contributions, not just contributions to the code. In addition to contributing to the code, you can help the project by:

- Writing, reviewing, and revising documentation, modules, and tutorials
- Opening issues on bugs, feature requests, or docs
- Spreading the word about how great this project is

The rest of this guide will explain our toolchain and how to set up your environment to contribute to the project.

9.1.3 Overview of how to contribute to this repository

To contribute to this repository, you first need to set up your own local repository:

- *Fork, clone, and branch the repo*
- *Set up your local preview environment*

After this initial setup, you then need to:

- *Sync local master branch with upstream master*
- Edit the documentation in reStructured Text
- *Preview HTML changes locally*
- Open a PR

Once a merge request gets approved, it can be merged!

9.1.4 Prerequisites

For local development, the following prerequisites are needed:

- git
- Python 3.6+
- Ability to create python venv

Windows 10 users

For the best experience, when contributing from a Windows OS to projects using Python-based tools like `pre-commit`, we recommend setting up Windows Subsystem for Linux (WSL), with the latest version being WSLv2.

The following gists on GitHub have been consulted with success for several contributors:

- Official Microsoft docs on installing WSL
- A list of PowerShell commands in a gist to Enable WSL and Install Ubuntu 20.04
  - Ensure you also read the comment thread below the main content for additional guidance about using Python on the WSL instance.

We recommend Installing Chocolatey on Windows 10 via PowerShell w/ Some Starter Packages. This installs `git`, `microsoft-windows-terminal`, and other helpful tools via the awesome Windows package management tool, Chocolatey.

`choco install git` easily installs `git` for a good Windows-dev experience. From the `git` package page on Chocolatey, the following are installed:

- Git BASH
- Git GUI
- Shell Integration

9.1.5 Fork, clone, and branch the repo

This project uses the fork and branch Git workflow. For an overview of this method, see Using the Fork-and-Branch Git Workflow.

- First, create a new fork into your personal user space.
- Then, clone the forked repo to your local machine.

```
# SSH or HTTPS
git clone <forked-repo-path>/heist-salt.git
```

Note: Before cloning your forked repo when using SSH, you need to create an SSH key so that your local Git repository can authenticate to the GitLab remote server. See GitLab and SSH keys for instructions, or Connecting to GitHub with SSH.

Configure the remotes for your main upstream repository:

```
# Move into cloned repo
cd heist-salt

# Choose SSH or HTTPS upstream endpoint
git remote add upstream git-or-https-repo-you-forked-from
```

Create new branch for changes to submit:

```
git checkout -b amazing-feature
```

9.1.6 Set up your local preview environment

If you are not on a Linux machine, you need to set up a virtual environment to preview your local changes and ensure the *prerequisites* are met for a Python virtual environment.

From within your local copy of the forked repo:

```
# Setup venv
python3 -m venv .venv
# If Python 3.6+ is in path as 'python', use the following instead:
# python -m venv .venv

# Activate venv
source .venv/bin/activate
# On Windows, use instead:
# .venv/Scripts/activate
```

(continues on next page)

(continued from previous page)

```
# Install required python packages to venv
pip install -U pip setuptools wheel pre-commit nox
pip install -r requirements.txt

# Setup pre-commit
pre-commit install
```

`pre-commit` and `nox` Setup

This project uses pre-commit and nox to make it easier for contributors to get quick feedback, for quality control, and to increase the chance that your merge request will get reviewed and merged.

`nox` handles Sphinx requirements and plugins for you, always ensuring your local packages are the needed versions when building docs. You can think of it as `Make` with superpowers. It is also used to execute the rest of the test suite.

What is pre-commit?

`pre-commit` is a tool that will automatically run local tests when you attempt to make a git commit. To view what tests are run, you can view the `.pre-commit-config.yaml` file at the root of the repository.

One big benefit of pre-commit is that *auto-corrective measures* can be done to files that have been updated. This includes Python formatting best practices, proper file line-endings (which can be a problem with repository contributors using differing operating systems), and more.

If an error is found that cannot be automatically fixed, error output will help point you to where an issue may exist.

9.1.7 Sync local master branch with upstream master

If needing to sync feature branch with changes from upstream master, do the following:

Note: This will need to be done in case merge conflicts need to be resolved locally before a merge to master in the upstream repo.

```
git checkout master
git fetch upstream
git pull upstream master
git push origin master
git checkout my-new-feature
git merge master
```

9.1.8 Preview HTML changes locally

To ensure that the changes you are implementing are formatted correctly, you should preview a local build of your changes first. To preview the changes:

```
# Activate venv
source .venv/bin/activate
# On Windows, use instead:
# .venv/Scripts/activate

# Generate HTML documentation with nox
nox -e 'docs-html(clean=False)'

# Sphinx website documentation is dumped to docs/_build/html/*
# You can view this locally
# firefox example
firefox docs/_build/html/index.html
```

Note: If you encounter an error, Sphinx may be pointing out formatting errors that need to be resolved in order for `nox` to properly generate the docs.

9.1.9 Testing a `pop` project

```
# View all nox targets
nox -l

# Run tests
nox -e 'tests'
```

This project is a `pop` project which makes use of `pytest-pop`, a `pytest` plugin. For more information on `pytest-pop`, and writing tests for `pop` projects:

- pytest-pop README
- pytest documentation

9.2 License

Note: For a simplified breakdown of license information, it may be helpful to use tl;drLegal.

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.
```

(continues on next page)

(continued from previous page)

```
      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."
```

(continues on next page)

(continued from previous page)

```
   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
```

(continues on next page)

(continued from previous page)

```
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
    any Contribution intentionally submitted for inclusion in the Work
    by You to the Licensor shall be under the terms and conditions of
    this License, without any additional terms or conditions.
    Notwithstanding the above, nothing herein shall supersede or modify
    the terms of any separate license agreement you may have executed
    with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
    names, trademarks, service marks, or product names of the Licensor,
    except as required for reasonable and customary use in describing the
    origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
    agreed to in writing, Licensor provides the Work (and each
    Contributor provides its Contributions) on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied, including, without limitation, any warranties or conditions
    of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
    PARTICULAR PURPOSE. You are solely responsible for determining the
    appropriateness of using or redistributing the Work and assume any
    risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
    whether in tort (including negligence), contract, or otherwise,
    unless required by applicable law (such as deliberate and grossly
    negligent acts) or agreed to in writing, shall any Contributor be
    liable to You for damages, including any direct, indirect, special,
    incidental, or consequential damages of any character arising as a
    result of this License or out of the use or inability to use the
    Work (including but not limited to damages for loss of goodwill,
    work stoppage, computer failure or malfunction, or any and all
    other commercial damages or losses), even if such Contributor
    has been advised of the possibility of such damages.
```

(continues on next page)

(continued from previous page)

```
   9. Accepting Warranty or Additional Liability. While redistributing
      the Work or Derivative Works thereof, You may choose to offer,
      and charge a fee for, acceptance of support, warranty, indemnity,
      or other liability obligations and/or rights consistent with this
      License. However, in accepting such obligations, You may act only
      on Your own behalf and on Your sole responsibility, not on behalf
      of any other Contributor, and only if You agree to indemnify,
      defend, and hold each Contributor harmless for any liability
      incurred by, or claims asserted against, such Contributor by reason
      of your accepting any such warranty or additional liability.

   END OF TERMS AND CONDITIONS

   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "[]"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright 2021 VMware, Inc.

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

CHAPTER

TEN

INDICES AND TABLES

- modindex