
Panhandler Documentation

Release 1.0

sp-solutions

Apr 17, 2019

Contents:

1	Welcome to panhandler!	1
2	Running Panhandler	3
2.1	Running the Panhandler Docker Container	3
2.2	Building Panhandler	4
2.3	Running Panhandler manually	4
2.4	Requirements	5
3	Using Panhandler	7
3.1	Access the web portal	7
3.2	Set the Configuration Target	7
3.3	Choose Skillets to View by Category	7
3.4	Select the Template to Load	8
4	Adding a New Skillet Repository	9
4.1	Import a New Skillet	9
4.2	Update a Skillet Repository	10
5	Panhandler Metadata Files	13
5.1	Basic concepts	13
5.2	YAML syntax	13
5.3	Snippet details	14
5.4	Hints	15
6	Example Skillet	17
6.1	XML Fragment	17
6.2	.meta-cnc file	18
6.3	Rendered Form	18
7	Panhandler Environments	19
7.1	What is an Environment	19
7.2	Unlocking Environments	19
8	Keeping Up to Date	23
8.1	Ensuring you have the latest docker image	23
9	About	25

10 Disclaimer	27
11 Indices and tables	29

CHAPTER 1

Welcome to panhandler!

Panhandler is a lightweight utility used to aggregate and view or load configuration templates. The primary focus is PAN-OS devices such as the NGFW or Panorama yet may be extended to other elements such as Terraform and 3rd party devices.

Using predefined templates helps fast-track the loading of well known or recommended configurations without extensive searching and scrolling through GUI-click documentation. Each collection of configuration templates are known as *skillets* that are either preloaded into panhandler at runtime or can be manually added as needed.

Skillets can be based on xml, json, text or any other config type used by each device. They are grouped by output action including:

- panos: load into a NGFW and commit
- panorama: load into Panorama and commit
- gpcs: load into Panorama with a push to GPCS
- template: simple text render to the screen

To load a configuration into a device with panhandler, the user simply has to add the target information for the device to be configured, select the skillet to load, enter the form data, and submit. Panhandler then captures the form data, grabs each configuration element, and loads into the specified device.

Running Panhandler

The recommended way to run panhandler is to pull and run the docker container.

2.1 Running the Panhandler Docker Container

To get the latest version of panhandler as a docker container.

2.1.1 Using a standard web port

```
docker run -t -p 80:80 paloaltonetworks/panhandler
```

Then access the UI via <http://localhost:80>

The default username and password is: *paloalto* and *panhandler*

2.1.2 Using an alternate TCP port

If port 80 is unavailable, you can switch to a different port. This example uses port 9999.

```
docker run -t -p 9999:80 paloaltonetworks/panhandler
```

Then access the UI via <http://localhost:9999>

To persist any environments and secrets, you can mount a volume on the */root/.pan_cnc* folder like this:

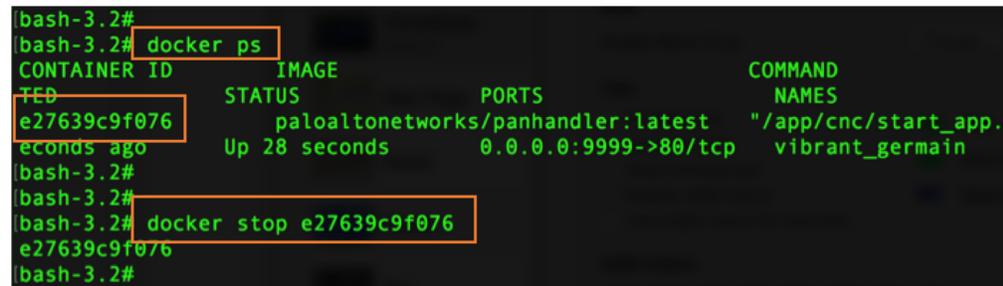
```
docker run -t -p 9999:80 -v ~/.pan_cnc:/root/.pan_cnc paloaltonetworks/panhandler
```

Note: The *-t* option for *terminal* allows you to view panhandler output data in the terminal window. This is useful for determining any skillet errors that write to terminal output.

2.1.3 Stopping the docker container

The docker container runs in the background. You can stop the container by using its container ID.

```
docker ps
docker stop { CONTAINER ID }
```

A terminal window showing the execution of Docker commands. The prompt is 'bash-3.2#'. The first command is 'docker ps', which outputs a table of running containers. The second command is 'docker stop e27639c9f076', which successfully stops the container. The container ID 'e27639c9f076' is highlighted with a red box in both the command and the output table.

CONTAINER ID	IMAGE	PORTS	COMMAND NAMES
e27639c9f076	paloaltonetworks/panhandler:latest	0.0.0.0:9999->80/tcp	"/app/cnc/start_app.. vibrant_germain

Note: If you need to remove the container, enter `docker rm { CONTAINER ID }` with CONTAINER ID as the ID used to stop. You must stop the container before deleting.

2.2 Building Panhandler

If you want to build panhandler from source (which is not recommended). You will need to update the git submodules, install the pip python requirements for both the app and also CNC, create the local db, and create a local user.

```
git clone https://github.com/PaloAltoNetworks/panhandler.git
cd panhandler
git submodule init
git submodule update
pip install -r requirements.txt
pip install -r cnc/requirements.txt
./cnc/manage.py migrate
./cnc/manage.py shell -c "from django.contrib.auth.models import User; User.objects.
↪create_superuser('paloalto', 'admin@example.com', 'panhandler')"
```

2.3 Running Panhandler manually

To start the application on your local machine on port 80:

```
cd panhandler/cnc
celery -A pan_cnc worker --loglevel=info
manage.py runserver 80
```

To use a different port, supply a different argument to the runserver command above. In this case, the server will start up on port 80. Browse to <http://localhost> in a web browser to begin. The default login credentials are 'paloalto' and 'panhandler'

2.4 Requirements

Panhandler has been tested to work on Docker version: 18.09.1 (Mac) and 18.09.0 (Linux). Please ensure you have the latest docker version installed for the best results.

Once installed and running, use your web browser to access panhandler.

3.1 Access the web portal

For your local device:

`http://localhost:80` (for a standard web port)

`http://localhost:9999` (using a defined port, eg. 9999)

The default username and password is: *paloalto* and *panhandler*

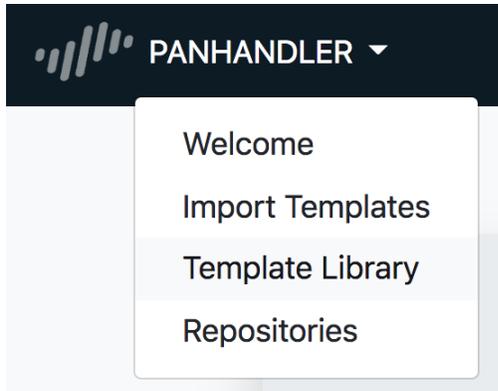
3.2 Set the Configuration Target

Before choosing skillets to load, set the configuration target IP and username/password credentials. This stores the device credentials to be used for API access.

Jump to *Panhandler Environments* to set the environment.

3.3 Choose Skillets to View by Category

From the main panhandler menu, select *Templates Library* to see a list of skillets to load.



Hit *Go* for the category of skillet required. Key categories include PAN-OS, Panorama, GPCS, and Templates for simple text render to screen.

3.4 Select the Template to Load

A list of templates will be available to load into your device. Select the desired item and enter the form data.

A screenshot of a web form titled "Pan-OS Configuration". The form has a header bar with the title and a gear icon. Below the header, the main content area is titled "Customize Configuration Variables". Under this title, there is a label "NGFW management IP address:" followed by a text input field containing the value "192.168.0.1". Below the input field is a blue "Submit" button.

The final form will be the target information for API config loading. Confirm the correct values and submit.

Warning: Validate the device type and software version matches the skillet. For example, you will get errors if trying to load a Panorama template into a firewall. There are also cases where you cannot mix software versions and loading a v8.1 configuration into a v8.0 device will result in errors.

Warning: Some templates may have dependencies requiring elements to be previously loaded into the system or from other templates. Examples may be certificates, security objects, log forwarding profiles, etc. Check template documentation and look for any specific dependencies.

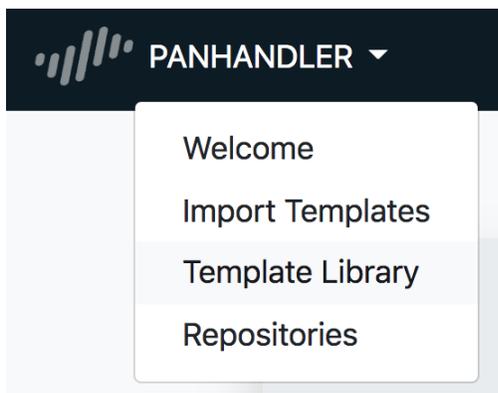
Once the load has completed, you can select another template to load to the same device or choose another Environment to load a configuration to another device.

Adding a New Skillet Repository

Panhandler is preloaded with a wide set of skillets yet you may still have to manually add skillet repos.

4.1 Import a New Skillet

From the main menu, choose *Import Templates*.



The import repository fields allow you to specify the repo name, repo url, and the branch to import.

Import Repository ⓘ ⚙

Enter a valid git url and desired branch below

Repository Name:

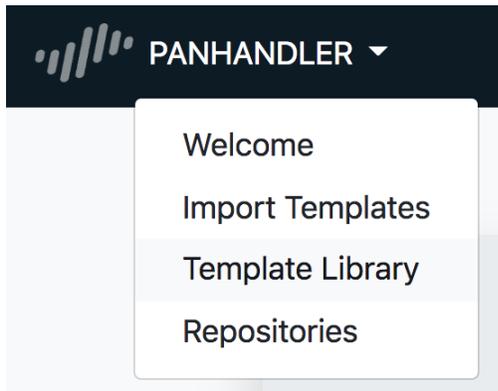
Git Repository HTTPS URL:

Branch:

Once successful, you will see the complete list of imported repositories including the newly added repo. At this stage, going to the *Template Library* will show any additional skillets in their respective categories.

4.2 Update a Skillet Repository

From the main menu, choose *Repositories*.



Click on *Details* for the repository of interest.

Repository Detail for aws-jenkins-exploit

Details

Terraform templates

<https://github.com/nembery/terraform.git>

Latest Updates

#	Message	Author	Date
1	update bootstrap vars	Nathan	2019-02-05 21:44:18-05:00
2	update labels in meta	Nathan	2019-02-05 21:41:29-05:00
3	update label on bootstrap meta-cnc	Nathan	2019-02-05 21:35:26-05:00

Commit History for branch: master

Metadata files

#	Label	Description
1	Pan-OS Bootstrap for Jenkins Exploit	Pan-OS Bootstrap for Jenkins Exploit
2	Pan-OS Bootstrap for Jenkins Exploit	Pan-OS Bootstrap for Jenkins Exploit
3	Step 1 AWS Infrastructure Jenkins Exploit	AWS Infrastructure Jenkins Exploit
4	Step 2 Pan-OS configuration for Jenkins Exploit	Pan-OS configuration for Jenkins Exploit

All Defined metadata files in repository: aws-jenkins-exploit

Update To Latest
Remove Repository
Repositories

The repo window will show a description of the repo along with the last few content changes.

Choose *Update to Latest* to check for and pull template updates.

Note: *Already up to date* will show that no changes were made to the source skillet and no updates required.

Panhandler Metadata Files

The heart of Panhandler is the *.meta-cnc.yaml* file. This allows a set of configuration snippets, known as a *skillet*, to be shared and consumed as a single unit. For example, to configure a default security profile you may need to configure multiple different parts of the PAN-OS configuration. Panhandler allows you to group those different ‘pieces’ and share them among different devices as a single unit. Often times these configuration bits (affectionately called ‘skillets’) need slight customization before deployment to a new device. The *.meta-cnc.yaml* file provides a means to templatzize these configurations and present a list of customization points, or variables, to the end user or consumer.

5.1 Basic concepts

In order to add multiple ‘bits’ of configuration to a device, we need to know the following things:

- XML Configuration fragment with optional variables defined in jinja2 format
- xpath where this xml fragment should be inserted into the candidate configuration
- the order in which these XML fragments must be inserted
- a list of all variables that require user input
- target version requirements. For example: PAN-OS 8.0 or higher

This is all accomplished by adding multiple files each containing an XML configuration fragment and a *.meta-cnc.yaml* file that describes the load order, variables, target requirements, etc.

5.2 YAML syntax

Each *skillet* is structured as a series of files in a single directory. This directory may contain a number of template files (XML, YAML, JSON, etc) and a *.meta-cnc.yaml* file. Note the following:

1. A *.meta-cnc.yaml* file that is formatted with using YAML with the following format:

```
name: config_set_name
description: config_set description
extends: name_of_required_major_skilllet
variables:
  - name: INF_NAME
    description: Interface Name
    default: Ethernet1/1
    type_hint: text
snippets:
  - xpath: some/xpath/value/here
    name: config_set_knickname
    file: filename of xml snippet to load that should exist in this directory
```

2. Multiple configuration files. Each should contain a valid template fragment and may use jinja2 variables. These templates may be XML, JSON, YAML, Text, etc. For PAN-OS devices, these are XML fragments from specific stanzas of the PAN-OS device configuration tree.

5.3 Snippet details

Each `.meta-cnc.yaml` file must contain the following top-level keys:

- `name`: name of this configuration set
- `description`: Short description
- `extends`: name of another skilllet that is a requirement for this one. PAN-OS and Panorama types will load extends prior to loading this one
- `variables`: Described in detail below
- `snippets`: a dict containing the following keys
 - `name`: knickname of the skilllet
 - `file`: relative path to the configuration template
 - `xpath` (optional): XPath where this fragment belongs in the target OS hierarchy (for XML skilllets)

Note: Each Metadata file type has it's own format for the 'snippets' section. `file` and `xpath` are only used in `panos` and `panorama` types. Other types such as `template` or `rest` may have a different format.

5.3.1 Snippet details per Metadata type

Required fields for each metadata type is listed below:

- **panos, panorama, panorama-gpcs**
 - `name` - name of this snippet
 - `file` - path to the XML fragment to load and parse
 - `xpath` - XPath where this fragment belongs
- **template**
 - `name` - name of this snippet
 - `file` - path to the jinja2 template to load and parse

- **terraform**
 - None - snippets are not used for terraform
- **rest**
 - name - name of the snippet
 - path - REST URL path component (*path: /api/?type=keygen&user={{ username }}&password={{ password }}*)
 - operation - type of REST operation (GET, POST, DELETE, etc)
 - payload - path to a jinja2 template to load and parse to be send as POSTed payload
 - content_type - Content-Type header to add. For example: application/json
 - accepts_type - Accepts-Type header to add. For examle: /

Each skillet can define nultiple variables that will be interpolated using the Jinja2 templating language. Each variable defined in the *variables* list should define the following:

1. name: The name of the variable found in the skillets. For example:

```
{{ name }}
```

2. description: A brief description of the variable and it's purpose in the configuration
3. label: Human friendly label to display to user
4. extends: Name of another skillet to load
5. default: A valid default value which will be used if not value is provided by the user
6. type_hint: Used to constrain the types of values accepted. May be implemented by additional third party tools. Examples are *text*, *text_field*, *ip_address*, *password*, *dropdown*, and *checkbox*.

5.4 Hints

5.4.1 Ensuring all variables are defined

When working with a large amount of configuration tmlates, it's easy to miss a variable definition. Use this one-liner to find them all.

cd into a skillet dir and run this to find all vars

```
grep -r '{{' . | cut -d'{' -f3 | awk '{ print $1 }' | sort -u
```

5.4.2 YAML Syntax

YAML is notoriously finicky about whitespace and formatting. While it's a relatively simple structure and easy to learn, it can often also be frustrating to work with, especially for large files. A good reference to use to check your YAML syntax is the [YAML Lint site](#).

In this example, we will create a skillet that allows the user to customize a single variable.

6.1 XML Fragment

First, we'll extract the parts of the configuration that comprise this 'unit' of configuration changes (a skillet). For example, this portion of the configuration describes the log-settings we would like to modify:

```
<system>
  <match-list>
    <entry name="dhcp-log-match">
      <send-syslog>
        <member>mgmt-interface</member>
      </send-syslog>
      <filter>(eventid eq lease-start)</filter>
    </entry>
  </match-list>
</system>
<syslog>
  <entry name="mgmt-interface">
    <server>
      <entry name="mgmt-intf">
        <transport>UDP</transport>
        <port>514</port>
        <format>BSD</format>
        <server>{{ MGMT_IP }}</server>
        <facility>LOG_USER</facility>
      </entry>
    </server>
  </entry>
</syslog>
```

Notice here we have defined one variable: *MGMT_IP*. This will allow the user to insert their own management ip when deploying.

6.2 .meta-cnc file

```
name: example_log_setting
label: Log Setting Example
description: Example log setting to configure syslog
type: panos
extends:

labels:
  service_type: userid

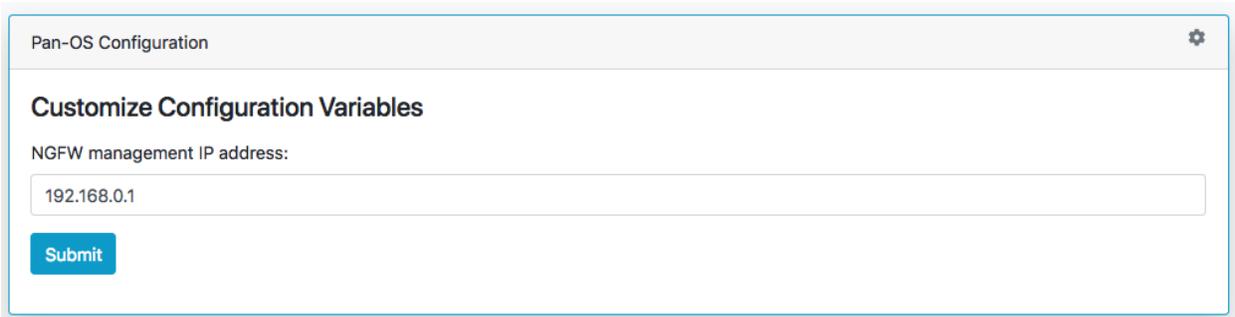
variables:
  - name: MGMT_IP
    description: NGFW management IP address
    default: 192.168.0.1
    type_hint: ip_address

snippets:
  - name: log_settings
    xpath: /config/shared/log-settings
    file: log_settings.xml
```

In this file, we give some basic information about what this skillet will do, what configuration bits will be applied, and what variables the user can customize. Notice in the ‘variables’ section, we specify a variable entry with a ‘name’ that matches the variable defined in the XML fragment. The ‘snippets’ section will inform Panhandler where in the configuration this fragment should be inserted (xpath) and where to find the fragment (file).

6.3 Rendered Form

This *.meta-cnc.yaml* will produce the following web form in Panhandler:



The screenshot shows a web interface for 'Pan-OS Configuration'. It features a section titled 'Customize Configuration Variables'. Below this title, there is a label 'NGFW management IP address:' followed by a text input field containing the value '192.168.0.1'. A blue 'Submit' button is located below the input field.

Panhandler Environments

Often times, it is desirable to store environment specific data outside of a git repository. Panhandler provides a mechanism to do this using 'Environments'.

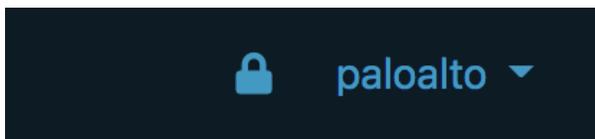
7.1 What is an Environment

An environment is a collection of secrets that can be loaded and managed as a unit. For example, you may want to keep all AWS related secrets together in an environment called 'AWS'. When panhandler displays a web form from a configuration set, any variables from the configuration template that share a name with a secret in the currently loaded environment, that value will be pre-populated.

This is especially useful if you have multiple environments such as 'AWS-QA', 'AWS-PROD', and 'AWS-DEV'.

7.2 Unlocking Environments

To load an environment, click on the 'lock' icon on the right of the navigation bar.



You will be presented with an unlock password dialog. This password will be used to protect any secrets you store in your environments in an encrypted file in your home directory. If this encrypted file does not already exist it will be created and protected with the password you enter here.

Unlock Environments ?

Enter master passphrase to unlock the environments configuration

Master PassPhrase:

Once unlocked, you can manage your environments by creating new ones, cloning, configuring, or deleting existing ones.

Local Panorama
PaloAltoNetworks Panorama

Local Pan-OS
PaloAltoNetworks Pan-OS VM50

Remote Pan-OS
afdasdf

Vistoq-Demo-B
Vistoq in GCP

Choosing the ‘Configure’ option on an environment allows you to add, remove, or overwrite secrets stored within them.

Environment: Local Panorama

Secrets stored in this environment

#	Key	Value	Options
1	TARGET_IP	192.168.55.7	Delete Secret
2	TARGET_USERNAME	admin	Delete Secret
3	TARGET_PASSWORD	admin	Delete Secret

All stored secrets for Local Panorama

Choosing to ‘Load’ an environment makes that env available to pre-populate template fields. It will also be available as a ‘pop-over’ that you can use to copy and paste secrets into template fields. This is useful when you want to store secrets like API_KEYS

Note: Template variables that share the same ‘name’ as a secret in the currently loaded environment will be pre-populated with the value of that secret. You can find the exact name of a specific variable field by looking at the ‘.meta-cnc.yaml’ file for that form.

 AWS Demo-A paloalto ▼

As panhandler is a quickly evolving project with new features added frequently, it is advisable to ensure you update to the latest periodically.

8.1 Ensuring you have the latest docker image

Panhandler is primarily distributed as a docker image on Docker [Hub](#). To ensure you have the latest version, check for new releases [here](#). To launch a newer version via docker:

```
docker run -p 80:80 paloaltonetworks/panhandler -d
```

This will create a container based on the latest image tag. Versioned panhandler images are also available and can be found on Docker Hub.

Note: You must periodically pull new images from Docker hub to ensure you have the latest software with new features and bug fixes.

To ensure you have the most up to date software, perform a docker pull and specify the 'latest' tag.

```
docker pull paloaltonetworks/panhandler:latest
docker run -p 80:80 paloaltonetworks/panhandler:latest -d
```


CHAPTER 9

About

Panhandler is a tool to manage and share PAN-OS configuration sets. A configuration set can be a full device configuration, or a set of configuration elements. Panhandler allows you to import git repositories that contain one or more configuration templates. Each template contains a set of configuration elements and variables that can be customized for each deployment. Variables are presented in an auto-generated web form for an operator to complete. Once complete, the template is rendered and pushed to a PAN-OS device.

CHAPTER 10

Disclaimer

This software is provided without support, warranty, or guarantee. Use at your own risk.

CHAPTER 11

Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)