



# The Zen of Ansible

Tim Appnel

Sr. Product Manager, Red Hat Ansible

GitHub: [tima](#)

Matrix: [@tima:ansible.im](#)

ANSIBLE

# About me



10+ years of experience with Ansible as a contributor, customer, consultant, evangelist, product manager, and “jack of all trades.”

The synchronize module in Ansible is all my fault. (Sorry engineering)

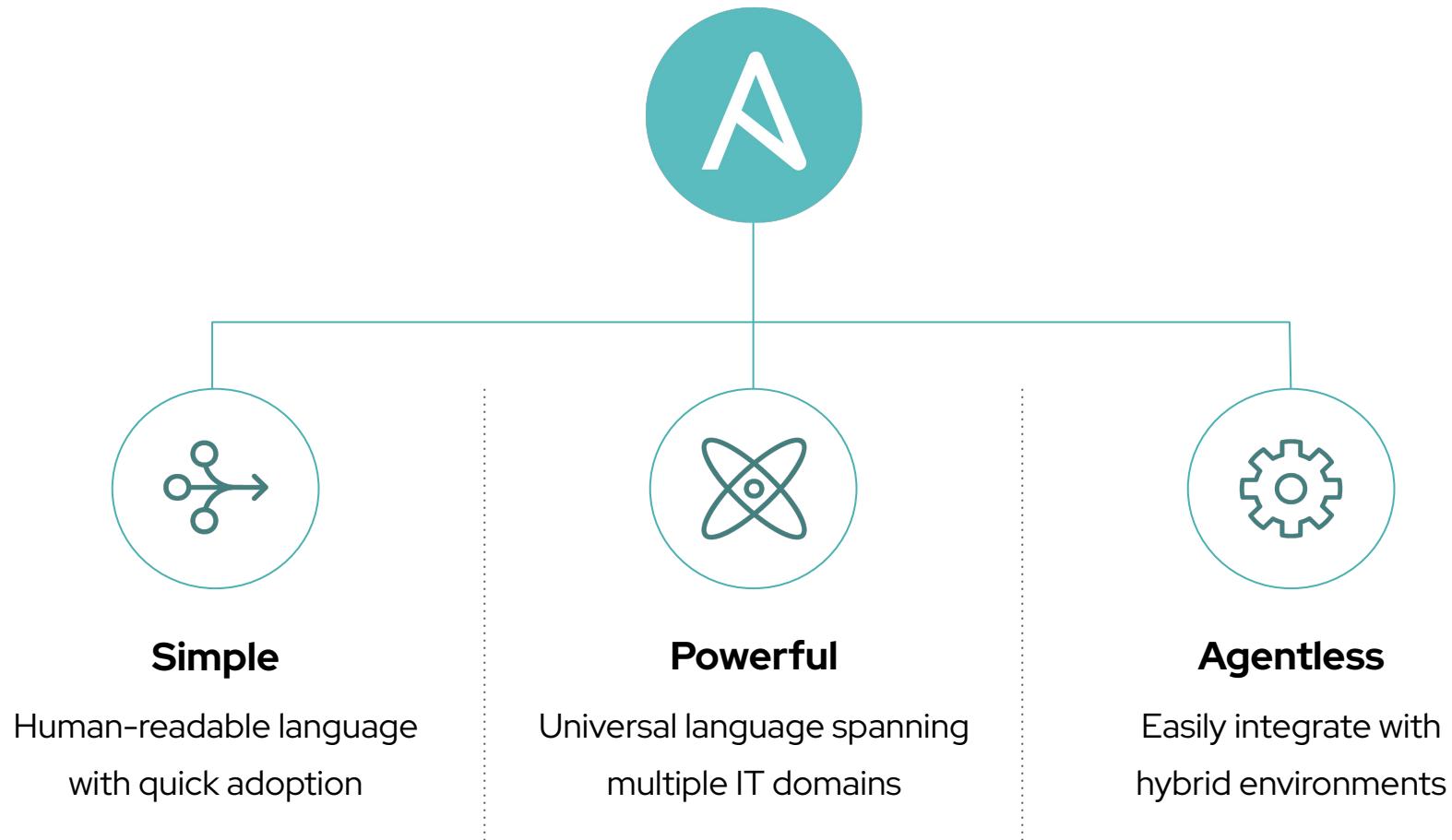
# About this talk



Spiritual successor  
of the “Ansible best  
practices” talk first  
presented at Red  
Hat Summit June  
2016 and later at  
dozens of events  
over the years

Inspired by “The Zen  
of Python” by Tim  
Peters

# The Ansible way

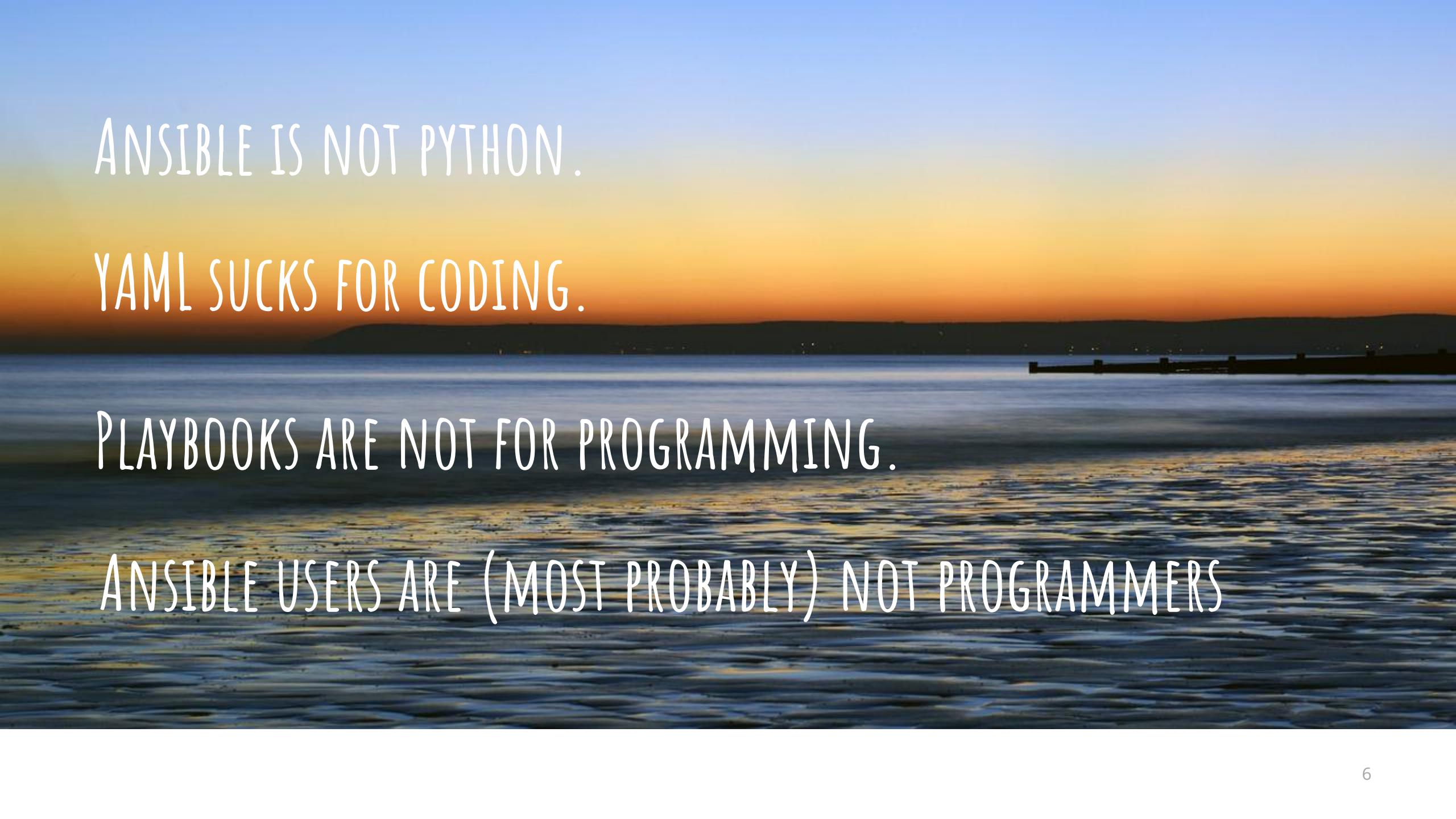


# About this talk



The Zen of Ansible applies to playbooks, roles and the interfaces and functional design of modules and plugins.

Apply The Zen of Python to how you write and develop your code.

The background of the slide is a photograph of a sunset or sunrise over a calm body of water. The sky is a gradient from blue at the top to orange and yellow near the horizon. In the distance, there are dark silhouettes of hills or land. The water in the foreground has small, golden reflections from the setting sun.

ANSIBLE IS NOT PYTHON.

YAML SUCKS FOR CODING.

PLAYBOOKS ARE NOT FOR PROGRAMMING.

ANSIBLE USERS ARE (MOST PROBABLY) NOT PROGRAMMERS



## Example: Overuse of command modules forcing programming

```
- hosts: all
  vars:
    cert_store: /etc/mycerts
    cert_name: my cert
  tasks:
    - name: check cert
      ansible.builtin.shell: certify --list --name={{ cert_name }} --cert_store={{ cert_store }} | grep "{{ cert_name }}"
      register: output

    - name: create cert
      ansible.builtin.command: certify --create --user=chris --name={{ cert_name }} --cert_store={{ cert_store }}
      when: output.stdout.find(cert_name) != -1
      register: output

    - name: sign cert
      ansible.builtin.command: certify --sign --name={{ cert_name }} --cert_store={{ cert_store }}
      when: output.stdout.find("created") != -1
```

## Example: Custom module to abstract user from programming in playbook

```
- hosts: all
  vars:
    cert_store: /etc/mycerts
    cert_name: my cert
  tasks:
    - name: create and sign cert
      umbrellacorp.nest.certify:
        state: present
        sign: yes
        user: chris
        name: "{{ cert_name }}"
        cert_store: "{{ cert_store }}"
```



## Example: External script and programming to pass parameters

```
- ansible.builtin.set_fact:  
  
    splitter_cmd: >  
        python3 /tmp/list_changed_targets.py  
        --branch {{ zuul.branch }}  
        {% if ansible_test_splitter_releases_to_test is defined %}  
            --ansible-releases {{ ansible_test_splitter_releases_to_test | join(' ') }}{% endif %}  
        {% if ansible_test_splitter_total_job is defined %}--total-job {{  
            ansible_test_splitter_total_job }}{% endif %}  
        {% if ansible_test_splitter_test_changed|bool %}--test-changed{% else %}--test-all-the-targets{%  
            endif %}  
        {{ ansible_test_splitter_check_for_changes_in | join(' ') }}
```



## Example: External script and building command line

```
- name: Evaluate security group rules
  ansible.builtin.command: >
    python {{ ansible_role_dir }}/files/validate_security_group_rules.py
    --dest_subnet_cidrs "{{ rds_subnets_cidrs }}"
    --dest_security_groups "{{ rds_security_groups.security_groups }}"
    --dest_port "{{ rds_instance_endpoint_port }}"
    --src_security_groups "{{ ec2_security_groups.security_groups }}"
    --src_private_ip "{{ ec2_private_ip_addrs | first }}
```



Example: Purpose built module in a collection that avoids programming in the playbook.

```
- name: Evaluate Security Group Rules
  cloud.aws_troubleshooting.validate_security_group_rules:
    dest_subnet_cidrs: "{{ rds_subnets_cidrs }}"
    dest_security_groups: "{{ rds_security_groups.security_groups }}"
    dest_port: "{{ rds_instance_endpoint_port }}"
    src_security_groups: "{{ ec2_security_groups.security_groups }}"
    src_private_ip: "{{ ec2_private_ip_addrs | first }}
```

CLEAR IS BETTER THAN CLUTTERED.

CONCISE IS BETTER THAN VERBOSE.

SIMPLE IS BETTER THAN COMPLEX.

READABILITY COUNTS.



## Example: Functional automation tasks using shorthand that is hard to read

```
- name: install telegraf
  ansible.builtin.yum: name=telegraf-{{ telegraf_version }} state=present update_cache=yes disable_gpg_check=yes enable
  notify: restart telegraf

- name: configure telegraf
  ansible.builtin.template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  ansible.builtin.service: name=telegraf state=started enabled=yes
```

## Example: Functional and more readable Ansible automation

```
- name: install telegraf
  ansible.builtin.yum:
    name: telegraf-{{ telegraf_version }}
    state: present
    update_cache: yes
    disable_gpg_check: yes
    enablerepo: telegraf
    notify: restart telegraf

- name: configure telegraf
  ansible.builtin.template:
    src: telegraf.conf.j2
    dest: /etc/telegraf/telegraf.conf
    notify: restart telegraf

- name: start telegraf
  ansible.builtin.service:
    name: telegraf
    state: started
    enabled: yes
```

HELPING USERS GET THINGS DONE  
MATTERS MOST.

USER EXPERIENCE > IDEOLOGICAL  
PURITY.



## Example: User experience beating ideological purity

From "Ansible Best Practices: Roles & Modules"  
(circa 2018):

- [Modules] abstract users from having to know the details to get things done
- [Modules] are **not** one-to-one mapping of an API or command line tool interface
  - This is why you should not auto-generate your modules
- Keep parameters focused and narrowly defined
  - refrain from parameters that take complex data structures

```
- name: Create Service object with inline definition
  kubernetes.core.k8s:
    state: present
    definition:
      apiVersion: v1
      kind: Service
      metadata:
        name: web
        namespace: testing
        labels:
          app: galaxy
          service: web
      spec:
        selector:
          app: galaxy
          service: web
        ports:
          - protocol: TCP
            targetPort: 8000
            name: port-8000-tcp
            port: 8000
```

A wide-angle photograph of a winter landscape. In the foreground, a dark, frozen body of water reflects the surrounding scene. A small, bright yellow-orange tree stands out against the white snow on the left. On the right, a dense forest of evergreen trees is partially illuminated by warm sunlight, while the rest of the forest and the hillside behind it are in deep shadow. The sky is a clear, pale blue.

"MAGIC" CONQUERS THE MANUAL.

A photograph of a stack of six smooth, rounded stones of various sizes and colors (brown, tan, grey) balanced on a single dark rock. The stones are arranged in a slight curve. They are positioned on the left side of the frame, resting on a light-colored, textured rock. The background is a close-up of dark blue water with fine, dark ripples. The lighting creates highlights on the stones and the water's surface.

WHEN GIVING USERS OPTIONS,  
ALWAYS USE CONVENTION OVER  
CONFIGURATION.

## Example: Providing common values and reasonable defaults for users

### EXHIBIT A

```
# defaults_no_playbook.yml
---
- hosts: webservers
  roles:
    - role: apache_simple
      apache_http_port: 80
      apache_doc_root: /var/www/html
      apache_user: apache
      apache_group: apache
    - role: apache_simple
      apache_http_port: 8080
      apache_doc_root: /www/example.com
      apache_user: apache
      apache_group: apache
```

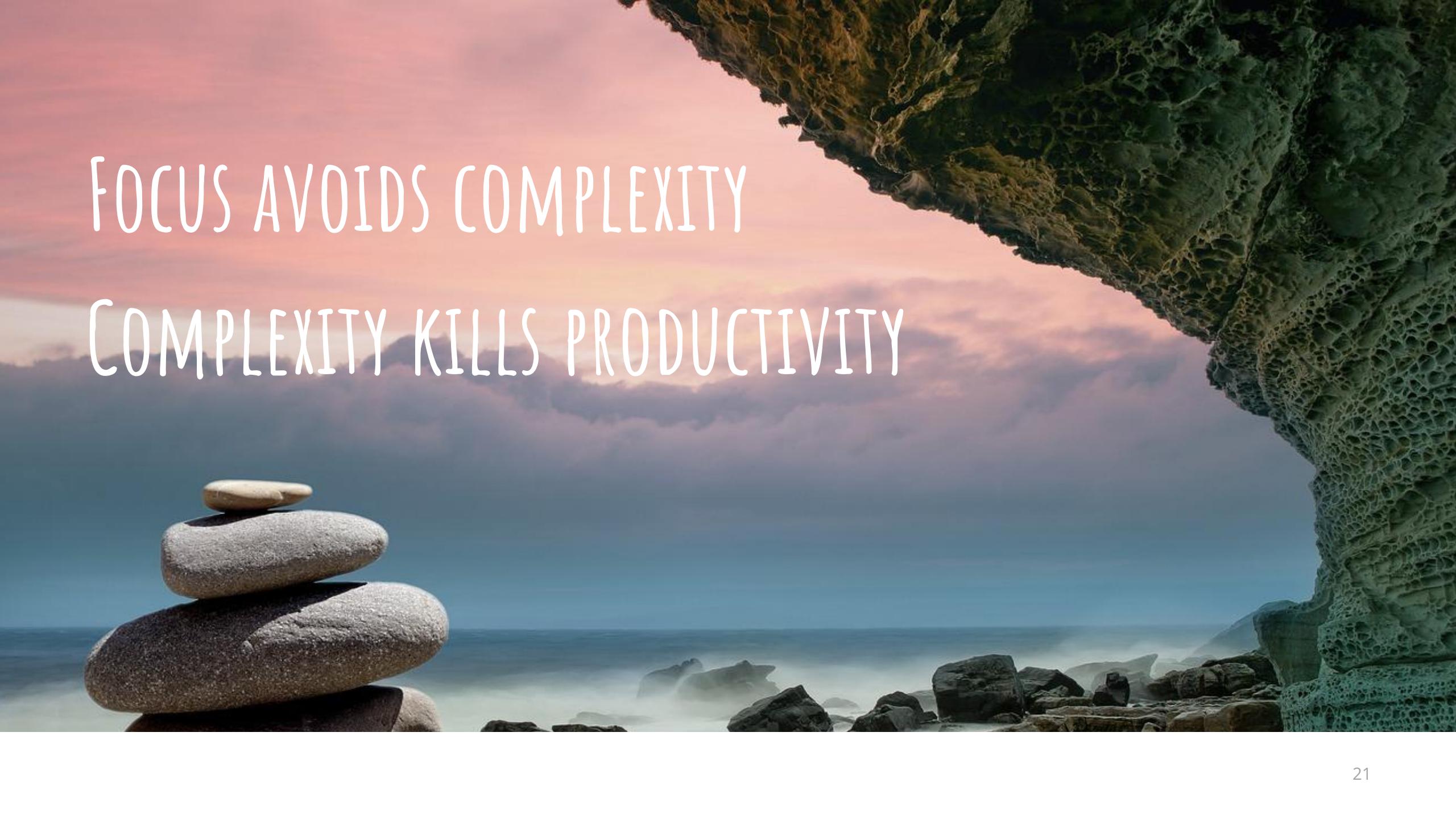
### EXHIBIT B

```
# defaults_yes_playbook.yml
---
- hosts: webservers
  roles:
    - role: apache_simple
      apache_http_port: 8080
      apache_doc_root: /www/example.com
      apache_user: apache
      apache_group: apache

# default/main.yml
---
apache_http_port: 80
apache_doc_root: /var/www/html
apache_user: apache
apache_group: apache
```

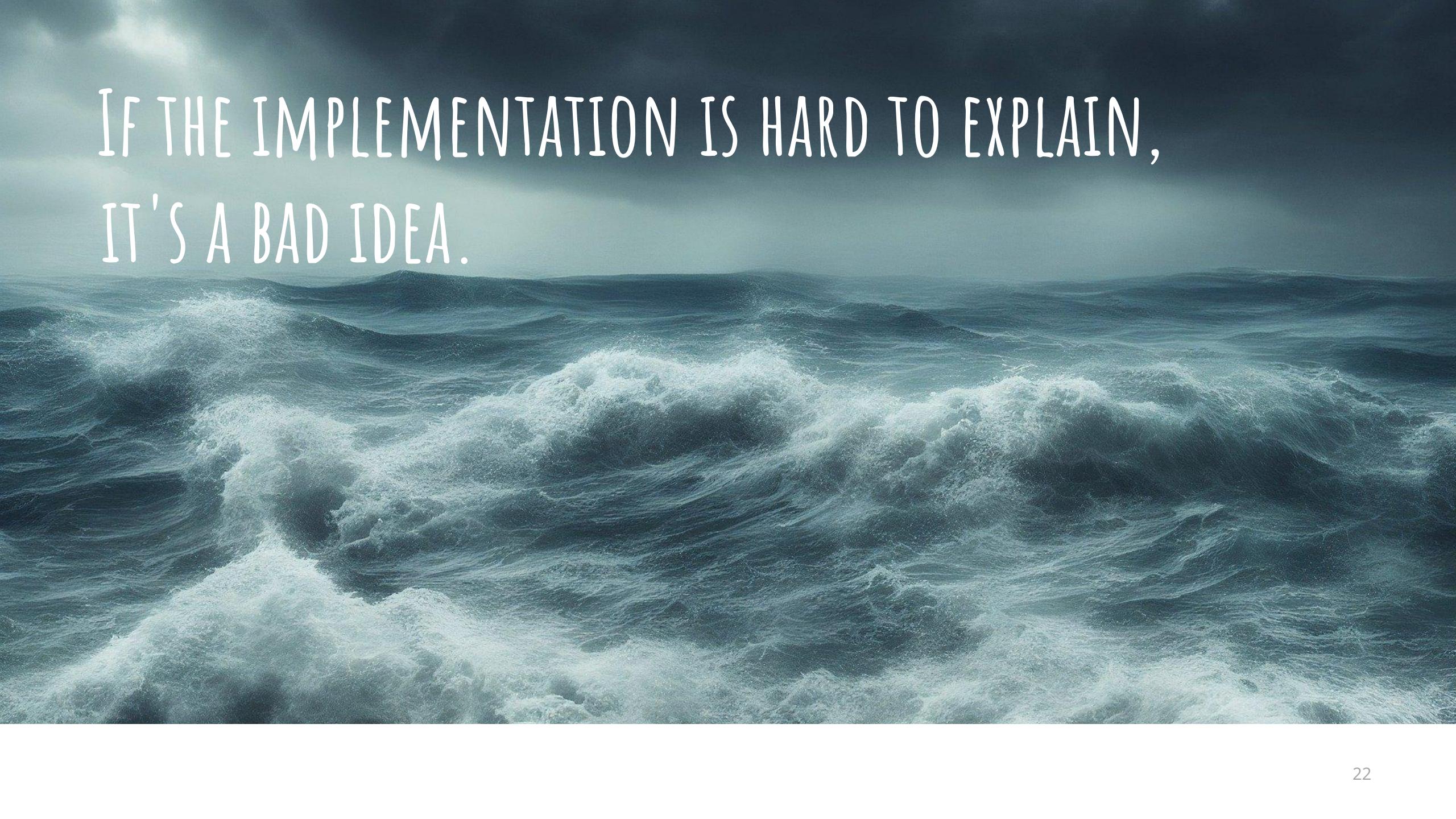
DECLARATIVE IS ALWAYS BETTER THAN IMPERATIVE.

MOST OF THE TIME.

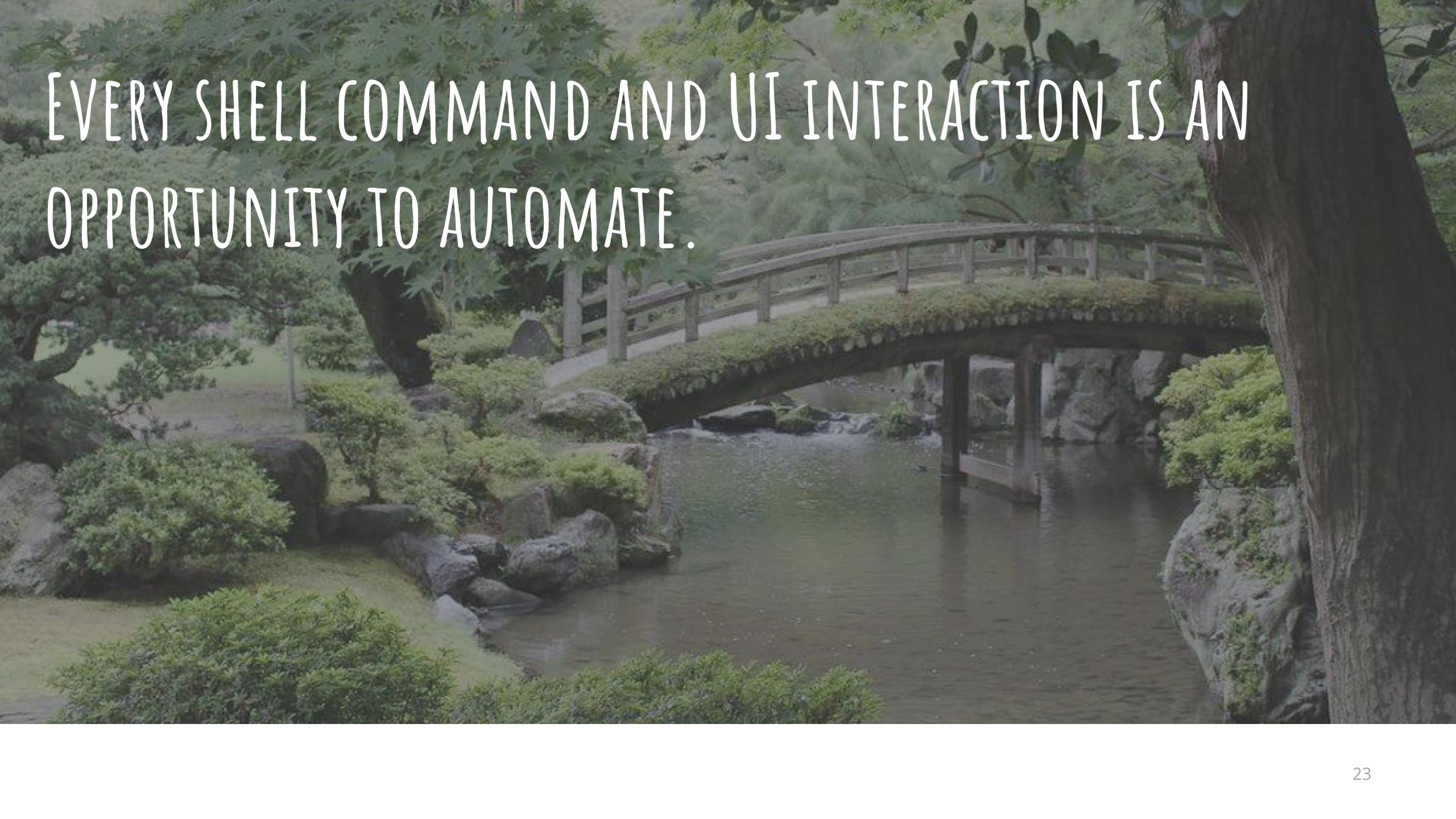


FOCUS AVOIDS COMPLEXITY

COMPLEXITY KILLS PRODUCTIVITY

The background of the slide is a photograph of a turbulent ocean. The water is a deep, dark teal or slate blue, with numerous white-capped waves crashing and churning across the frame. The horizon is visible in the distance, where the dark water meets a sky filled with heavy, grey clouds. The overall mood is one of chaos and difficulty.

IF THE IMPLEMENTATION IS HARD TO EXPLAIN,  
IT'S A BAD IDEA.

A photograph of a traditional Japanese garden. In the center, a curved stone bridge with a wooden railing spans a pond. The garden is lush with green trees, shrubs, and rocks. A large tree trunk is visible on the right side.

EVERY SHELL COMMAND AND UI INTERACTION IS AN  
OPPORTUNITY TO AUTOMATE.

## Installing Shipwright

The first step towards being able to leverage Shipwright to act as the facilitator for building execution environments is to install it into a Kubernetes environment.

Shipwright is available as an operator in [operatorhub.io](#) along with OperatorHub for installation in OpenShift. Execute the following command to deploy the operator to the cluster:

```
kubectl apply -f resources/operator/olm
```

Confirm the successful installation of the operator by checking the state of the ShipwrightBuild CustomResourceDefinition.

```
kubectl wait --for condition=established  
crd/shipwrightbuilds.operator.shipwright.io
```

Next, create a new namespace called shipwright-build and add the ShipwrightBuild custom resource, which will deploy the Shipwright build controller.

```
kubectl apply -f resources/operator/instance
```

Confirm the controller is running in the shipwright-build namespace:

```
kubectl get pods -n shipwright-build
```

A photograph of several smooth stones balanced on a sandy beach against a blue sky. In the foreground, there are two main stacks of stones. On the left, a stack of reddish-brown stones is balanced. On the right, a stack of dark grey/black stones is balanced. The background shows a sandy beach leading to a calm sea under a clear blue sky.

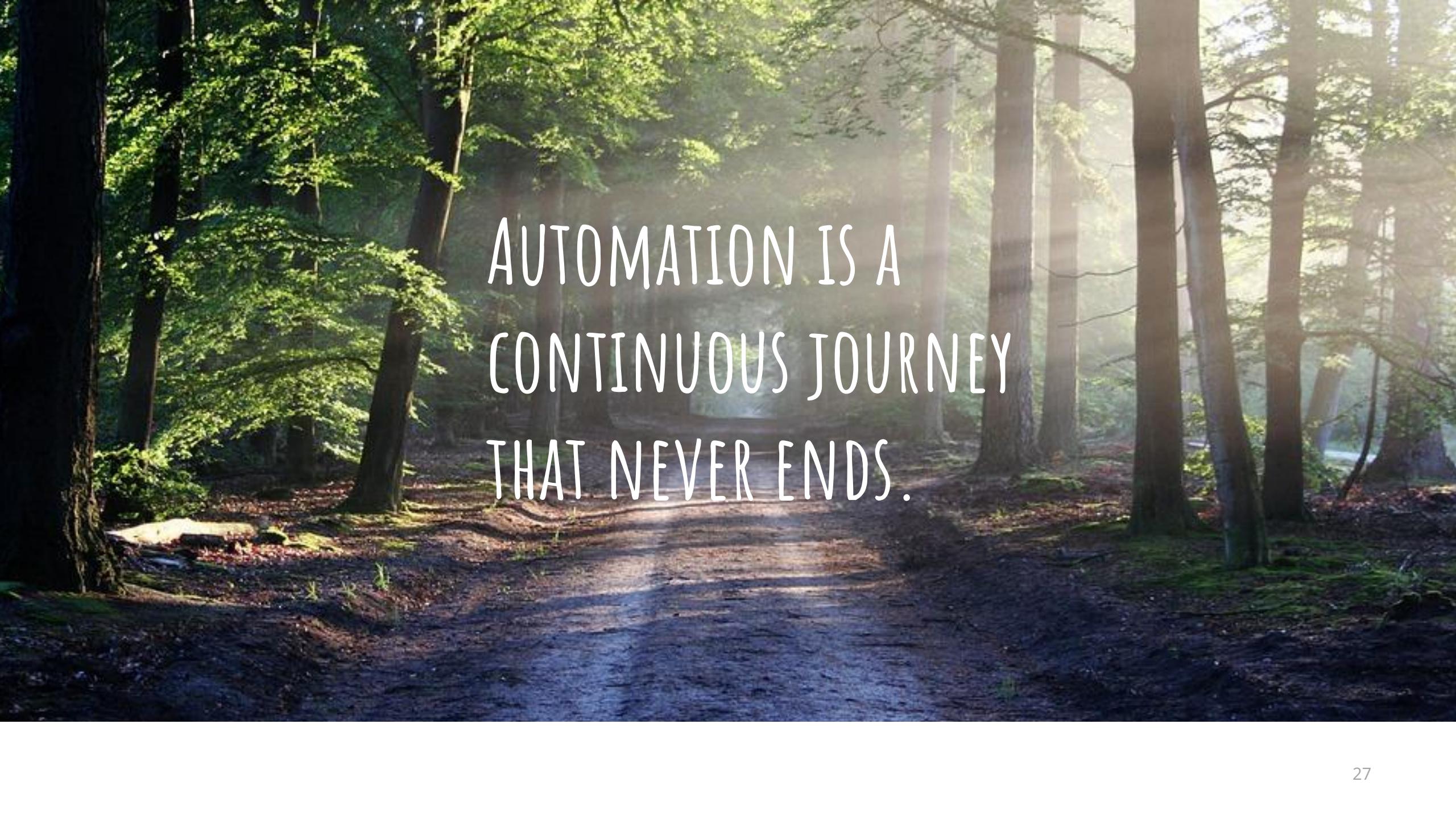
JUST BECAUSE SOMETHING WORKS, DOESN'T MEAN  
IT CAN'T BE IMPROVED.

FRICITION SHOULD BE ELIMINATED WHENEVER  
POSSIBLE.

## Example: Continuously improving by removing friction and clutter for users

```
# Create a Deployment reading a definition template from the Ansible controller local file
- name: My testing Deployment exists
  kubernetes.core.k8s:
    state: present
    definition: "{{ lookup('template', '/testing/deployment.j2') | from_yaml }}"

# Read definition template file from the Ansible controller file system
- name: My testing Deployment exists
  kubernetes.core.k8s:
    state: present
    template: '/testing/deployment.j2'
```

A photograph of a forest path. Sunlight filters through tall, thin trees, creating bright rays and dappled light on the dark, mossy ground. The path leads into the distance, suggesting a journey.

AUTOMATION IS A  
CONTINUOUS JOURNEY  
THAT NEVER ENDS.

## More Reading & Resources

- Good Practices for Ansible

- ↳ <https://redhat-cop.github.io/automation-good-practices/>

- Ansible Lint Documentation

- ↳ <https://ansible-lint.readthedocs.io/>

- PEP 20 – The Zen of Python

- ↳ <https://peps.python.org/pep-0020/>

- The Zen of Python, Explained

- ↳ <https://inventwithpython.com/blog/2018/08/17/the-zen-of-python-explained/>



# Thanks!

GitHub: <https://github.com/ansible/community>

Matrix: #community:ansible.com or #social:ansible.com

ANSIBLE