Forward-Looking Statements

This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at www.investors.splunk.com or the SEC's website at www.sec.gov. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

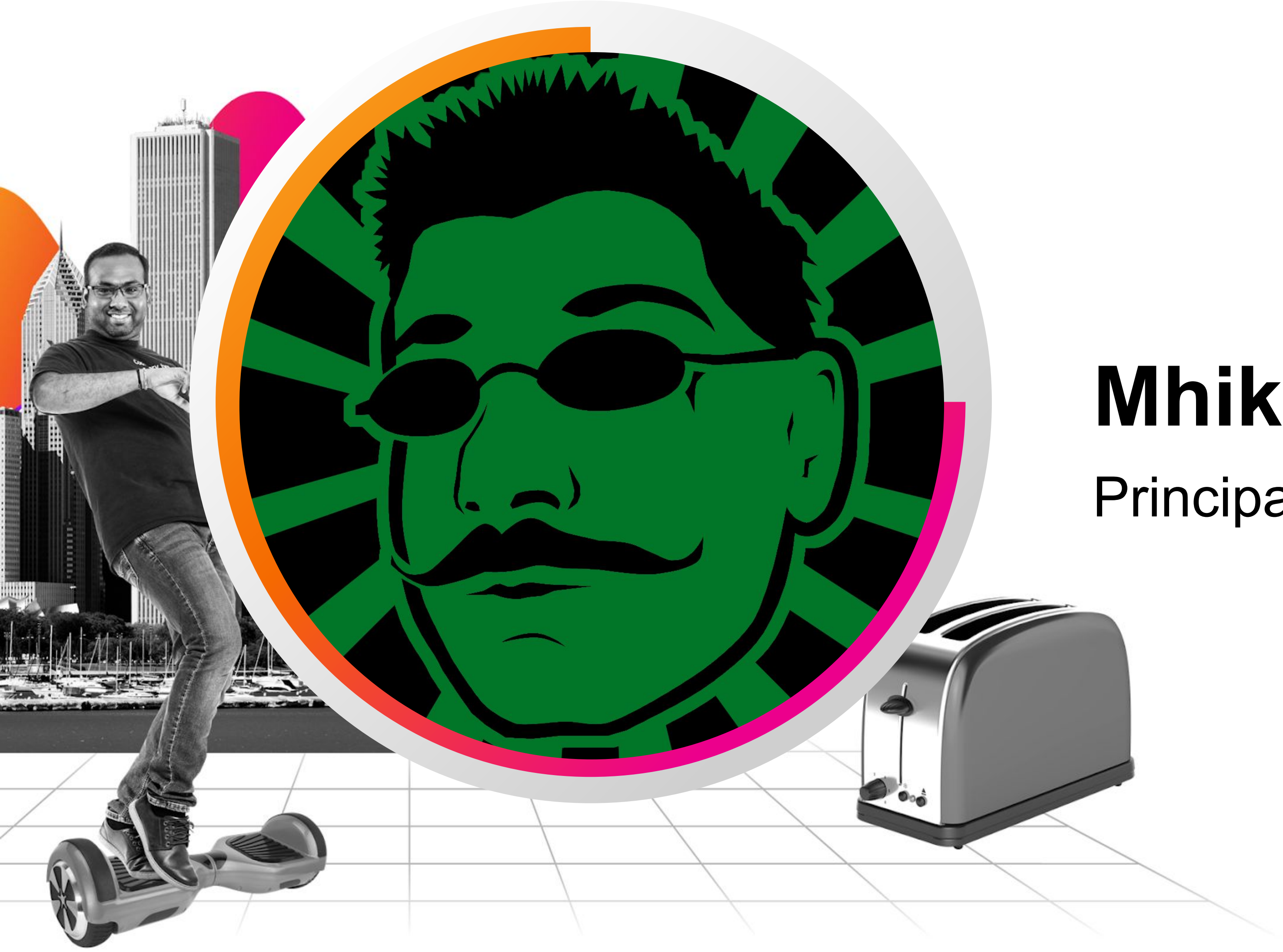
In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. We undertake no obligation either to develop the features or functionalities described, in beta or in preview (used interchangeably), or to include any such feature or functionality in a future release.

Building Smarter Playbooks Faster

SEC1216B

Mhike Funderburk

Principal Security Engineer | Stage 2 Security



Mhike Funderburk

Principal Security Engineer | S2 STAGE 2 SECURITY

Mhike Funderburk

Principal Security Engineer | S2 STAGE 2 SECURITY

Playbooks can be Challenging

Execution

Execution tooling is not granular enough

Timing

Waits, schedules, retries, and scope are inconsistent

Risky Clicks

Links are all or nothing

Content Management

Content migration is prone to human error

Custom Code

Code block consistency is hard to maintain

"Quality of Life" Apps

Making a difference with custom apps… mostly

Switchboard

Intelligent Playbook Execution

Runner

Manage Tailored Waits and Execution

Link

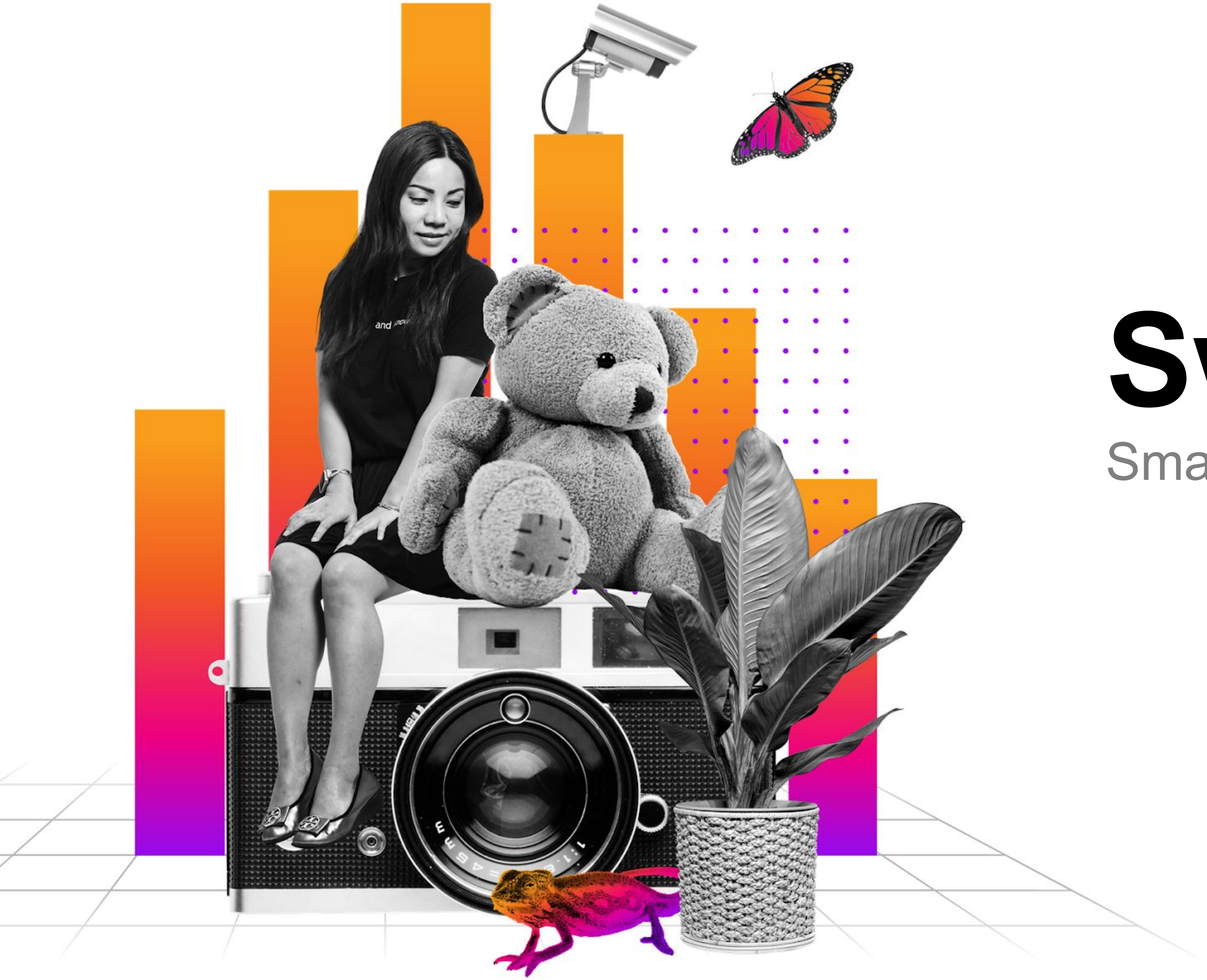
Create Easy, Visible, and Safe Links

Exodus

Migrate Content Between Systems With Approvals

Gettin Janky With It

Importing Custom Functions Into Code



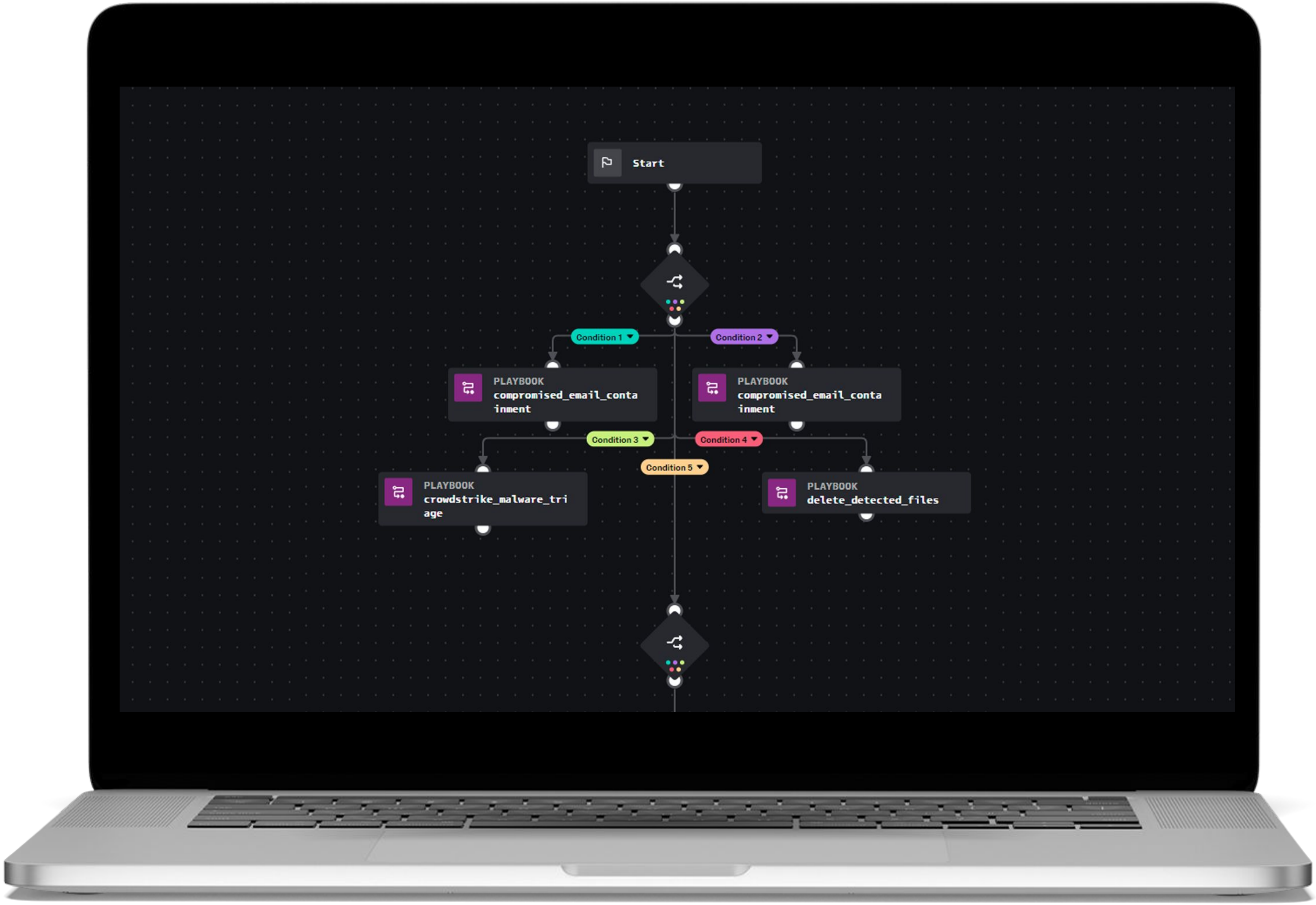
Switchboard

Smarter Execution

splunk> .conf22

Our First Attempt

- Scales poorly
- Difficult to manage
- Easy to break

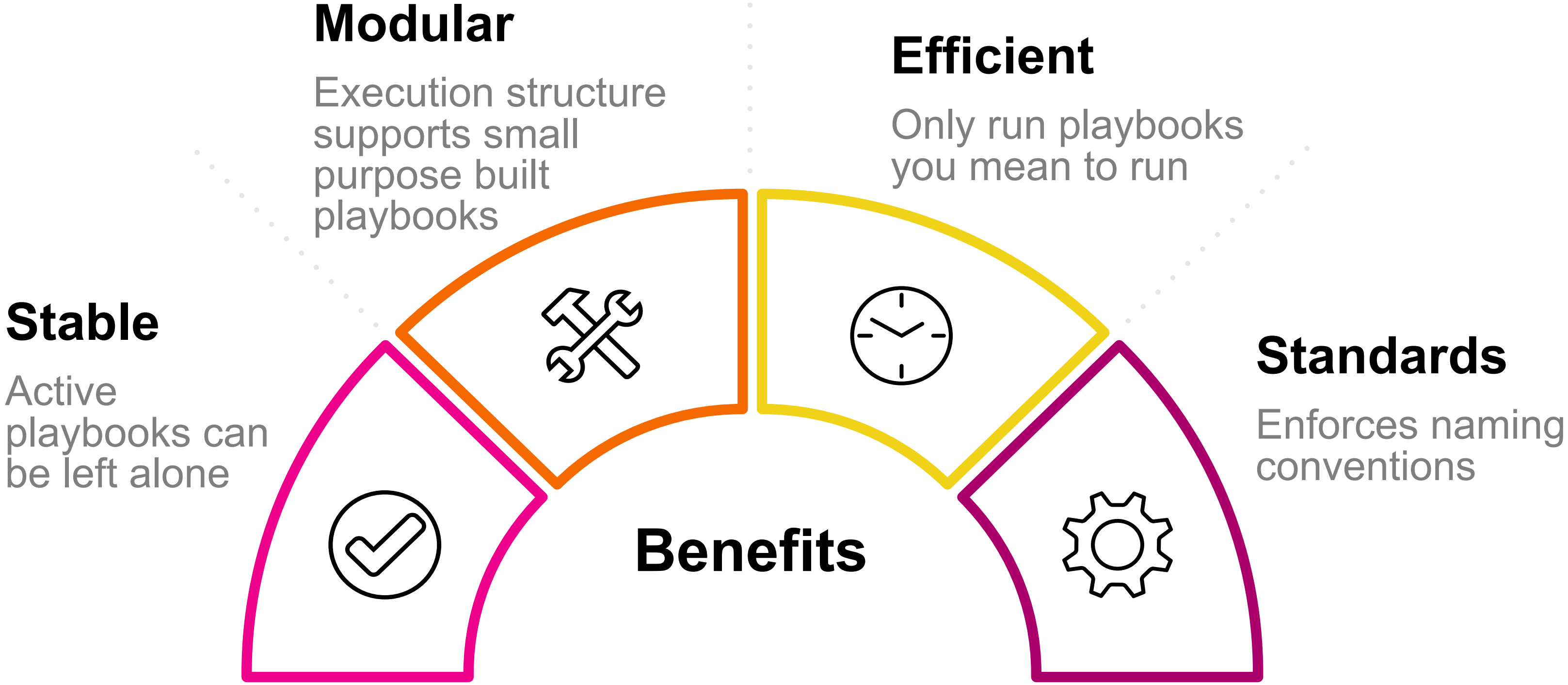


Switchboard

Running playbooks intentionally

Execute Playbooks By Type

- **Rule:** For matching correlation rule names
- **Product:** For matching product categories
- **Subject:** For matching substrings of rules for more generic matching
- **Field:** For matching when a given field exists



Stable

Active playbooks can be left alone

Modular

Execution structure supports small purpose built playbooks

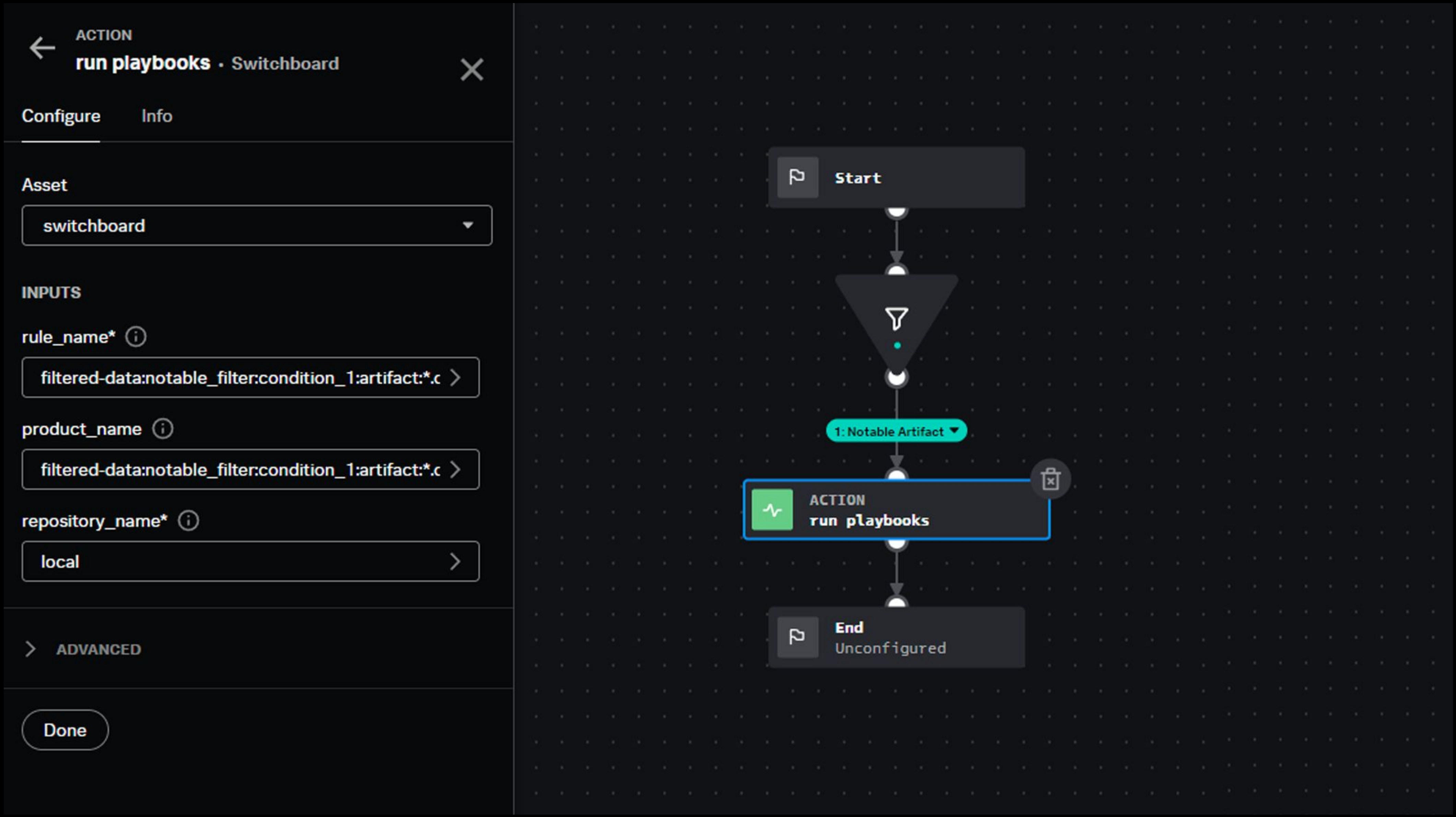
Efficient

Only run playbooks you mean to run

Standards

Enforces naming conventions

Benefits



ACTION

run playbooks · Switchboard

Configure | Info

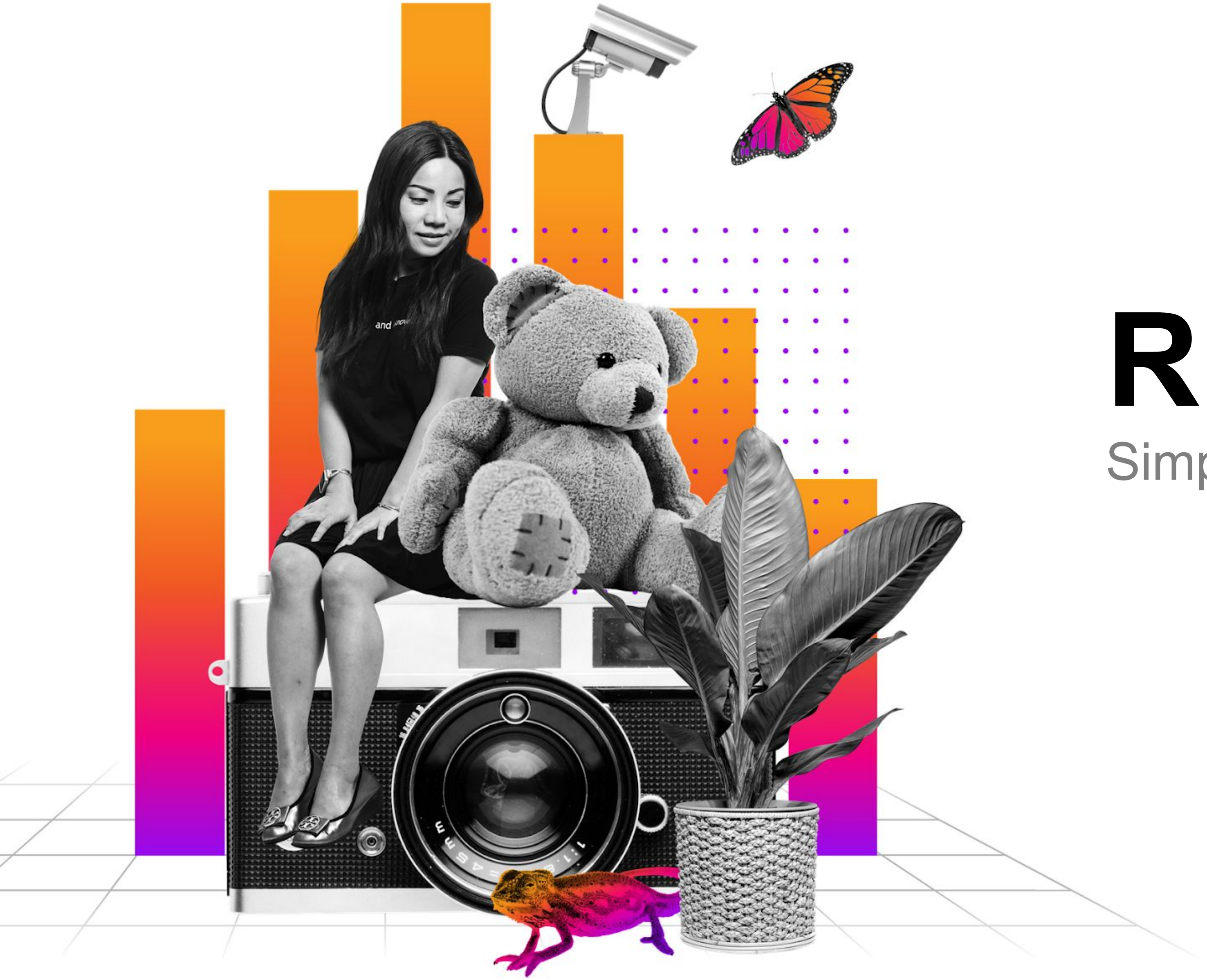
Asset: switchboard

INPUTS

rule_name*: filtered-data:notable_filter:condition_1:artifact:*.c

product_name: filtered-data:notable_filter:condition_1:artifact:*.c

repository_name*: local



Runner

Simplified Timing

splunk> .conf22

Runner

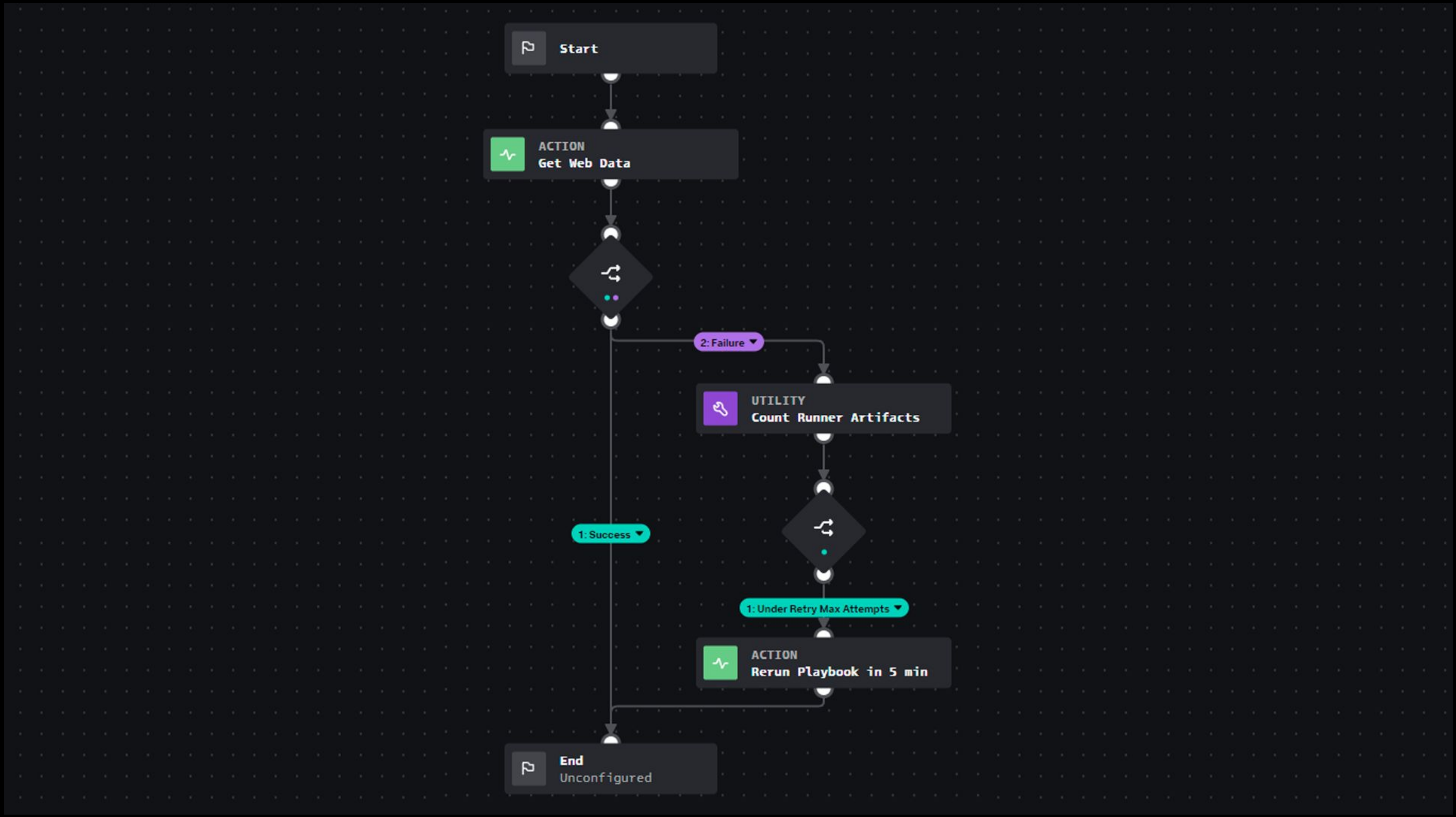
Delays, Retries,
and Artifact Scope
Made Easy

Delays and Retries

- Drops an artifact with state, reason, and time
- Executes with on poll so doesn’t distract from the normal playbook thread processing
- Can be static or random wait
- Number of playbooks executed per poll is configurable

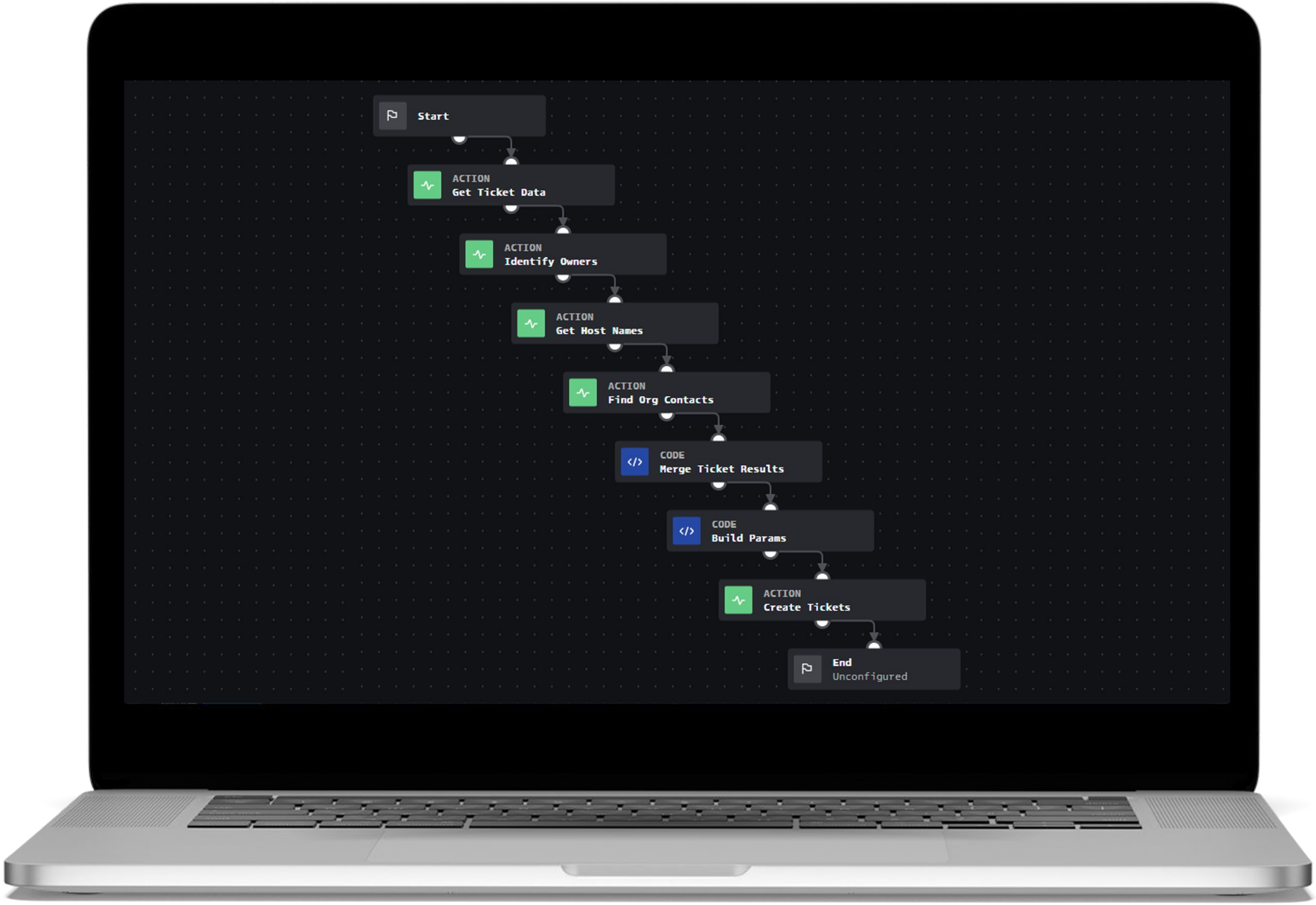
Artifact Scope

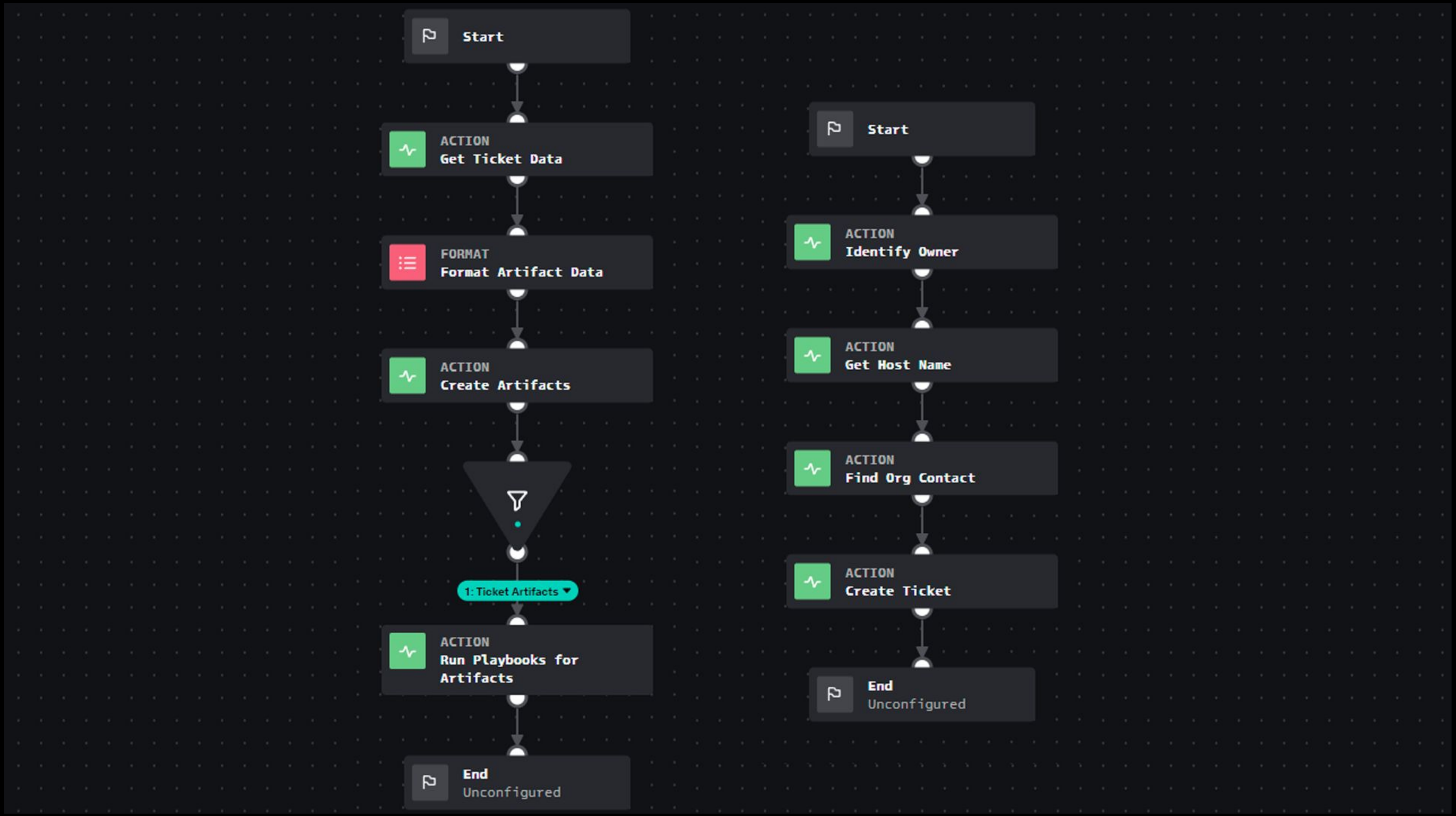
- Processes like ticket creation that require stitching data together can be difficult with multiple tickets
- Executing each artifact ticket with single artifact scope allows for a faster simpler playbook
- Less custom code to work through and maintain

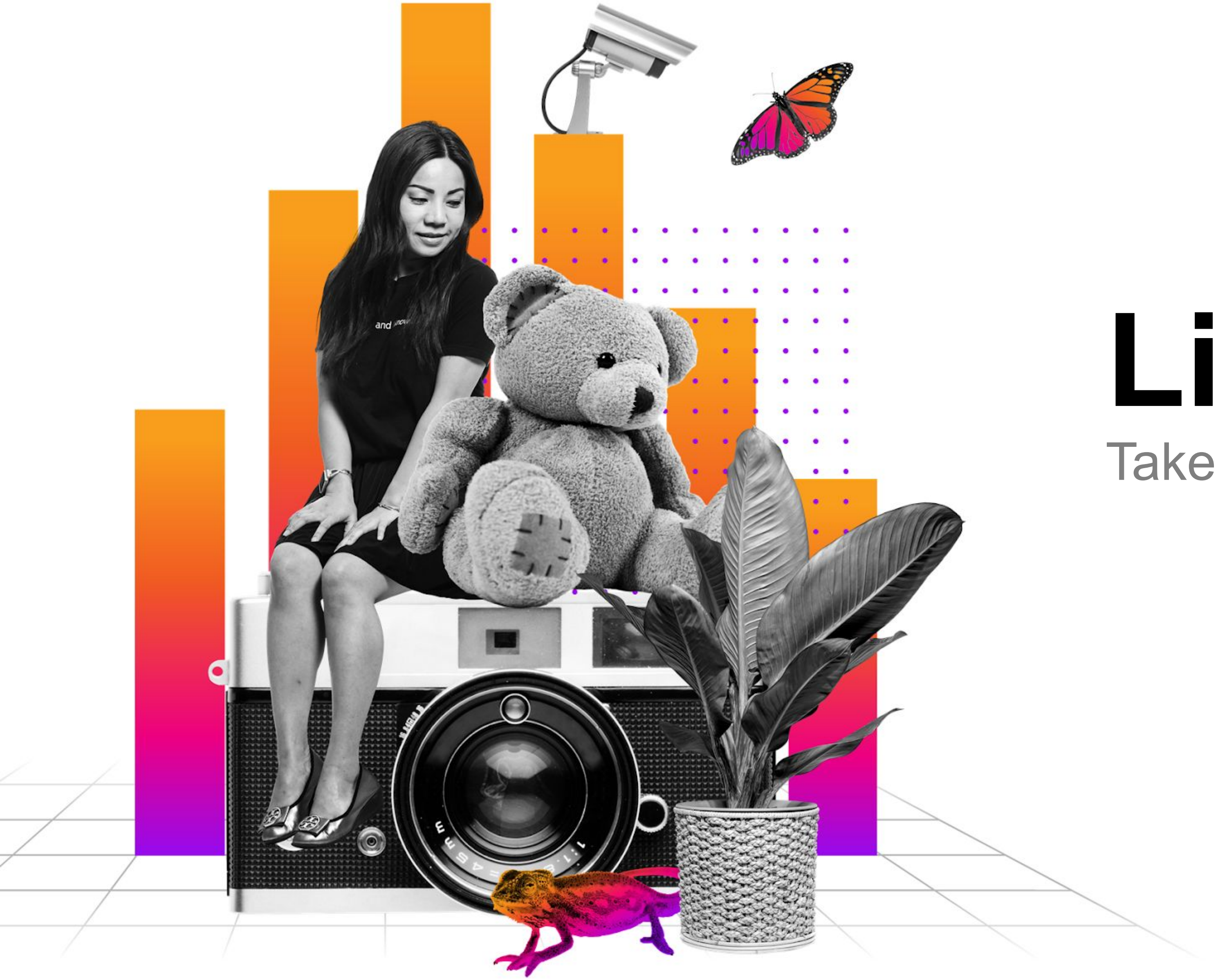

Start
ACTION
Get Web Data
2: Failure
UTILITY
Count Runner Artifacts
1: Success
1: Under Retry Max Attempts
ACTION
Rerun Playbook in 5 min
End
Unconfigured

Data Stitching

- Difficult to handle individual failures
- Code blocks are complex
- List order operations can be complicated to manage




Start
ACTION
Get Ticket Data
FORMAT
Format Artifact Data
ACTION
Create Artifacts
1: Ticket Artifacts
ACTION
Run Playbooks for Artifacts
End
Unconfigured
Start
ACTION
Identify Owner
ACTION
Get Host Name
ACTION
Find Org Contact
ACTION
Create Ticket
End
Unconfigured



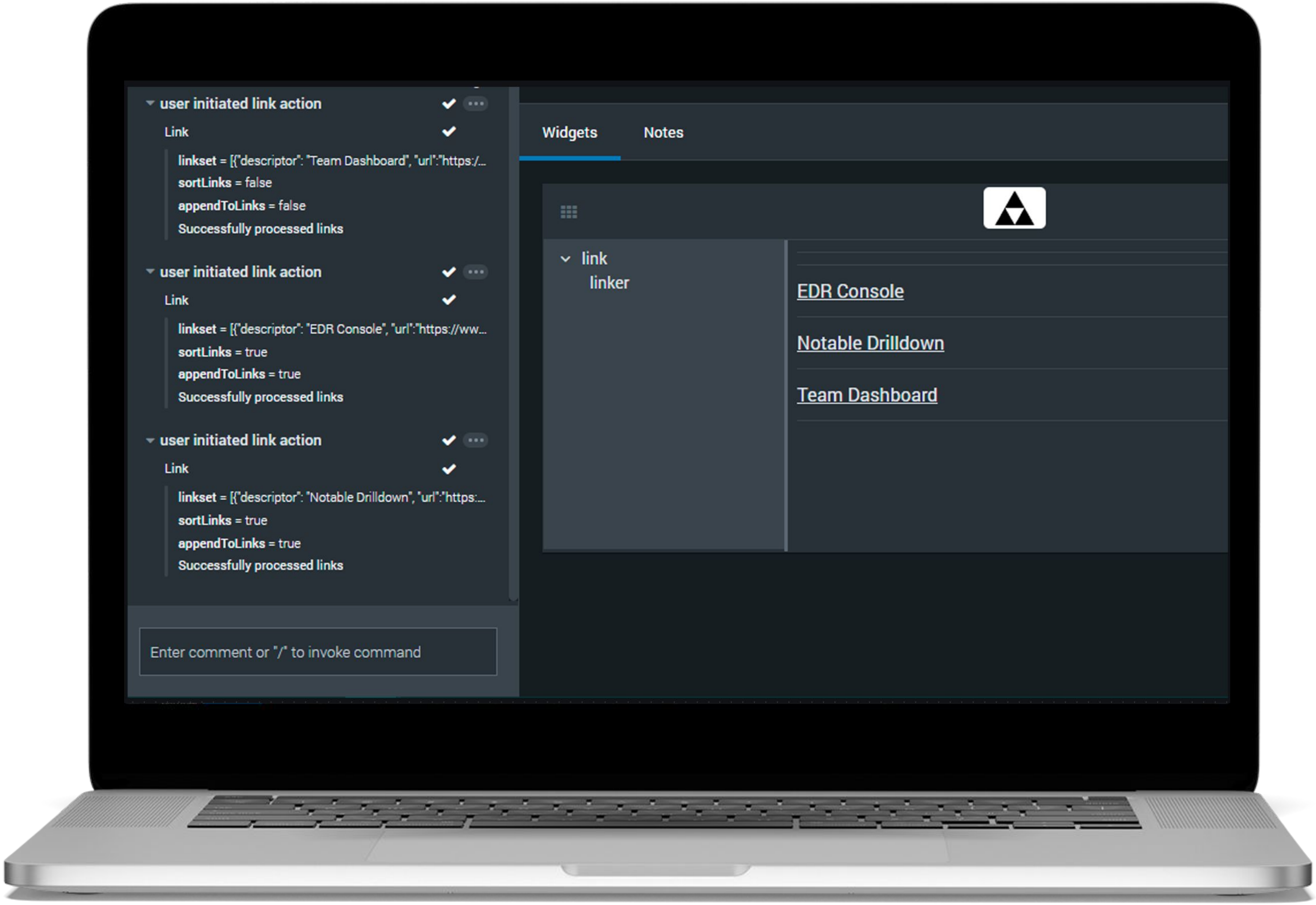
Link

Take the Risk Out of Clicks

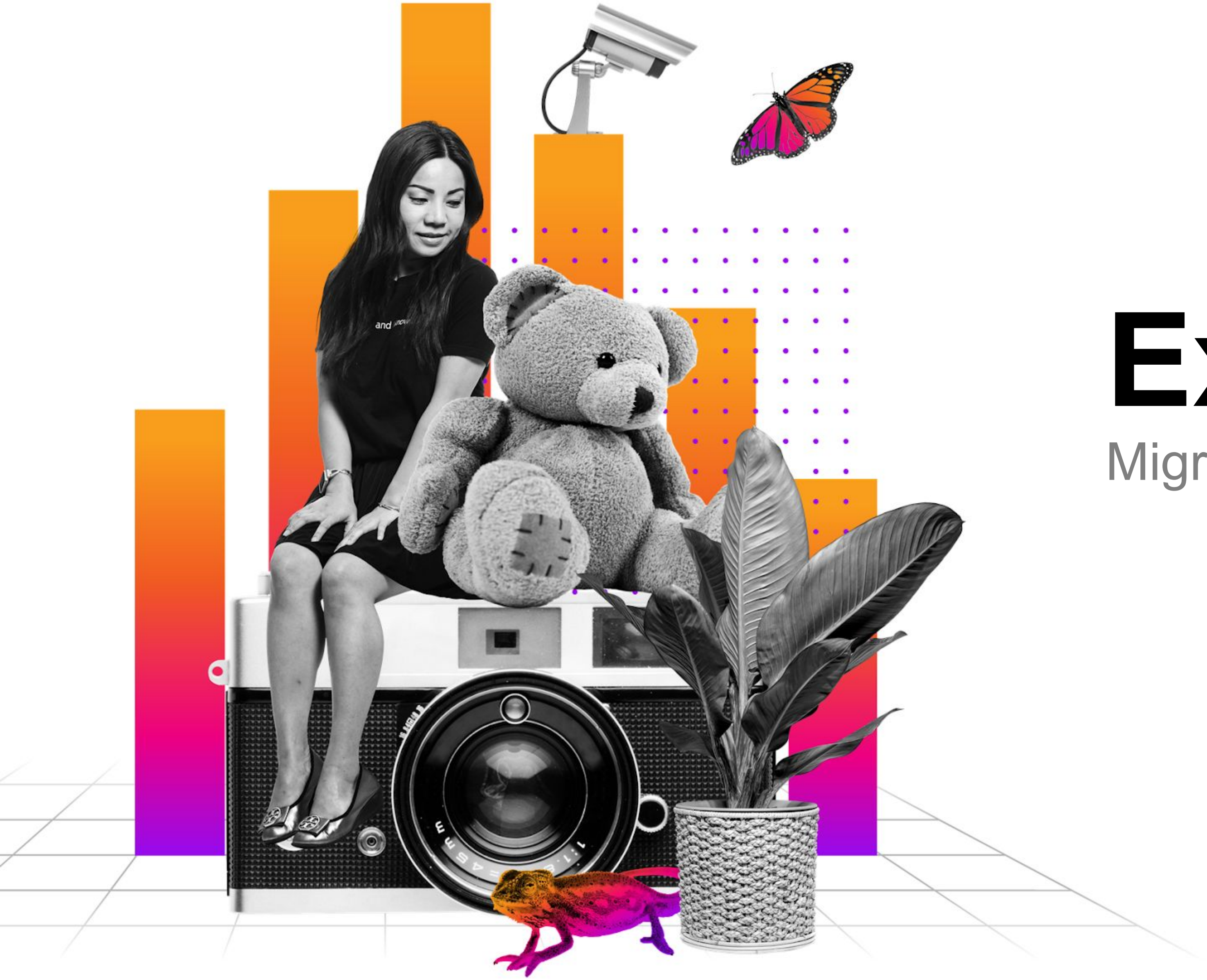
splunk> .conf22

Link Widget

- Clickable links are created in a widget
- Links can be appended and sorted from any playbook
- Links open in a new tab
- Widgets are shown by default so they are easy to find
- That risky admin setting can be switched off



```
user initiated link action
  Link
    linkset = [{"descriptor": "Team Dashboard", "url":"https:/...
    sortLinks = false
    appendToLinks = false
    Successfully processed links
user initiated link action
  Link
    linkset = [{"descriptor": "EDR Console", "url":"https://ww...
    sortLinks = true
    appendToLinks = true
    Successfully processed links
user initiated link action
  Link
    linkset = [{"descriptor": "Notable Drilldown", "url":"https:...
    sortLinks = true
    appendToLinks = true
    Successfully processed links
```



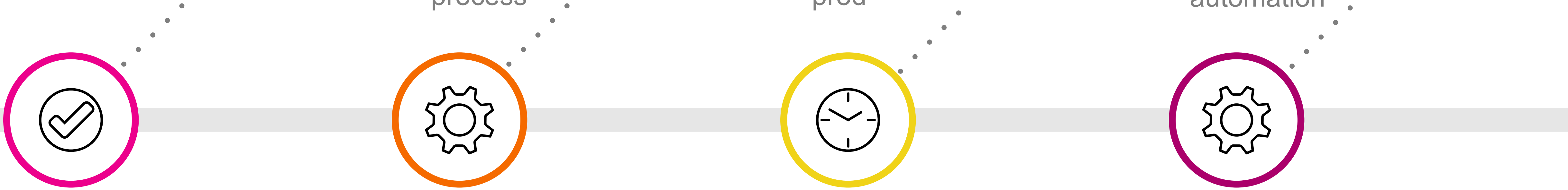
Exodus

Migrate Content Consistently

splunk> .conf22

Exodus Migration

Provide oversight and reduce human error



Enforces tagging and repo management

Playbook or Utility Created or Updated

In proper repo with prod tags applied

Provides permission controls to approval process

On Poll Creates Exodus Ticket

Exodus label is limited to approvers

Ticket contains name and ID of updated content

Requires oversight before code reaches prod

Dev Lead Approves Ticket

Via automation or manual intervention

Ideally after reviewing the material

Error prone operations handed off to automation

On Poll Copies Content to Prod

Fully Automated Using Rest API

Nothing is Perfect

Exodus is no exception

Pros

Migrates Playbooks

Migrates Utilities

Migrates New Assets

Adds Control Gateways

Removes Human Error

Creates Historical Record

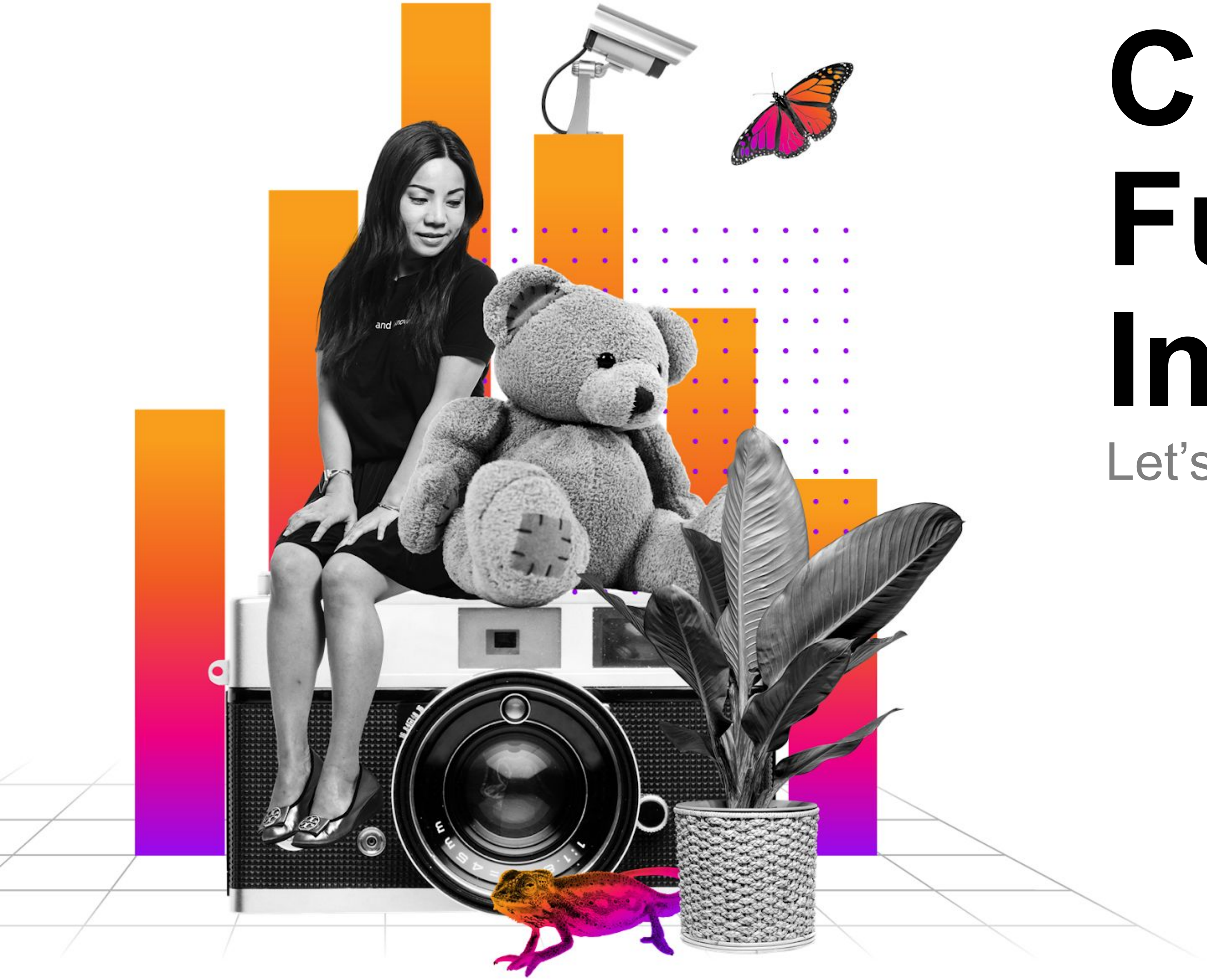
Cons

Requires HTTP access between hosts

Loses Tenant Assignments

Cannot Move Apps

Utilities Cannot be Tagged



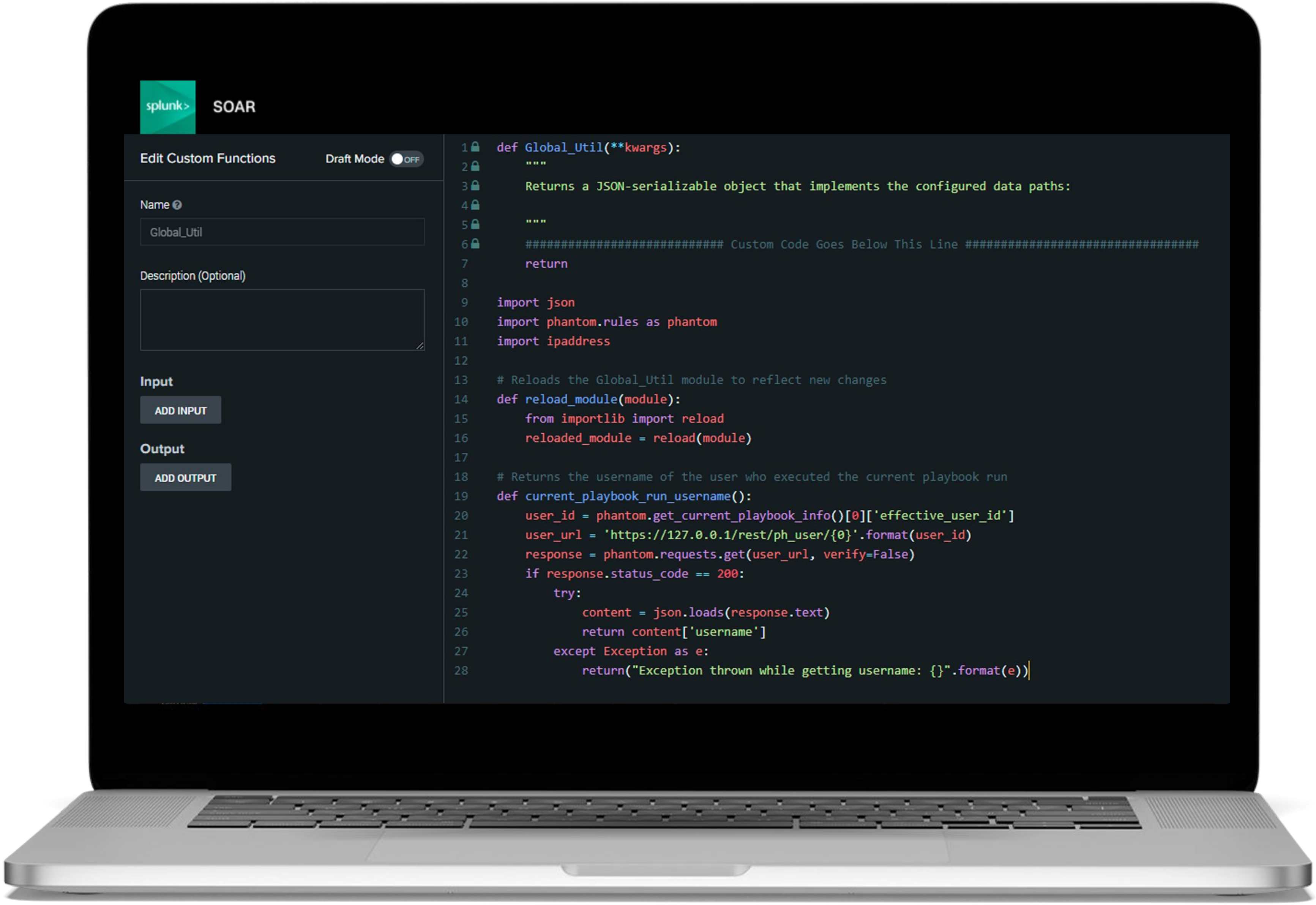
Custom Function Imports

Let’s get a little sketchy

splunk> .conf22

Hijacking a Utility Block

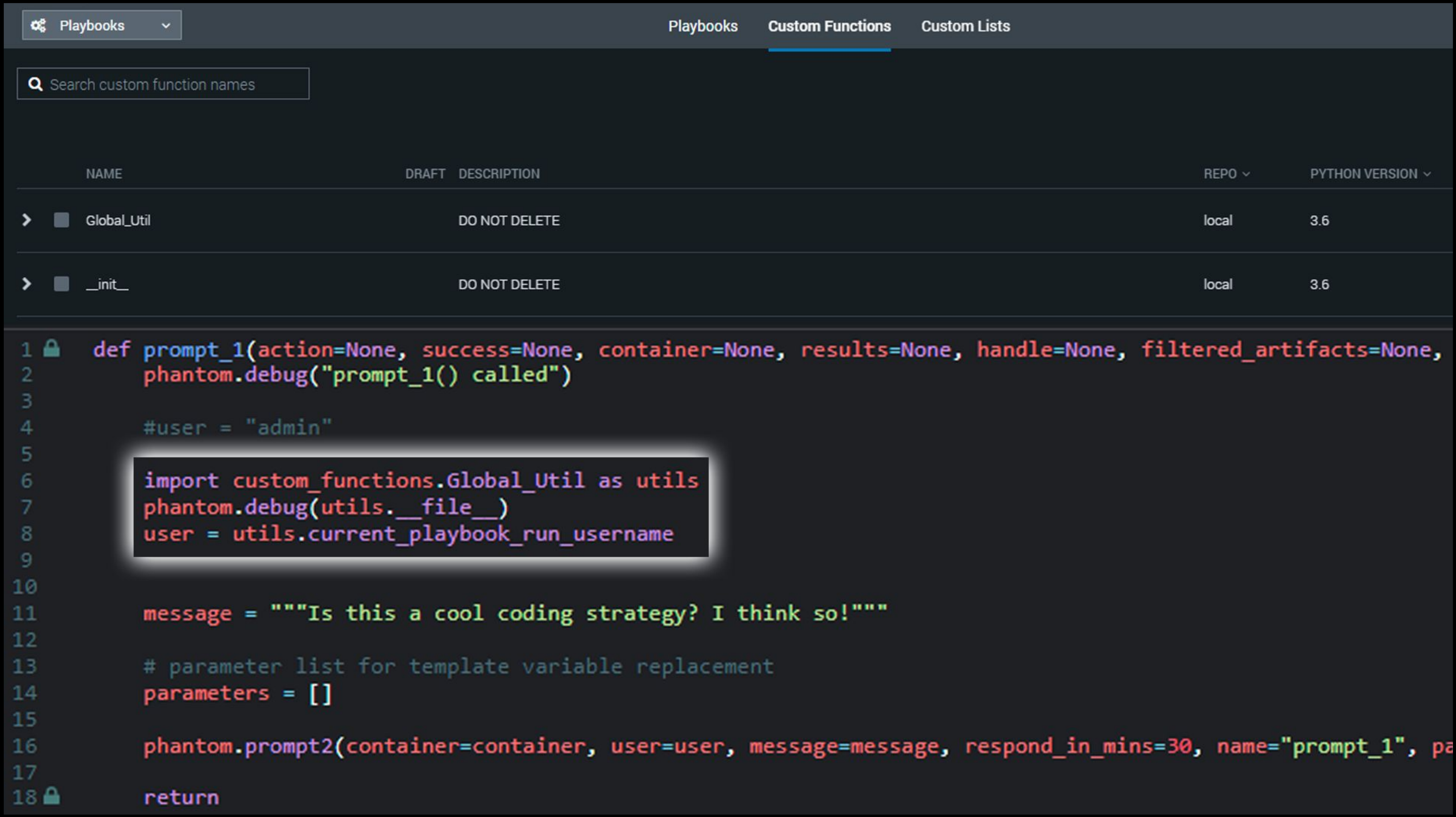
- Utilities live in the same directory as playbooks so it’s easy to import
- Requires a reload capability
- Requires an init function as well
- Required on a per repo basis (Don’t cross the streams)



Name: Global_Util

Description (Optional)

```python
def Global_Util(**kwargs):
    """
    Returns a JSON-serializable object that implements the configured data paths:

    """
    ############################ Custom Code Goes Below This Line #################################
    return

import json
import phantom.rules as phantom
import ipaddress

# Reloads the Global_Util module to reflect new changes
def reload_module(module):
    from importlib import reload
    reloaded_module = reload(module)

# Returns the username of the user who executed the current playbook run
def current_playbook_run_username():
    user_id = phantom.get_current_playbook_info()[0]['effective_user_id']
    user_url = 'https://127.0.0.1/rest/ph_user/{0}'.format(user_id)
    response = phantom.requests.get(user_url, verify=False)
    if response.status_code == 200:
        try:
            content = json.loads(response.text)
            return content['username']
        except Exception as e:
            return("Exception thrown while getting username: {}".format(e))
```

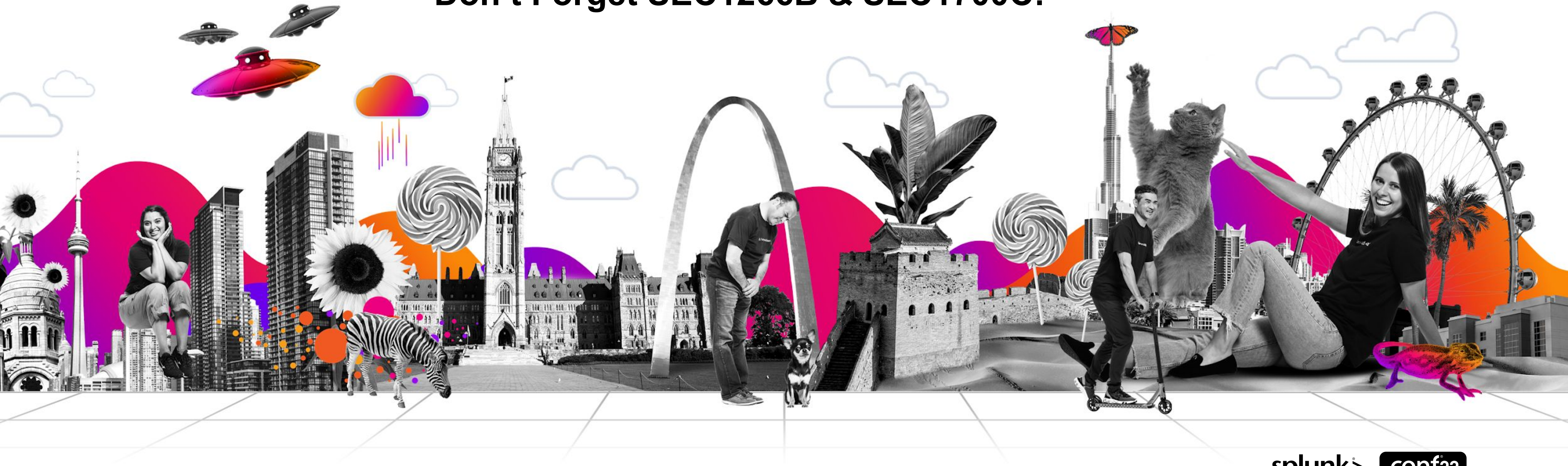


NAME	DRAFT	DESCRIPTION	REPO	PYTHON VERSION
Global_Util		DO NOT DELETE	local	3.6
__init__		DO NOT DELETE	local	3.6

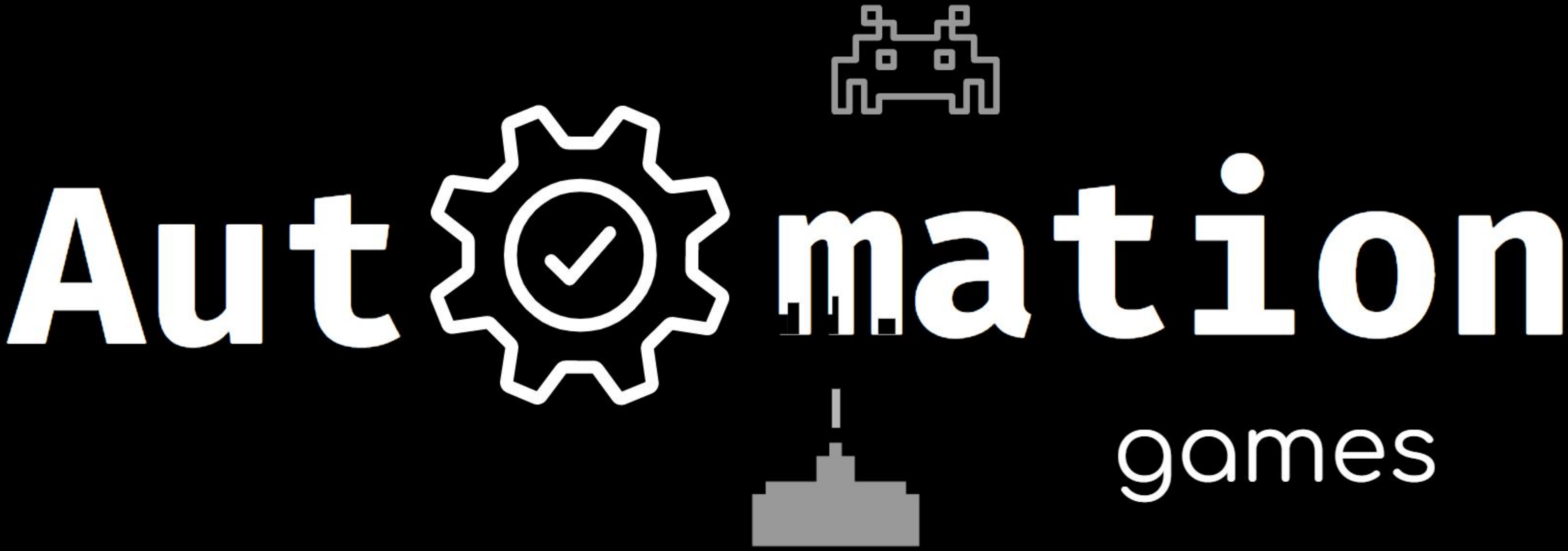
```python
def prompt_1(action=None, success=None, container=None, results=None, handle=None, filtered_artifacts=None,
    phantom.debug("prompt_1() called")

    #user = "admin"

    import custom_functions.Global_Util as utils
    phantom.debug(utils.__file__)
    user = utils.current_playbook_run_username


    message = """Is this a cool coding strategy? I think so!"""

    # parameter list for template variable replacement
    parameters = []

    phantom.prompt2(container=container, user=user, message=message, respond_in_mins=30, name="prompt_1", pa

    return
```

Thank You

Don’t Forget SEC1266B & SEC1700C!



splunk> .conf22



an interactive workshop on security automation

Sign up in the Session Scheduler or Mobile App

SEC1761 - The Automation Games
Thursday June 16 | 12-2pm PST