

Automated Testing of Managed Applications

MacSysAdmin Conference 2024

K Zehnder and G Pugh

Introductions



Introductions



Graham Pugh
Jamf

@MSA 2024

Graham



Graham




- 🇬🇧 (🇬🇧 🇨🇼) → 🇺🇸 → 🇨🇭 → 🇩🇪
- Senior Systems Engineer
- 2024-... @ Jamf MSP Services
- 2017-2023 @ ETH Zurich
- If I accidentally say "We" = Apple@ETH Team

Graham

Kati

Kati

- 
- Senior Systems Engineer
- In Education since 2004
- 2008-... @ ETH Zurich



Kati


Automated Testing of Managed Applications

As Part of a Fully Automated Software Deployment Workflow

INCOMING CALL

from your iPhone




Remind Me


Message



Decline


Accept

INCOMING CALL

from your iPhone




Remind Me


Message


Decline


Accept

Automated Testing of Managed Applications

As Part of a Fully Automated Software Deployment Workflow

Automated Testing of Managed Applications

Automated Testing of Managed Applications

- Software deployment workflow at ETH Zurich

Automated Testing of Managed Applications

- Software deployment workflow at ETH Zurich
- Testing Software - Why?

Automated Testing of Managed Applications

- Software deployment workflow at ETH Zurich
- Testing Software - Why?
- The Problems of Testing...

Automated Testing of Managed Applications

- Software deployment workflow at ETH Zurich
- Testing Software - Why?
- The Problems of Testing...
- Automated Software Testing

Automated Testing of Managed Applications

- Software deployment workflow at ETH Zurich
- Testing Software - Why?
- The Problems of Testing...
- Automated Software Testing
- What's Next / Conclusions

Software deployment workflow at ETH Zurich





AutoPkg everything

Graham R Pugh
Senior Client Engineer, IT Services
ETH Zurich
MacSysAdmin Virtual Conference, October 2021

Software Deployment

Software Deployment

- Check for a new version of a software title

Software Deployment

- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary

Software Deployment

- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary
- Upload package to Jamf Pro
- Create or update policies and smart groups targeted at testers

Software Deployment

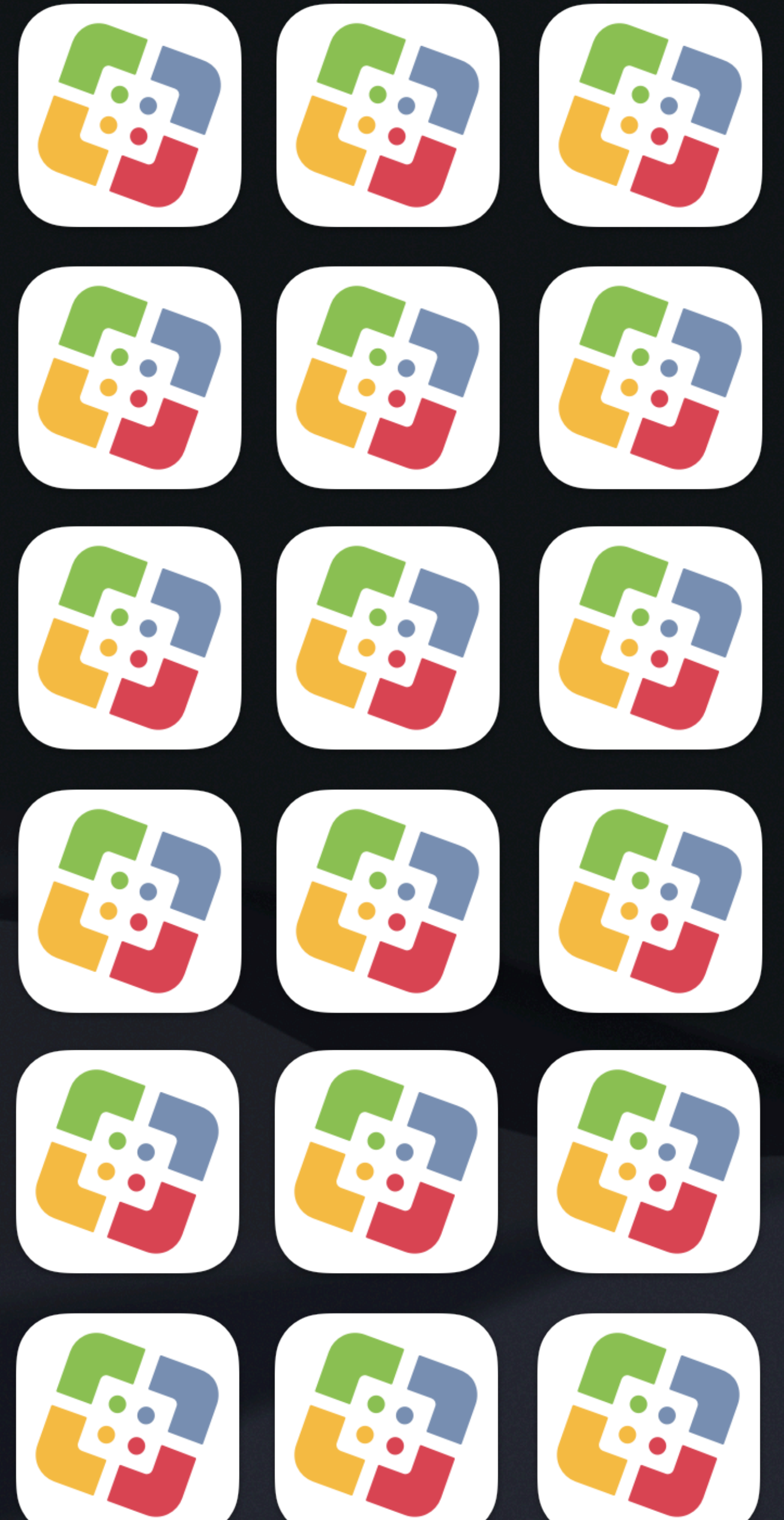
- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary
- Upload package to Jamf Pro
- Create or update policies and smart groups targeted at testers
- Testers do the tests

Software Deployment

- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary
- Upload package to Jamf Pro
- Create or update policies and smart groups targeted at testers
- Testers do the tests
- Create or update policies and smart groups available to production computers
- Delete the testing policies

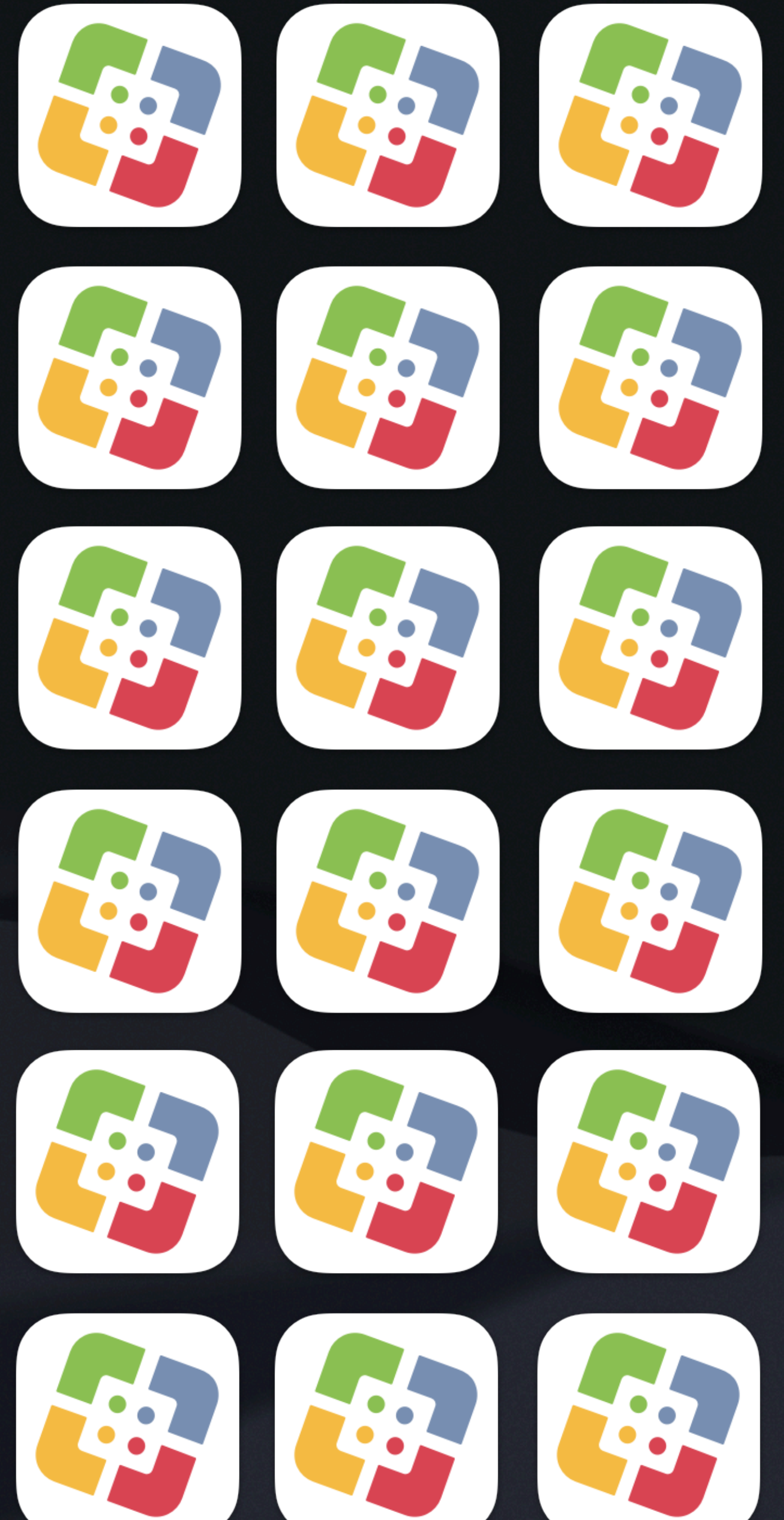
Software Deployment

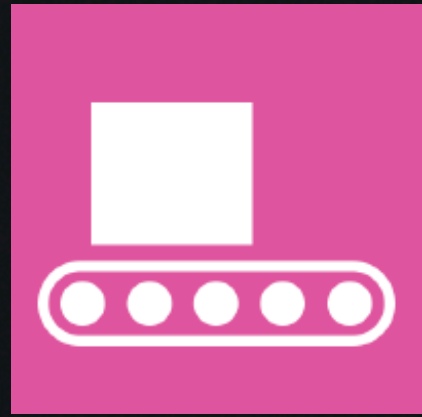
- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary
- Upload package to Jamf Pro
- Create or update policies and smart groups targeted at testers
- Testers do the tests
- Create or update policies and smart groups available to production computers
- Delete the testing policies
- Repeat on all Jamf tenants



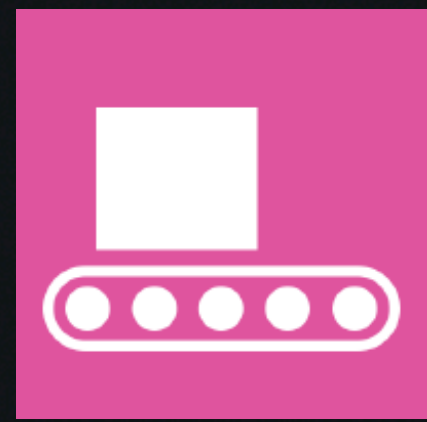
Software Deployment

- Check for a new version of a software title
- Download new version
- Verify security
- Repackage as necessary
- Upload package to Jamf Pro
- Create or update policies and smart groups targeted at testers
- Testers do the tests
- Create or update policies and smart groups available to production computers
- Delete the testing policies
- Repeat on all Jamf tenants

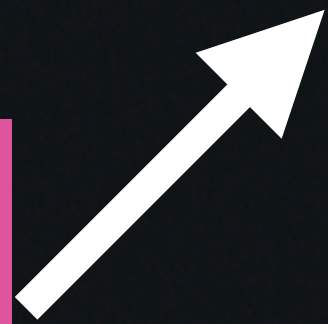


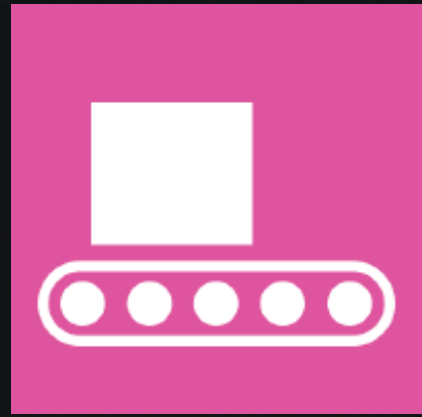


Testing

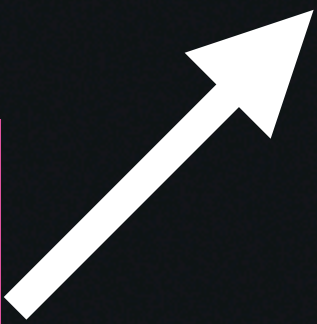


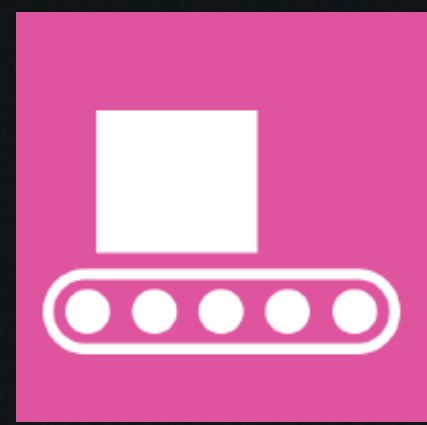
Testing



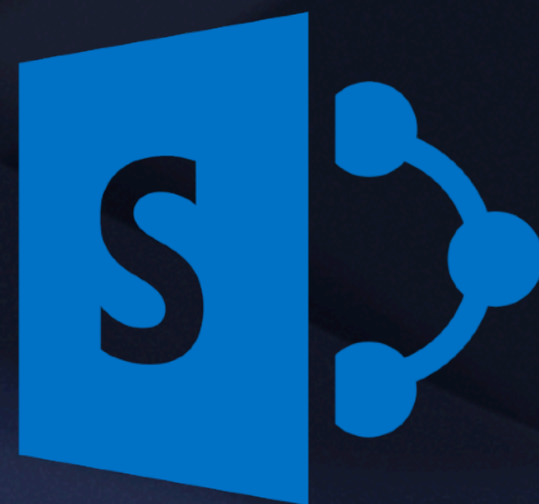


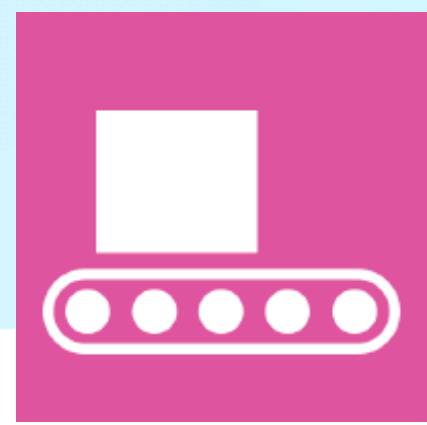
Testing



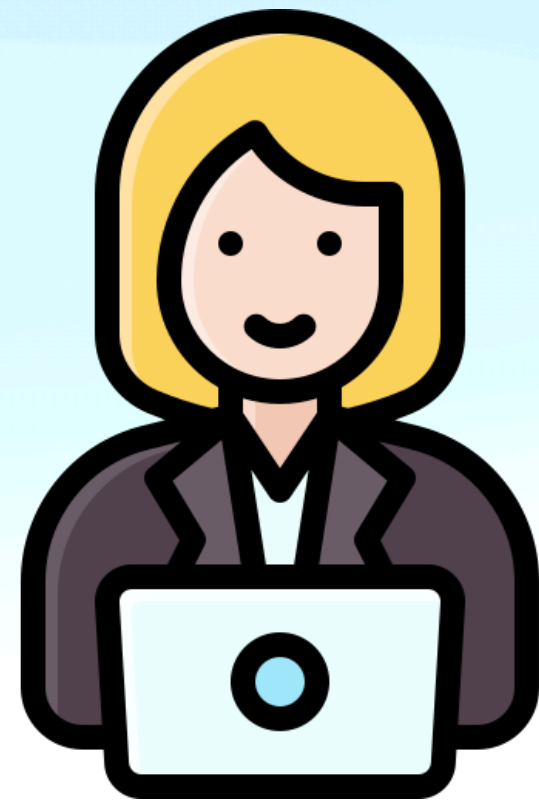
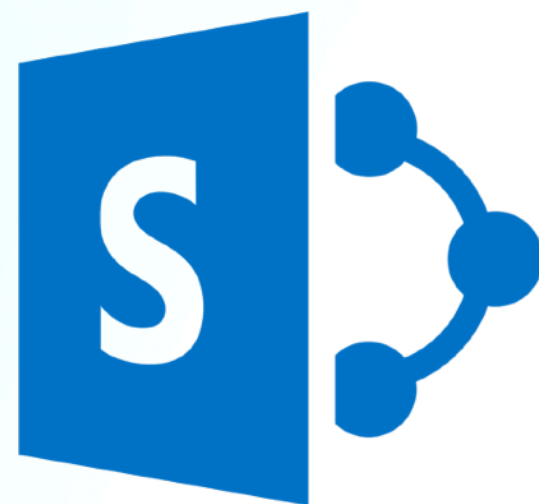


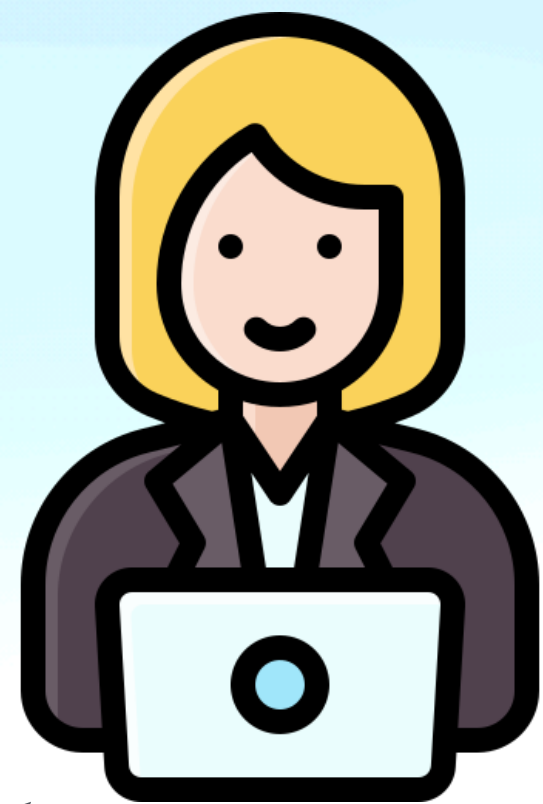
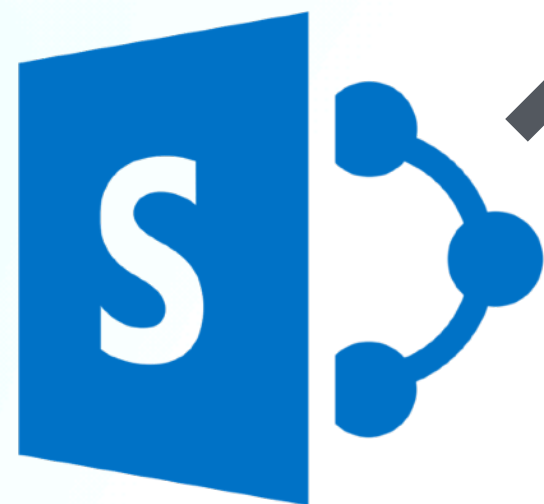
Testing

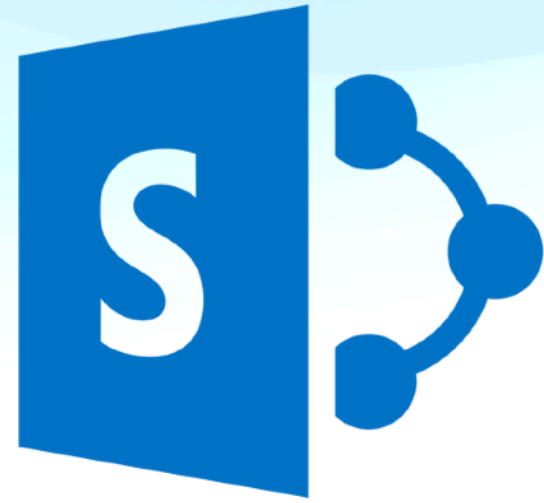
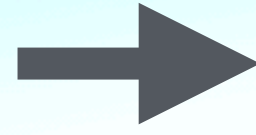
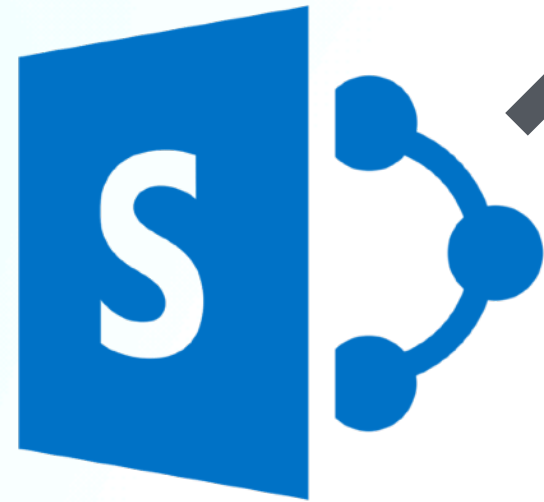


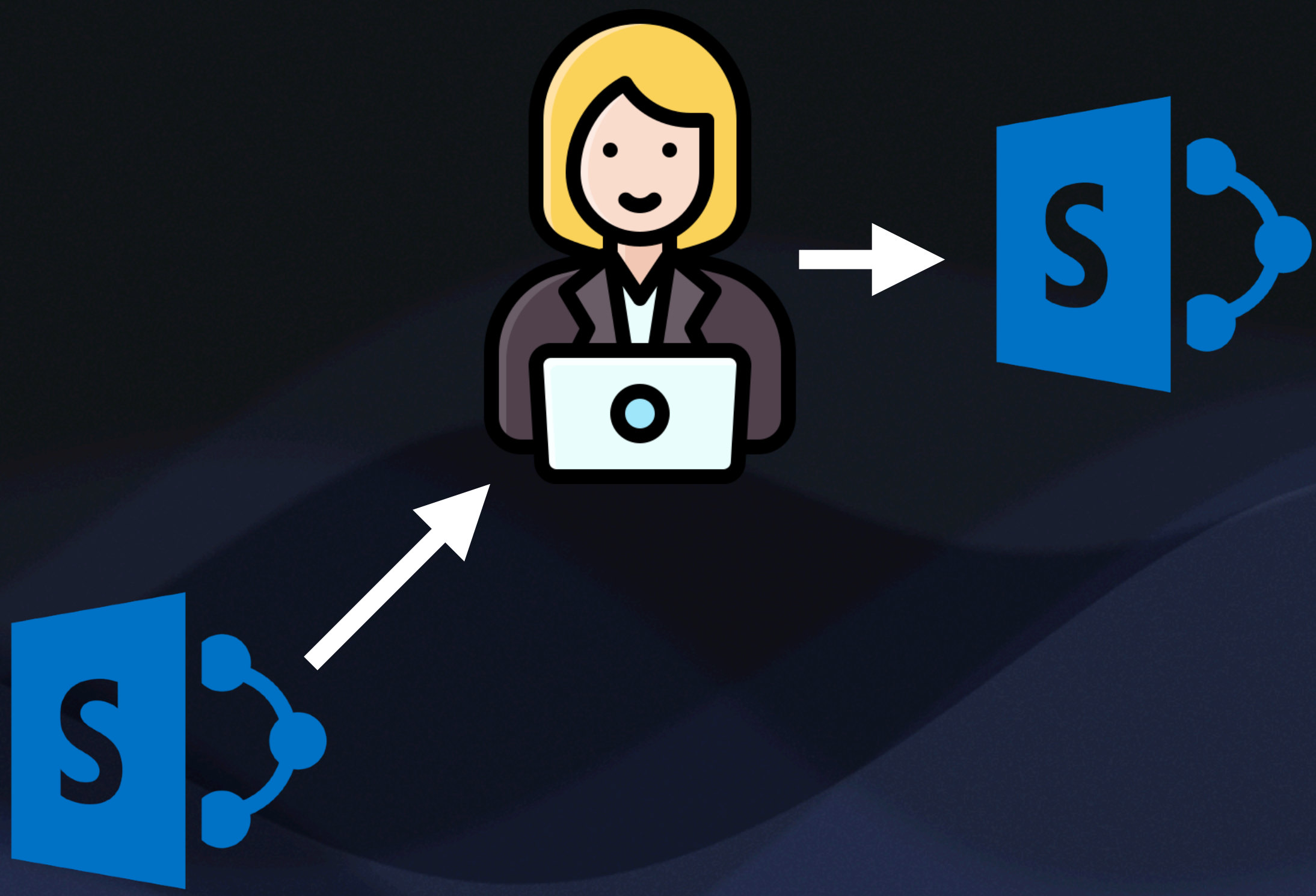


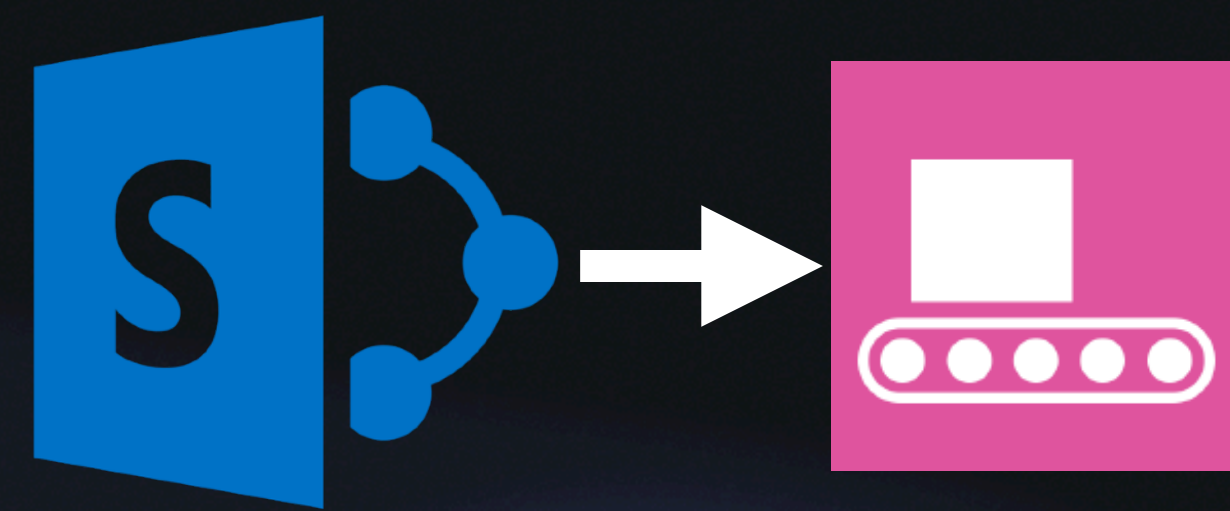
Testing





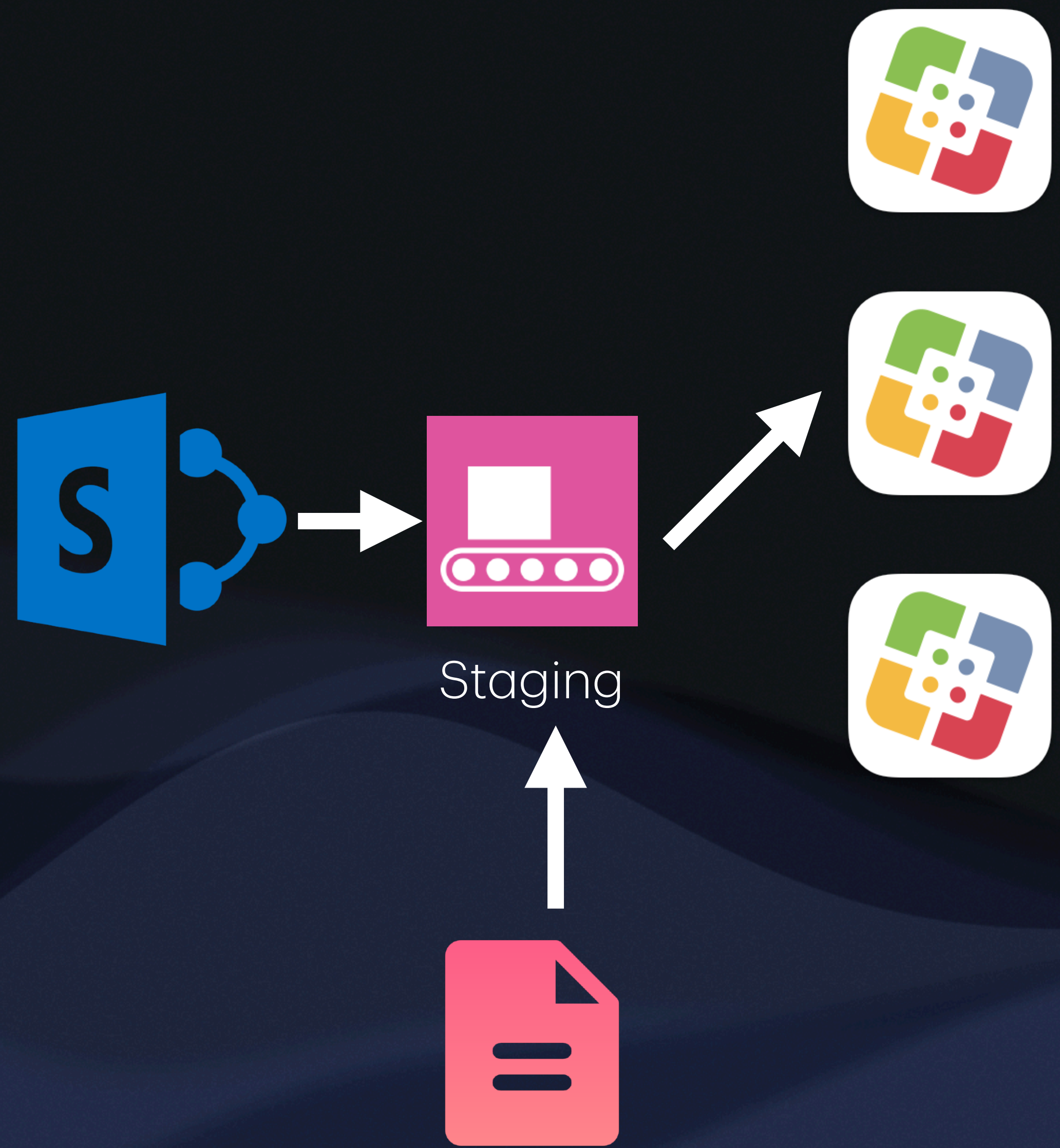


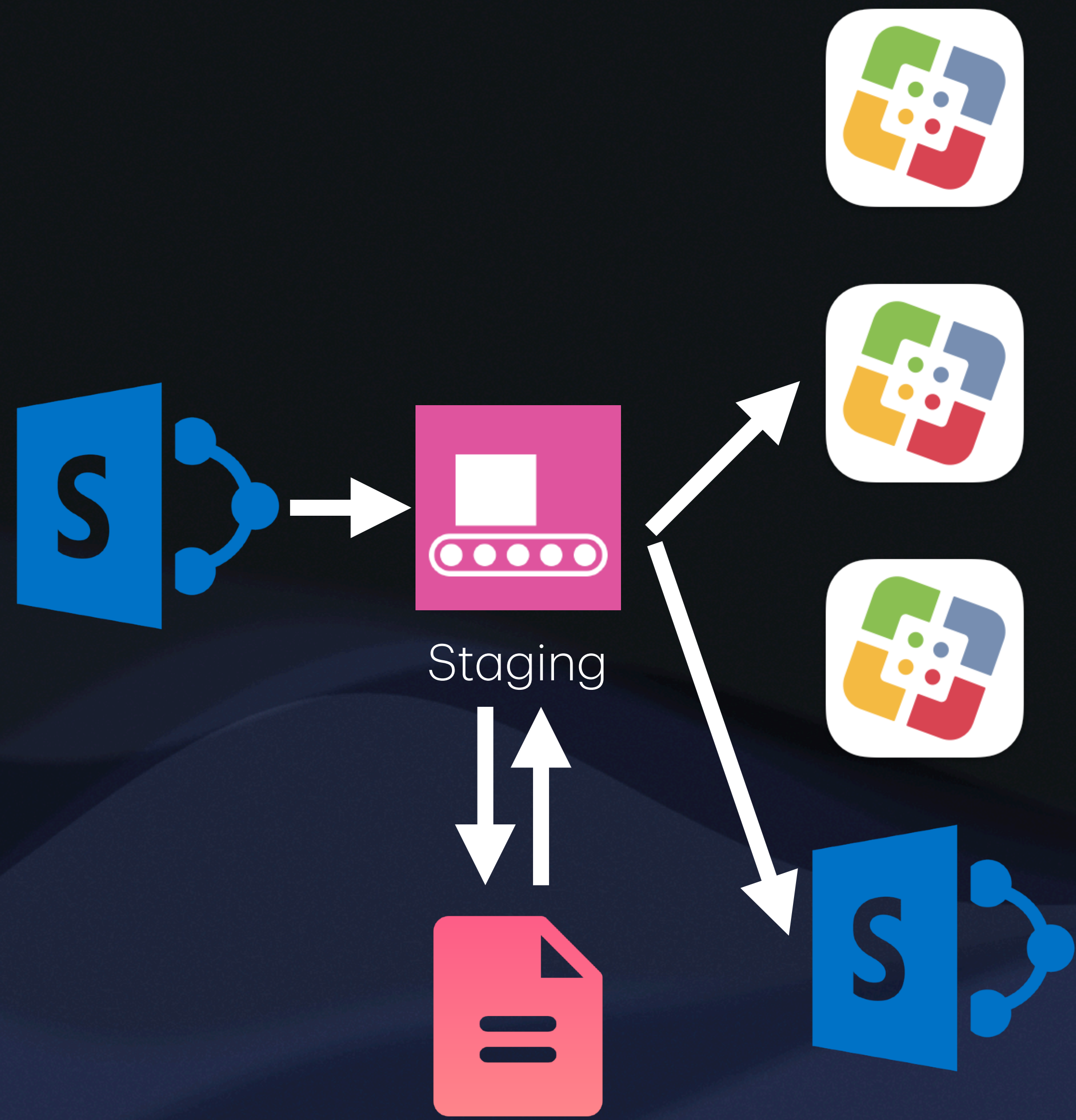


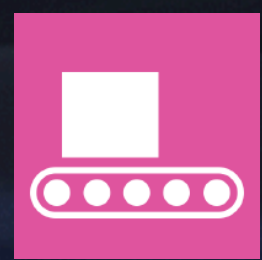


Staging





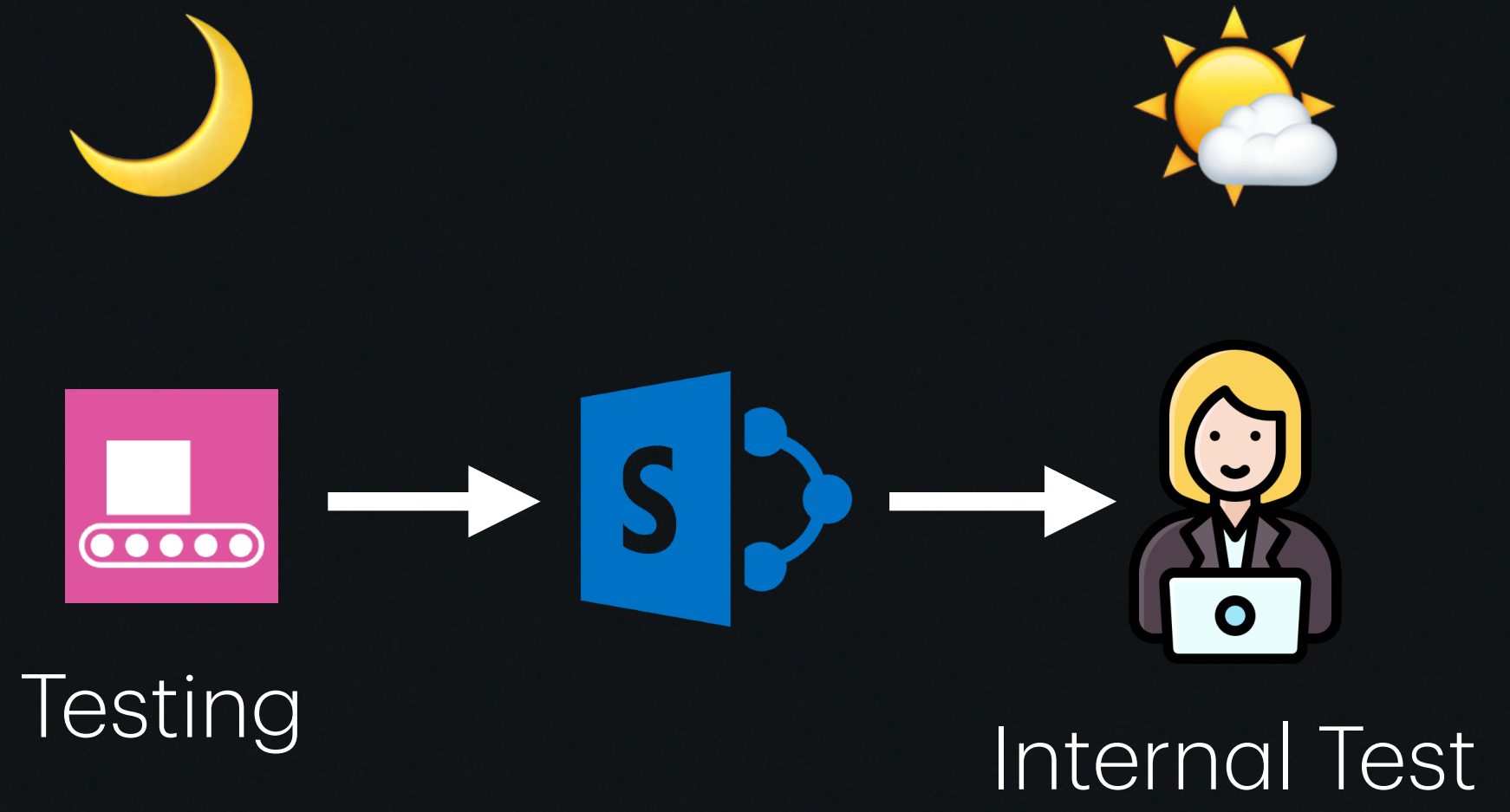




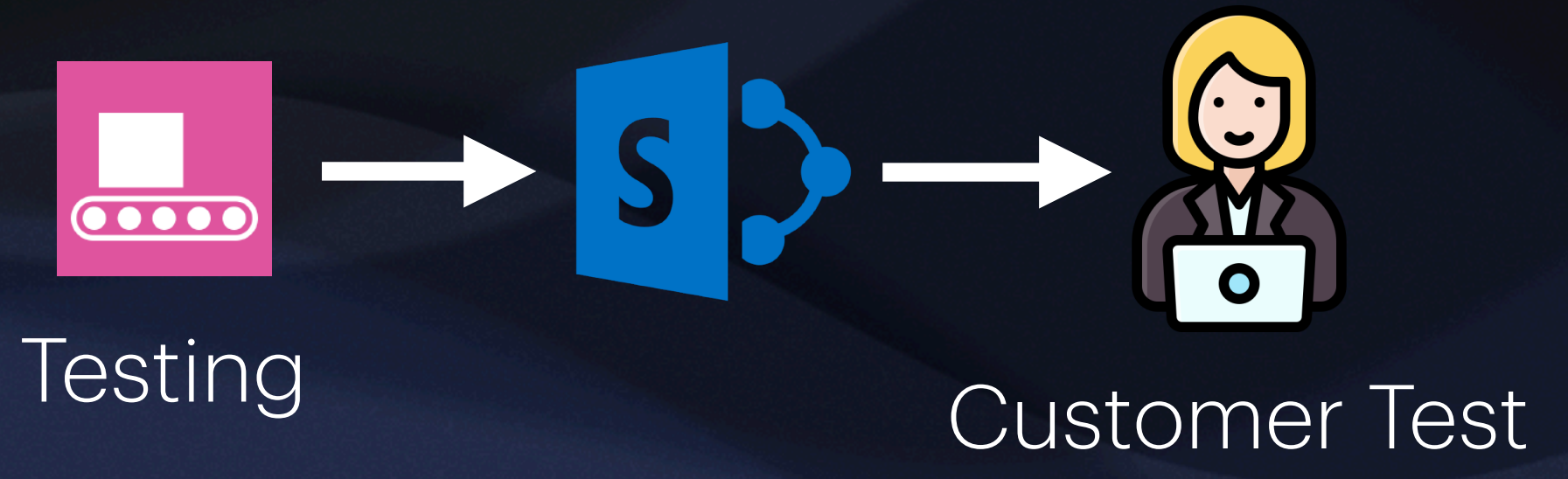
Testing

Customer Test

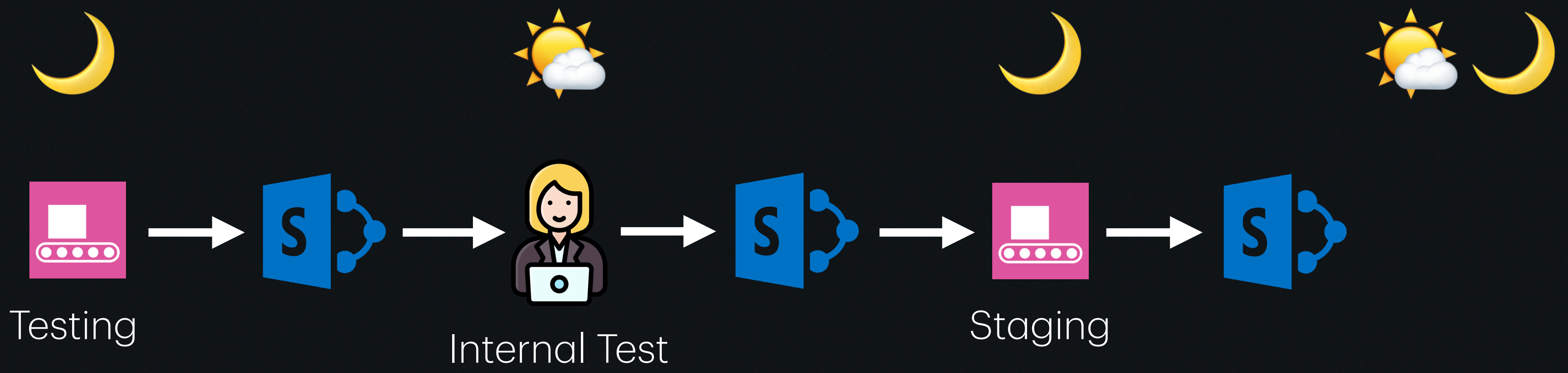
TST



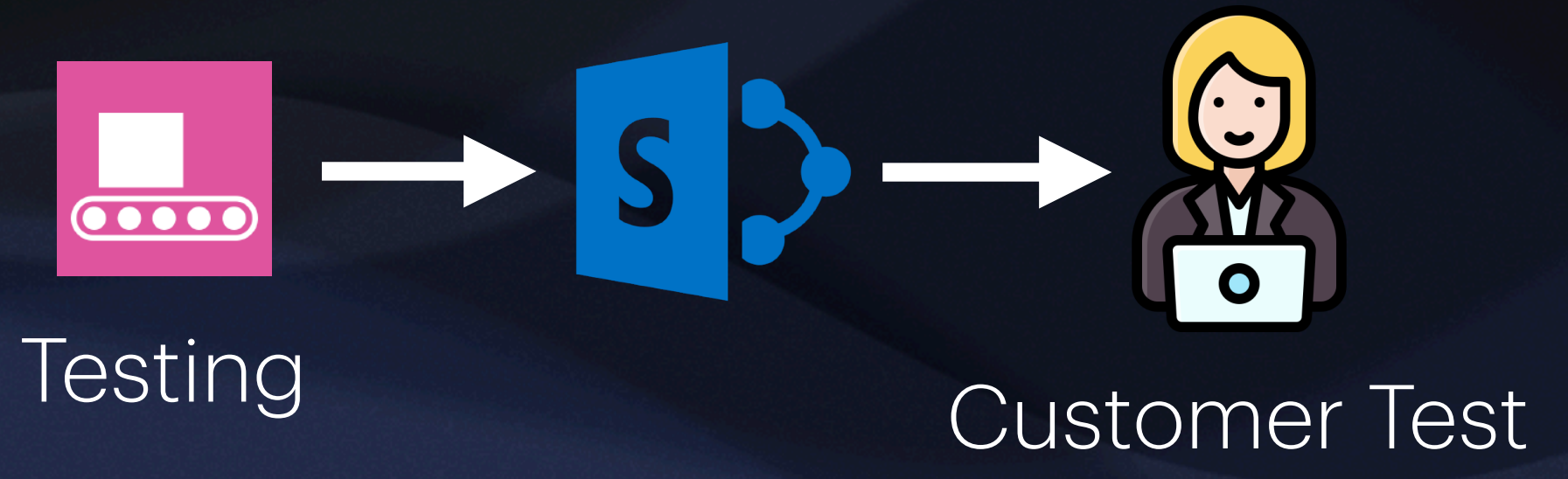
PRD



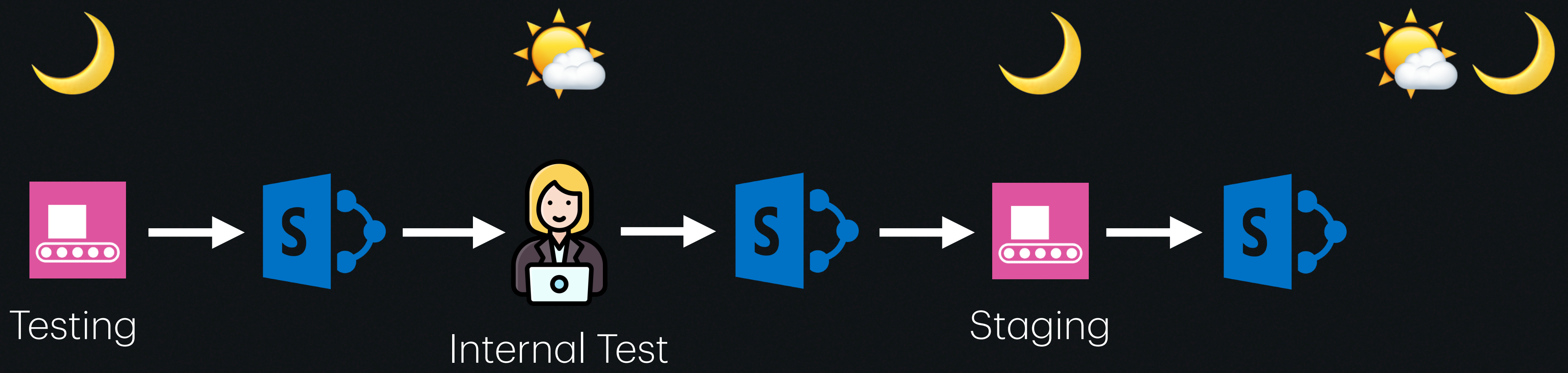
TST



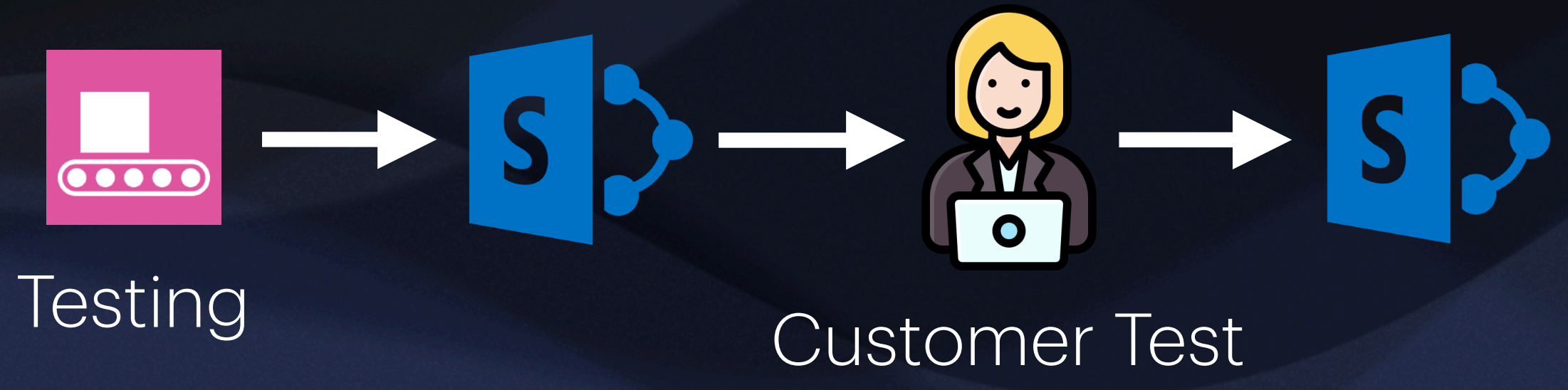
PRD



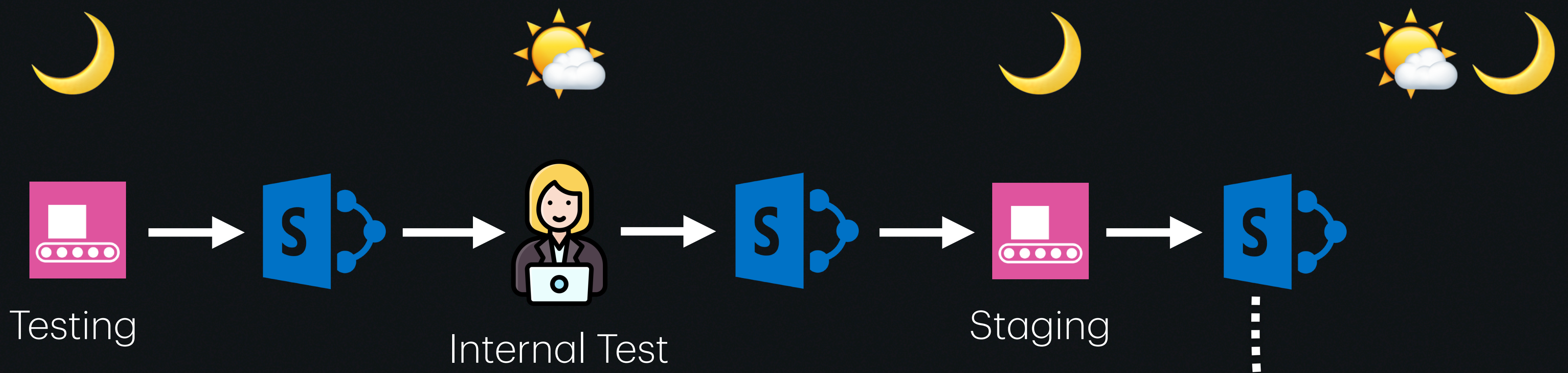
TST



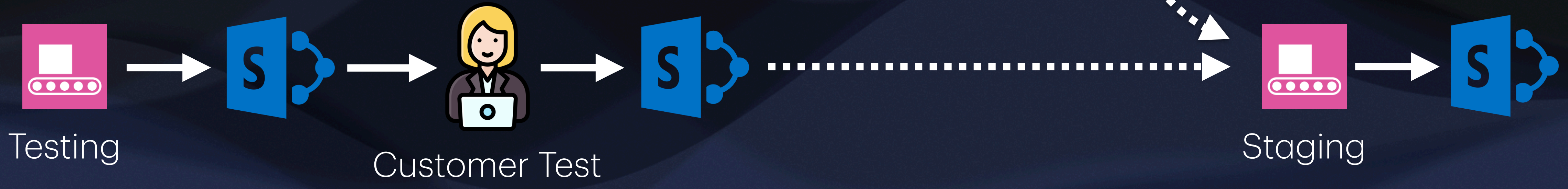
PRD



TST



PRD



Testing Software ... why?





Security



Security



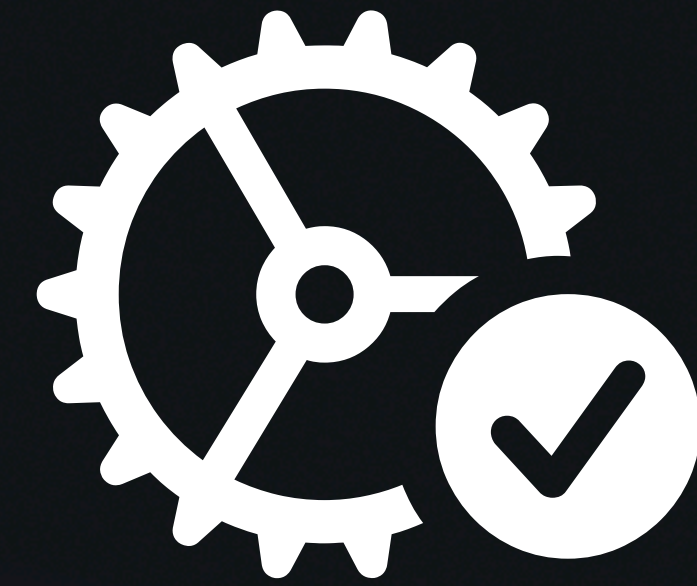
Functionality



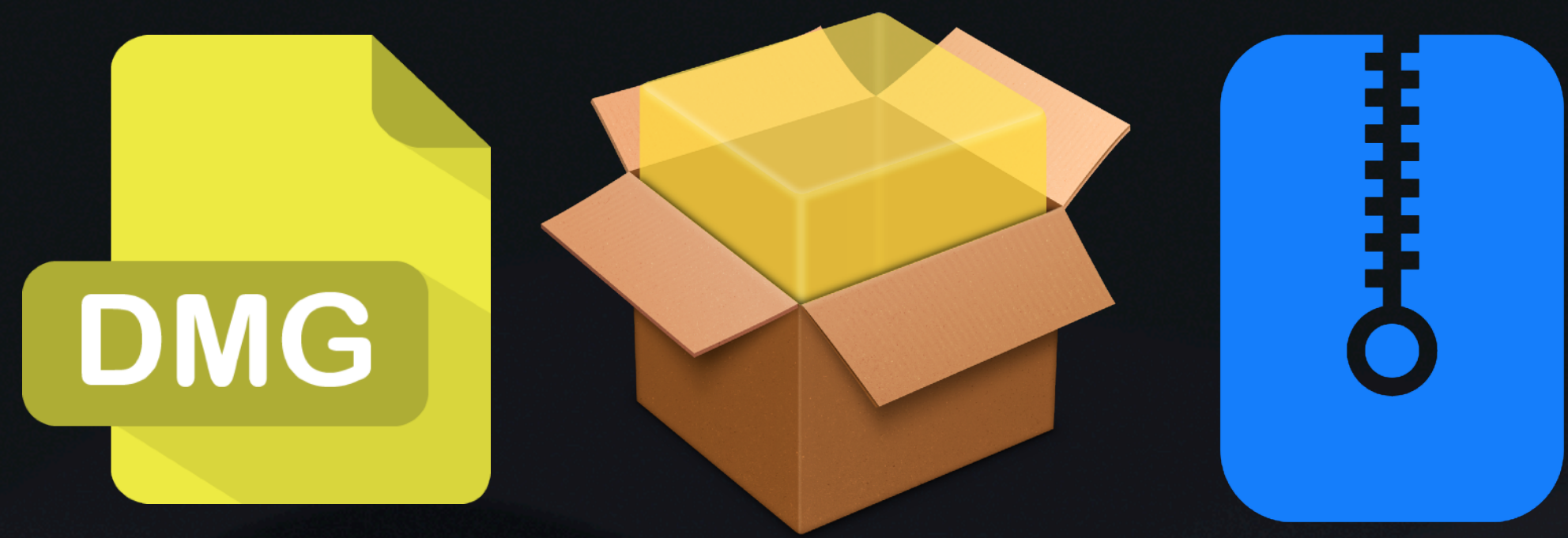
Security



Functionality



Management



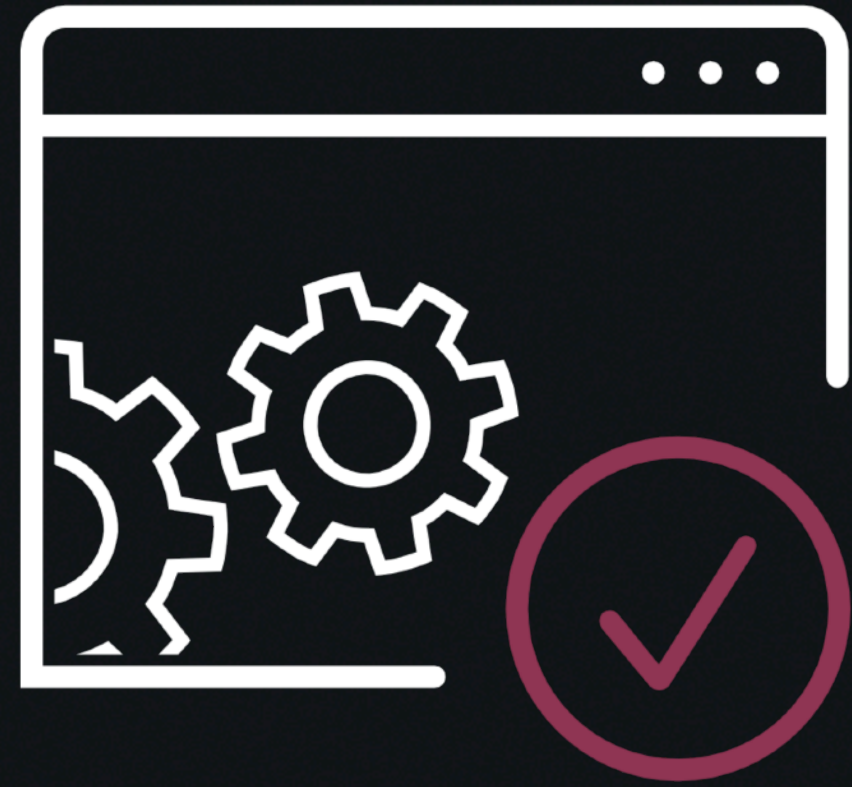


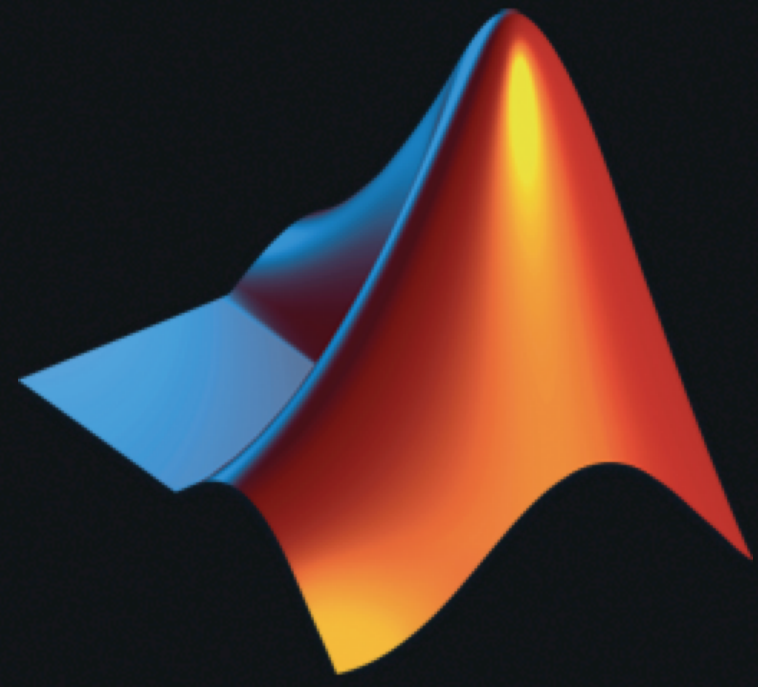
Pirate Bay

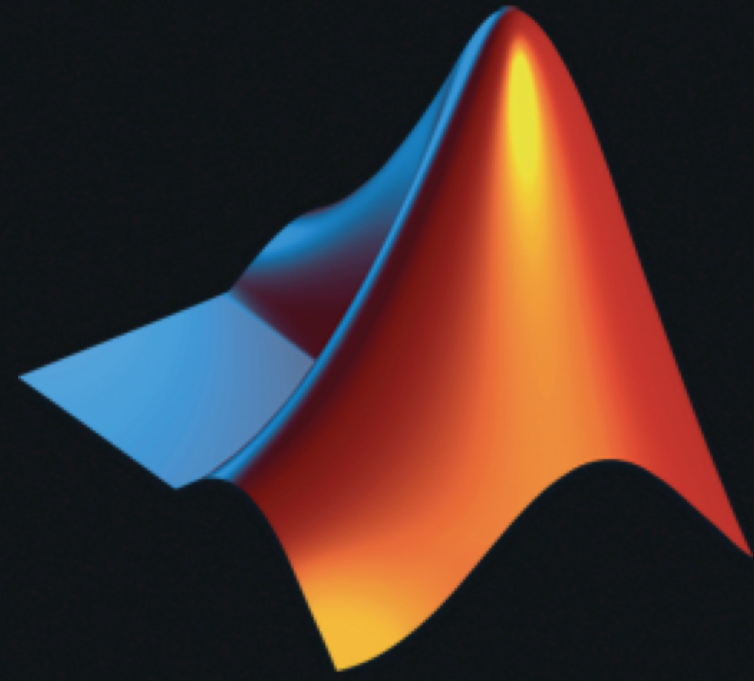


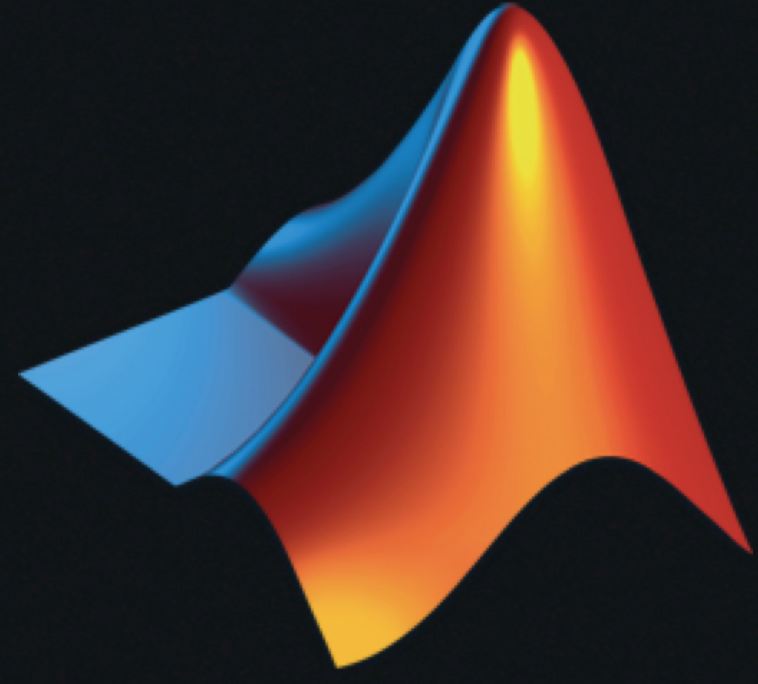


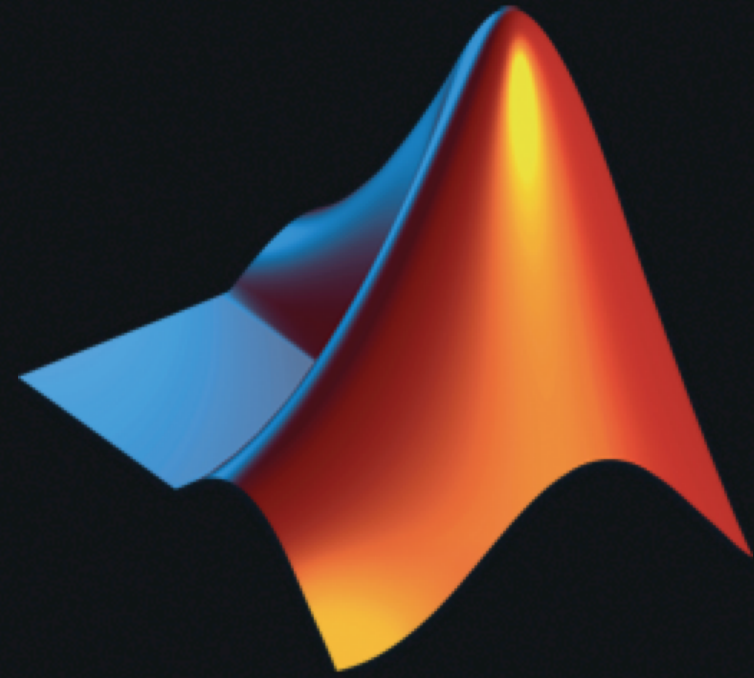


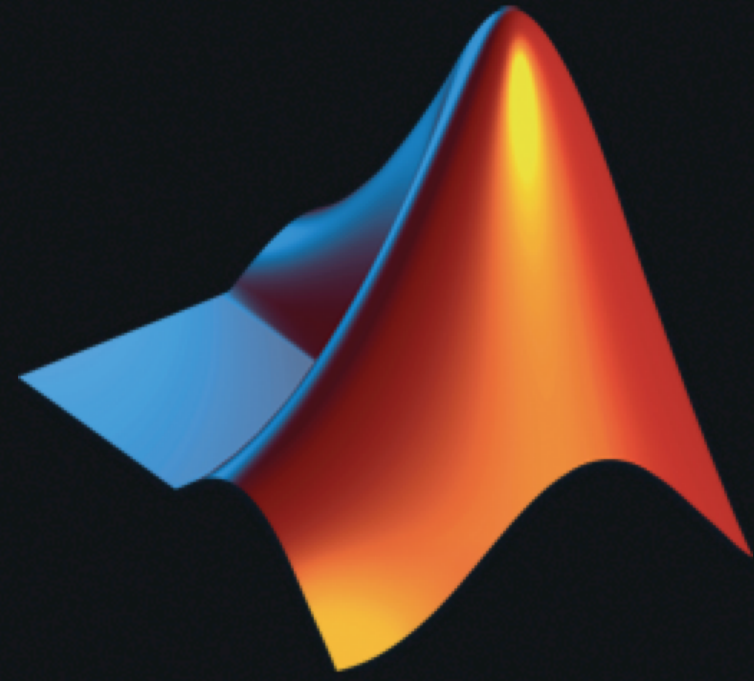


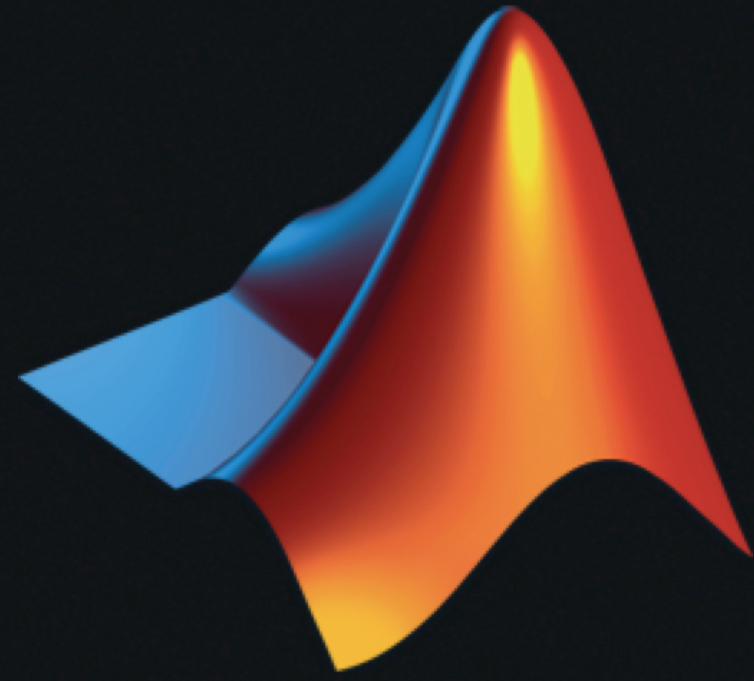


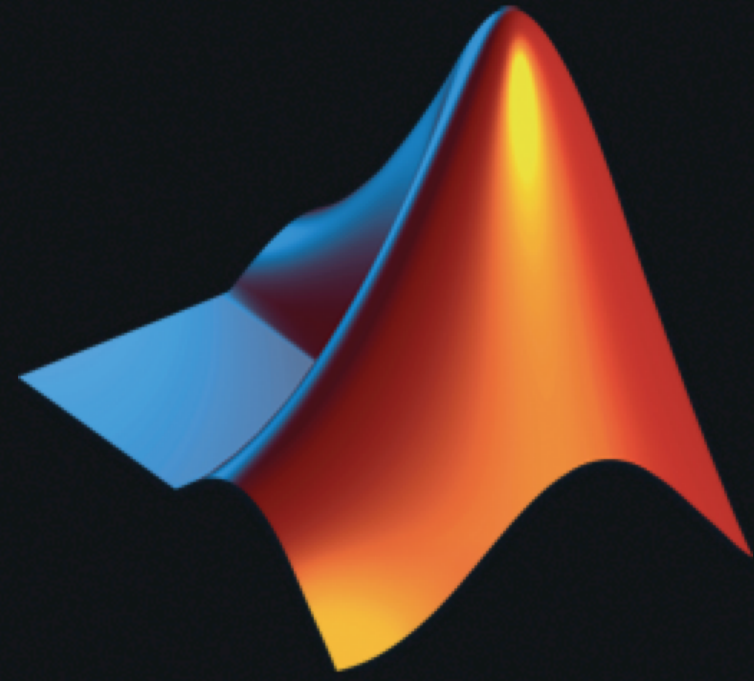














CROWDSTRIKE



ETH Self Service

Cloud Sandbox

Search

Home

Browse

Notifications

History



Featured

10 Results < >



Install macOS Updates

Info



Update Docker arm64

Update to 4.28.0



Update ETH Printers

Update to 2.1.3



Update ETH Support Menu Bar App

Update to 2.6



Update Google Chrome

Update to 129.0...



Update Logic Pro X

Update to 11.0.1

Browse

15 Results < >

- All >
- Creativity >
- Mac App Store Apps >
- Tools & Accessories >
- Untested >
- Browsers & Players >
- Science & Education >
- Connect & Remote >
- Development & Comp... >
- Productivity >
- Uninstallers >
- Administration >
- Imaging & Design >
- Security & Privacy >



ETH Self Service

Cloud Sandbox

Search

Home

Browse

Notifications

History



Featured

10 Results < >



Install macOS Updates

Info



Update Docker arm64

Update to 4.28.0



Update ETH Printers

Update to 2.1.3



Update ETH Support Menu Bar App

Update to 2.6



Update Google Chrome

Update to 129.0...



Update Logic Pro X

Update to 11.0.1

Browse

15 Results < >

- All >
- Creativity >
- Mac App Store Apps >
- Tools & Accessories >
- Untested >
- Browsers & Players >
- Science & Education >
- Connect & Remote >
- Development & Comp... >
- Productivity >
- Uninstallers >
- Administration >
- Imaging & Design >
- Security & Privacy >

Problems for End Users

Installation / Uninstallation / Functionality



Problems for End Users

Installation / Uninstallation / Functionality

update every day

update fails

g over itself

cannot install on mobile

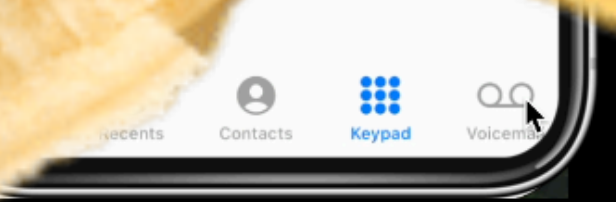
asking for license

pop ups, credentials,
helper tool

two apps with the
same name

cannot

app not available in
Self Service



Problems for End Users

Installation / Uninstallation / Functionality

update every day

update fails

g over itself

cannot install on mobile

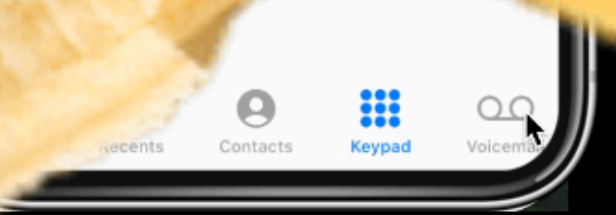
asking for license

pop ups, credentials,
helper tool

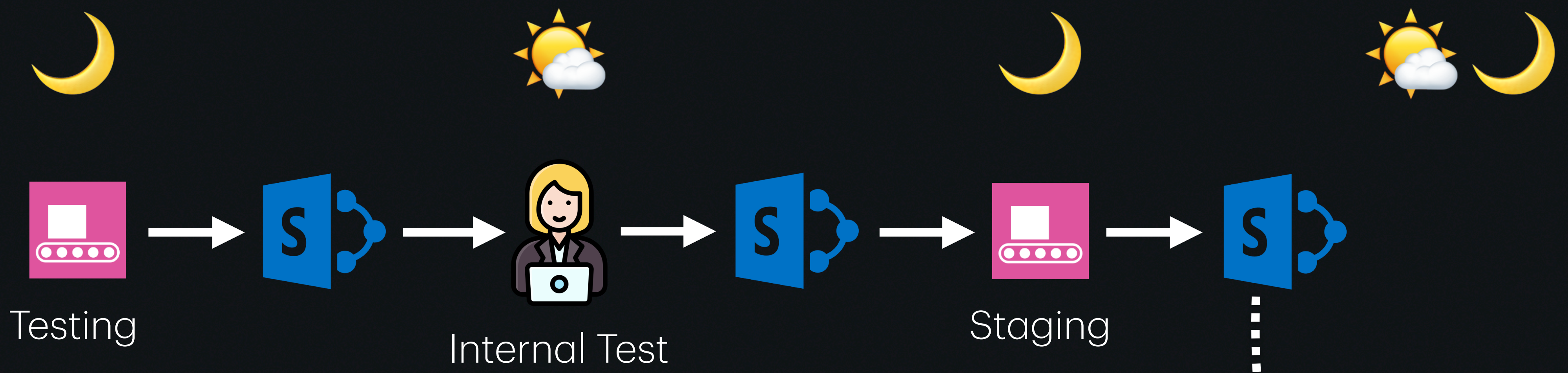
two apps with the
same name

cannot

app not available in
Self Service



TST

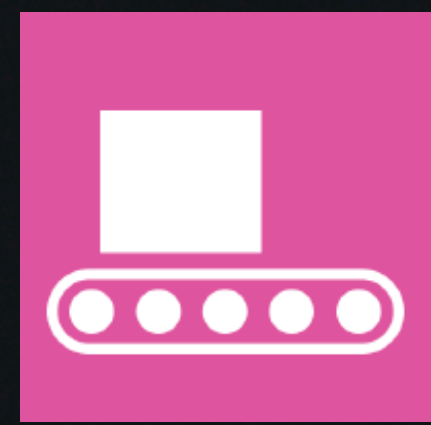


PRD

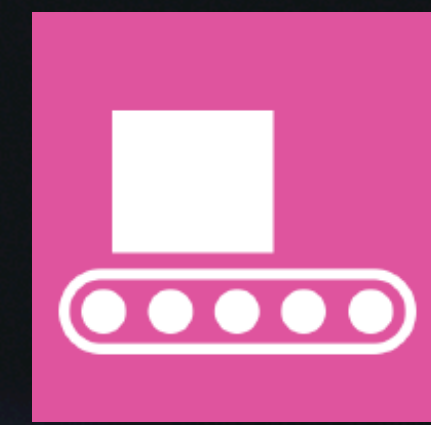
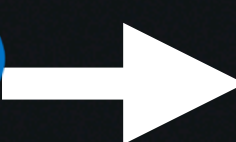
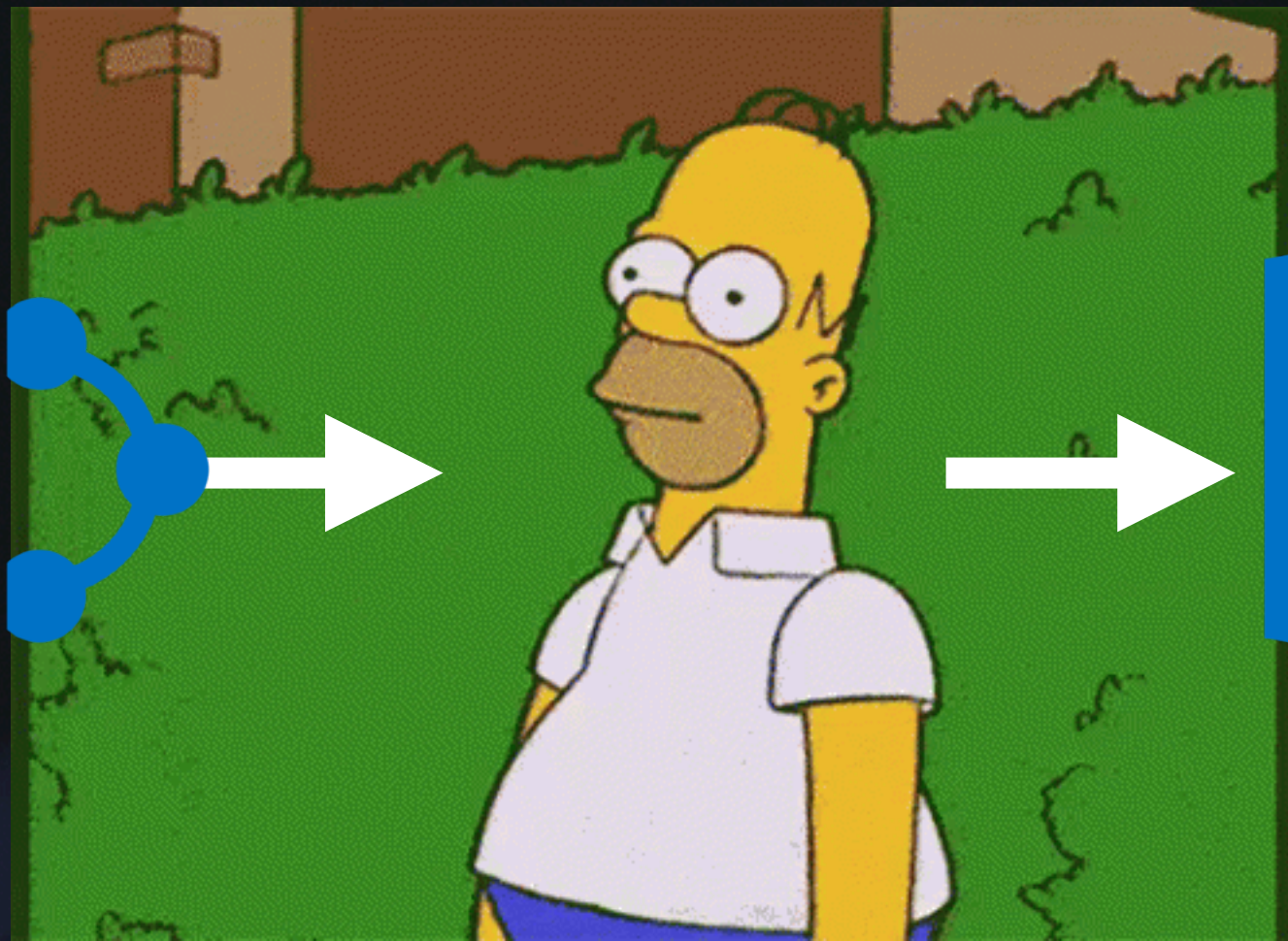
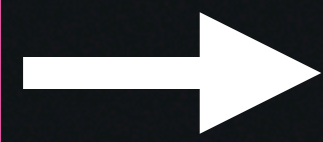


The Problem with Testing...

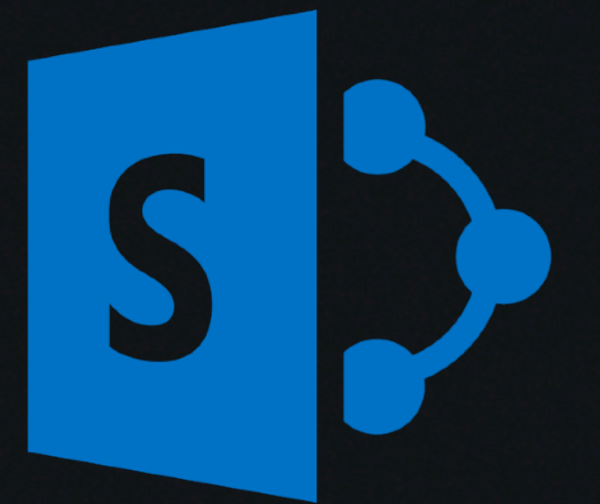


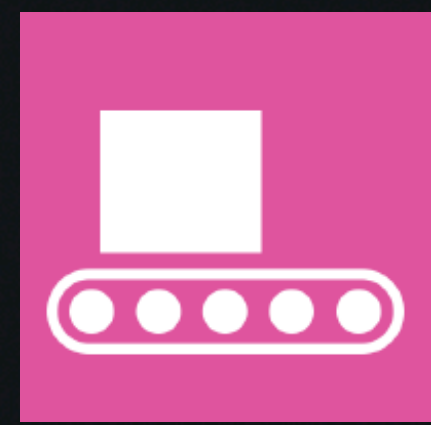


Testing

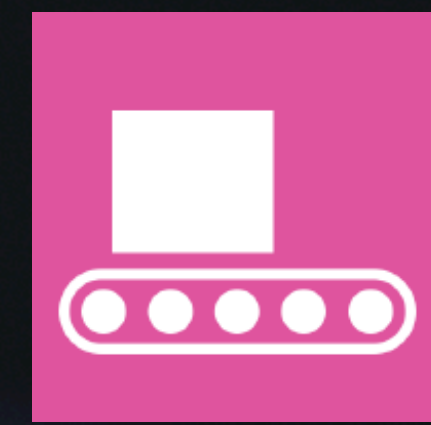
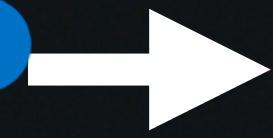
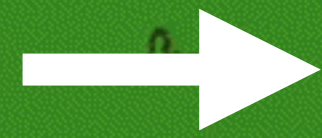
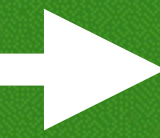
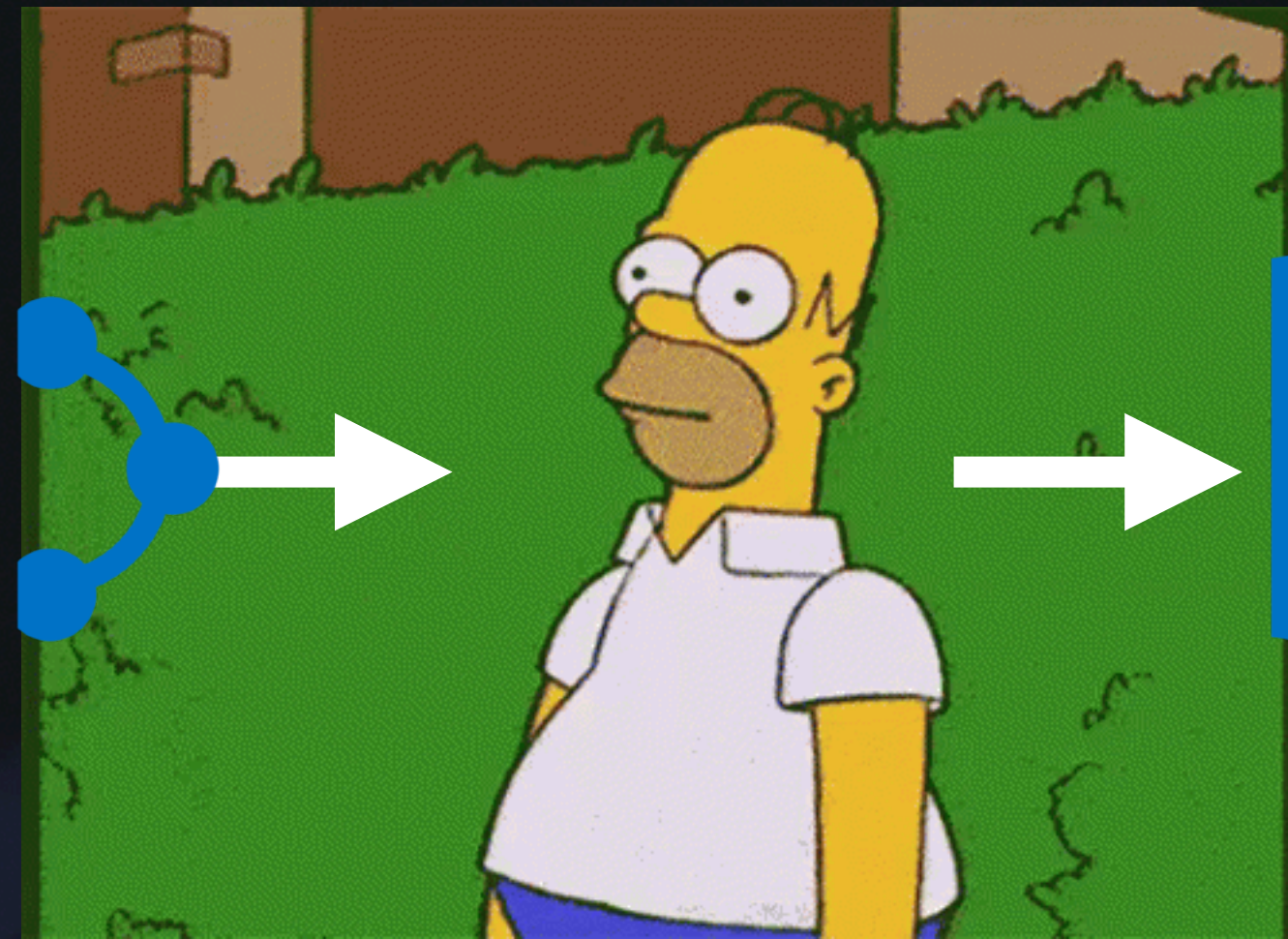
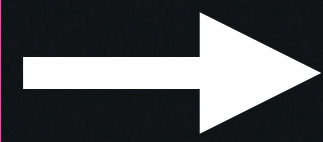


Staging

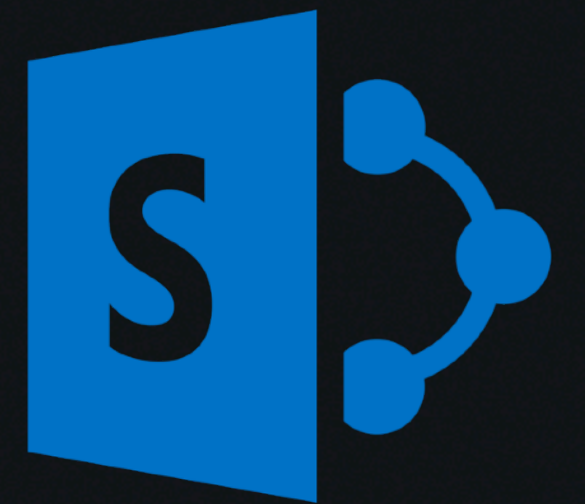
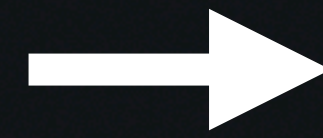


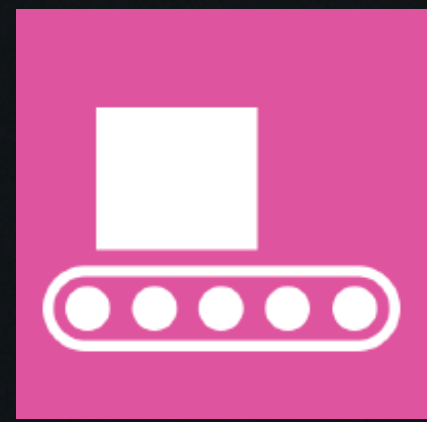


Testing

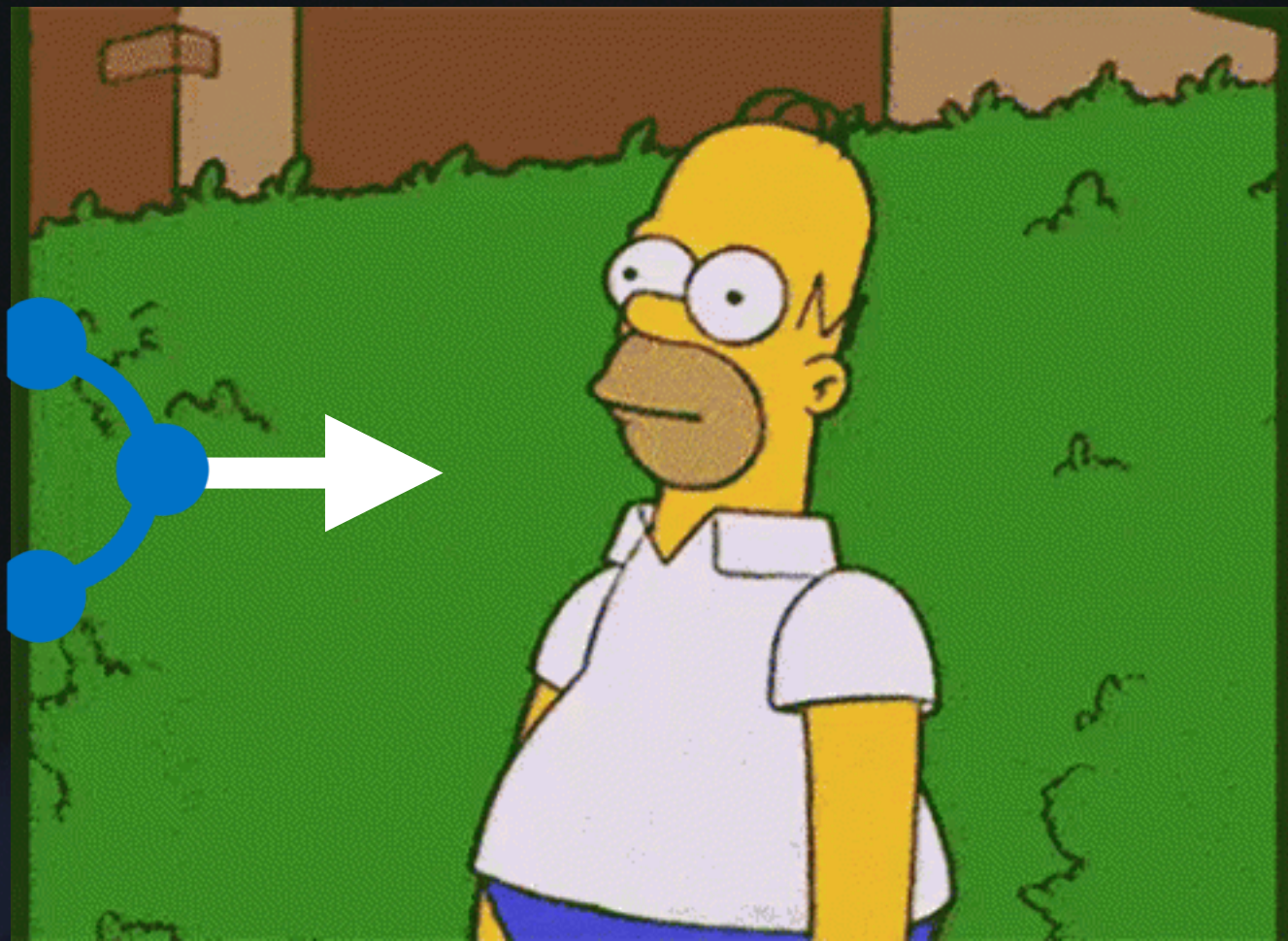
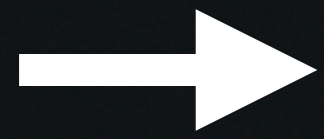


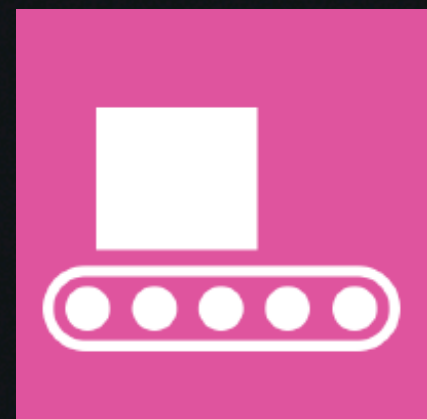
Staging



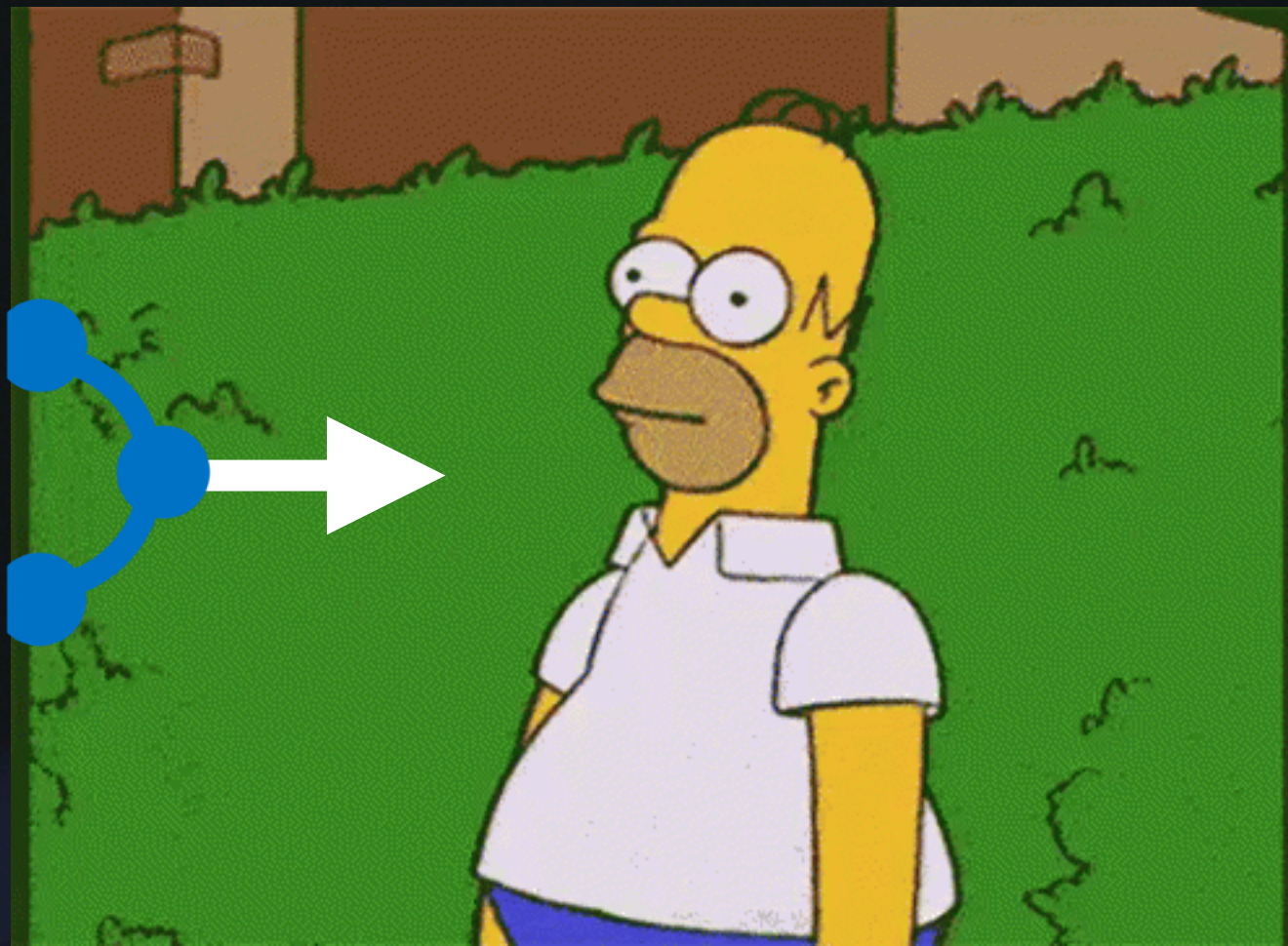
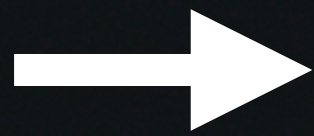


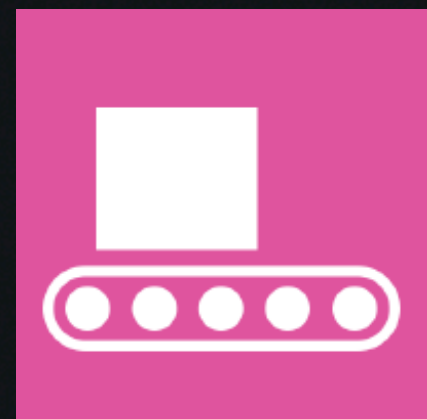
Testing



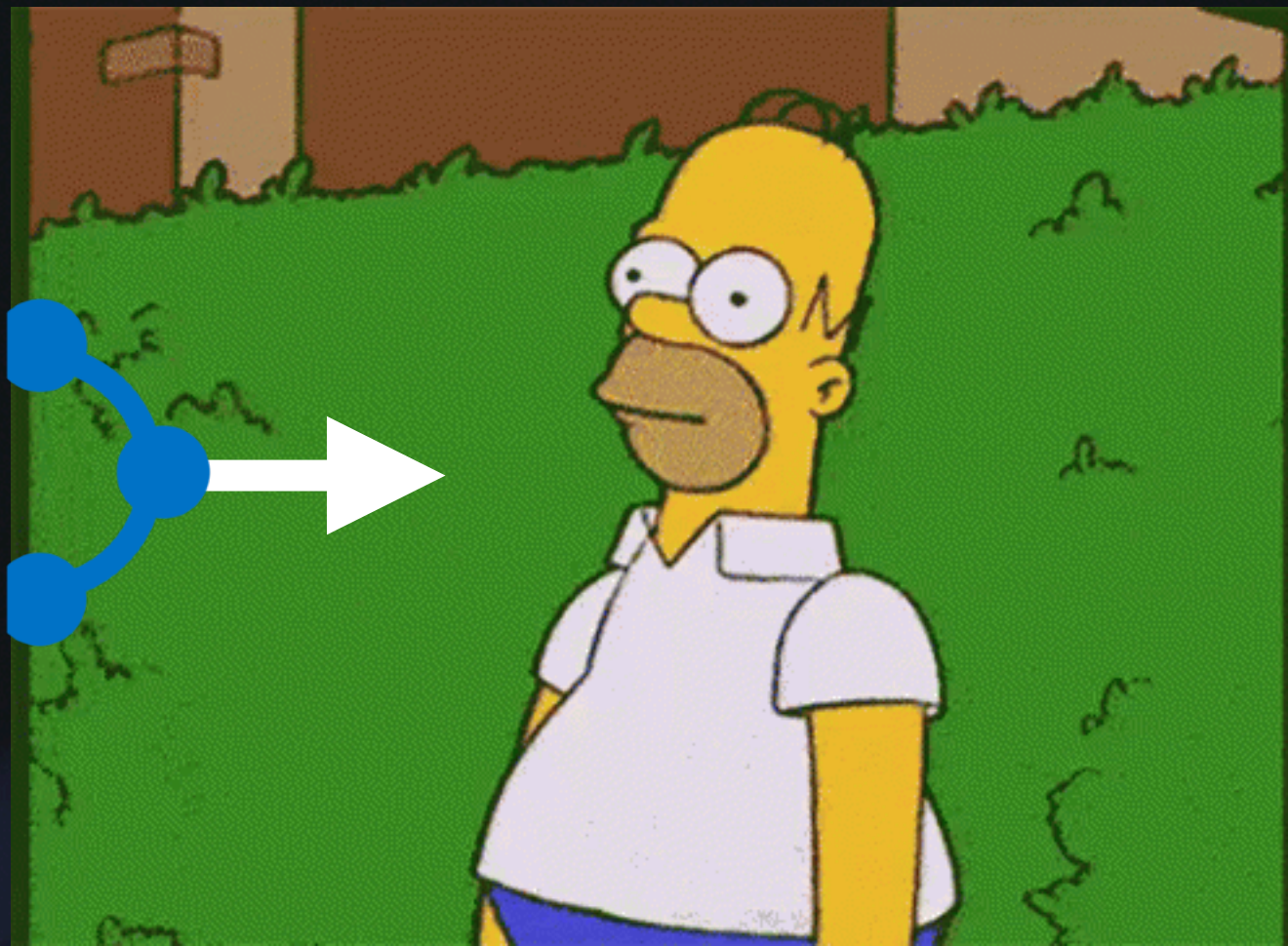
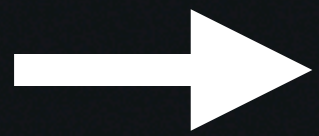


Testing





Testing



Problem

Discovery

Fix

Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles

Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles
Bundle Identifier change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version

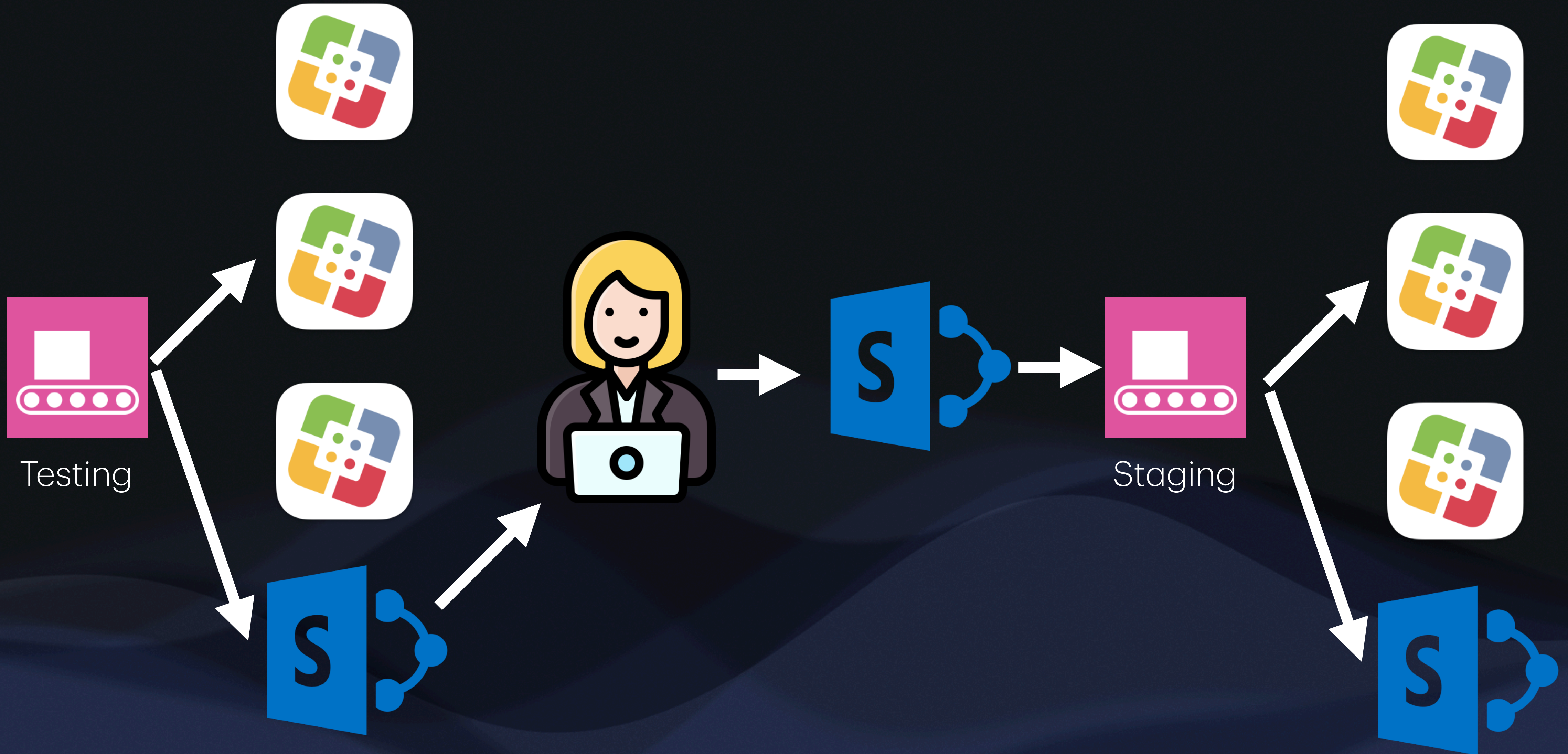
Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles
Bundle Identifier change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
App name change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version

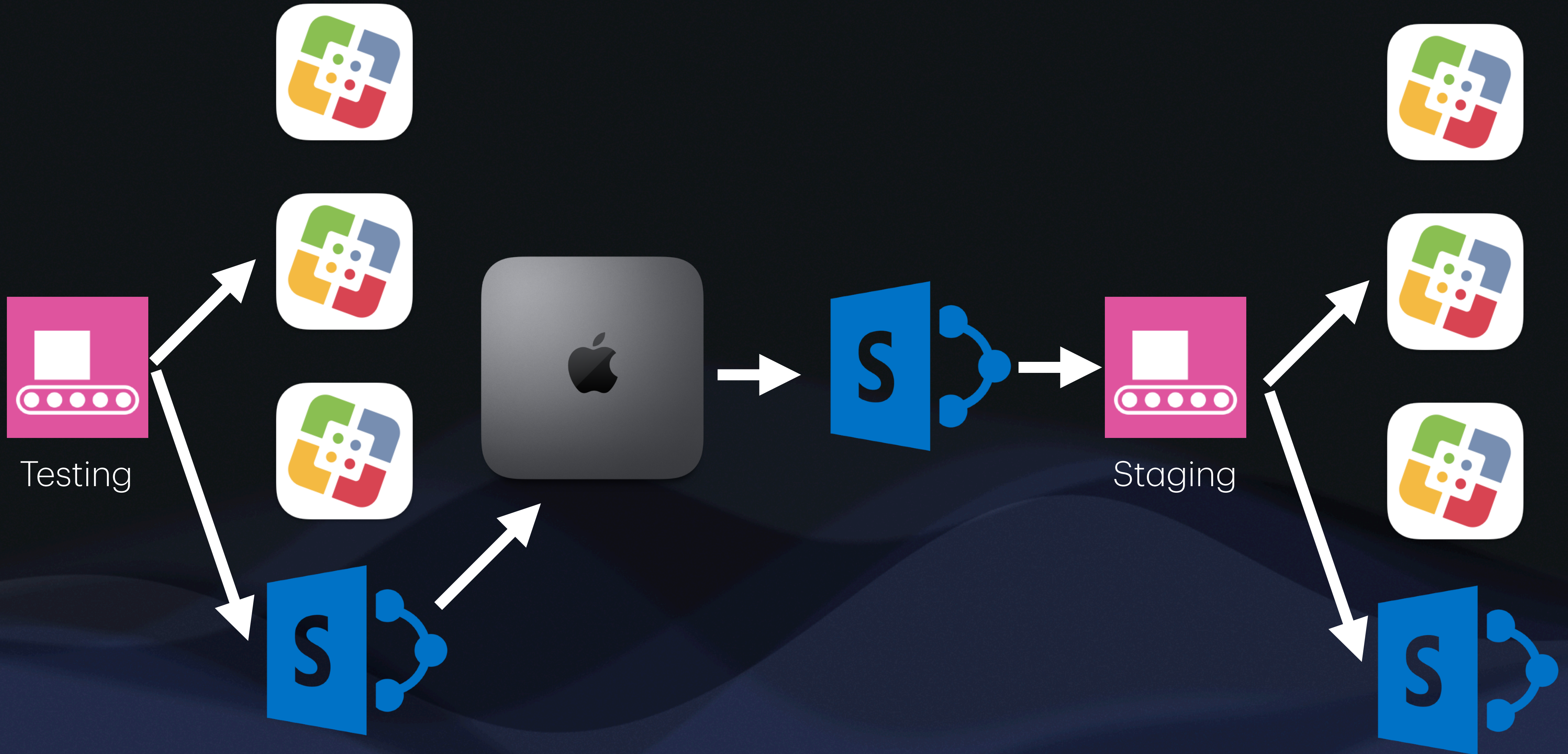
Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles
Bundle Identifier change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
App name change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
Version comparison mismatch	Retracted installer, app reinstalling, Self Service errors	Update AutoPkg recipe

Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles
Bundle Identifier change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
App name change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
Version comparison mismatch	Retracted installer, app reinstalling, Self Service errors	Update AutoPkg recipe
Broken AutoPkg recipe	App reinstalling, Self Service errors, app not available	Update AutoPkg recipe

Problem	Discovery	Fix
Functionality not as expected	End-user/community feedback	Contact vendor/wait for fix, deploy new profiles
Bundle Identifier change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
App name change	Uninstaller not working, multiple applications installed	Update AutoPkg recipe, remediate old version
Version comparison mismatch	Retracted installer, app reinstalling, Self Service errors	Update AutoPkg recipe
Broken AutoPkg recipe	App reinstalling, Self Service errors, app not available	Update AutoPkg recipe
Failed dependencies	Installer not working	Update AutoPkg recipe to include or trigger dependencies

Problem	Discovery	Auto-detection?
Functionality not as expected	End-user/community feedback	✗
Bundle Identifier change	Uninstaller not working, multiple applications installed	✓
App name change	Uninstaller not working, multiple applications installed	✓
Version comparison mismatch	Retracted installer, app reinstalling, Self Service errors	✓
Broken AutoPkg recipe	App reinstalling, Self Service errors, app not available	✓
Failed dependencies	Installer not working	✓

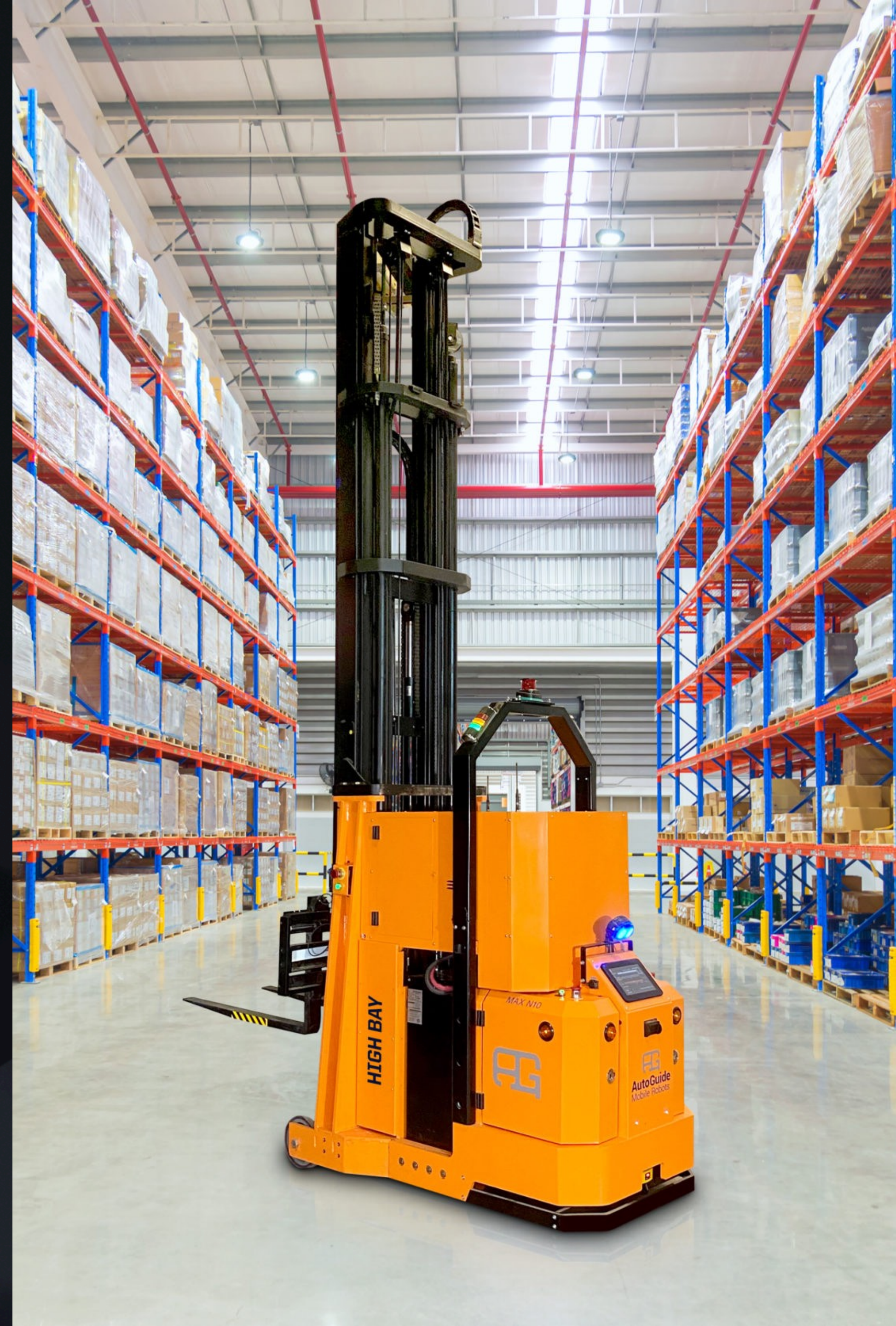




Testing

Staging

Automated App Testing



Automated testing

Advantages

Limitations

Automated testing

Advantages

Limitations

- Very limited app functionality testing

Automated testing

Advantages

Limitations

- Very limited app functionality testing
- Only app open/close can be tested

Automated testing

Advantages

Limitations

- Very limited app functionality testing
- Only app open/close can be tested
- Limited to software that fits that open/close functionality model

Automated testing

Advantages

Limitations

- Very limited app functionality testing
- Only app open/close can be tested
- Limited to software that fits that open/close functionality model
- Excludes plugins, collections, highly complex apps, some security software

Automated testing

Advantages

- No user interaction required

Limitations

- Very limited app functionality testing
- Only app open/close can be tested
- Limited to software that fits that open/close functionality model
- Excludes plugins, collections, highly complex apps, some security software

Automated testing

Advantages

- No user interaction required
- Extended testing workflow can test the execution of multiple policies

Limitations

- Very limited app functionality testing
- Only app open/close can be tested
- Limited to software that fits that open/close functionality model
- Excludes plugins, collections, highly complex apps, some security software

Automated testing

Advantages

- No user interaction required
- Extended testing workflow can test the execution of multiple policies
- Can run on a daily basis, ensuring new software is released promptly

Limitations

- Very limited app functionality testing
- Only app open/close can be tested
- Limited to software that fits that open/close functionality model
- Excludes plugins, collections, highly complex apps, some security software

Automated Software Testing

A Deep Dive





Jamf Groups



Jamf Groups

Jamf Policies



Jamf Groups

Jamf Binary

Jamf Policies



Jamf Groups

Jamf Binary

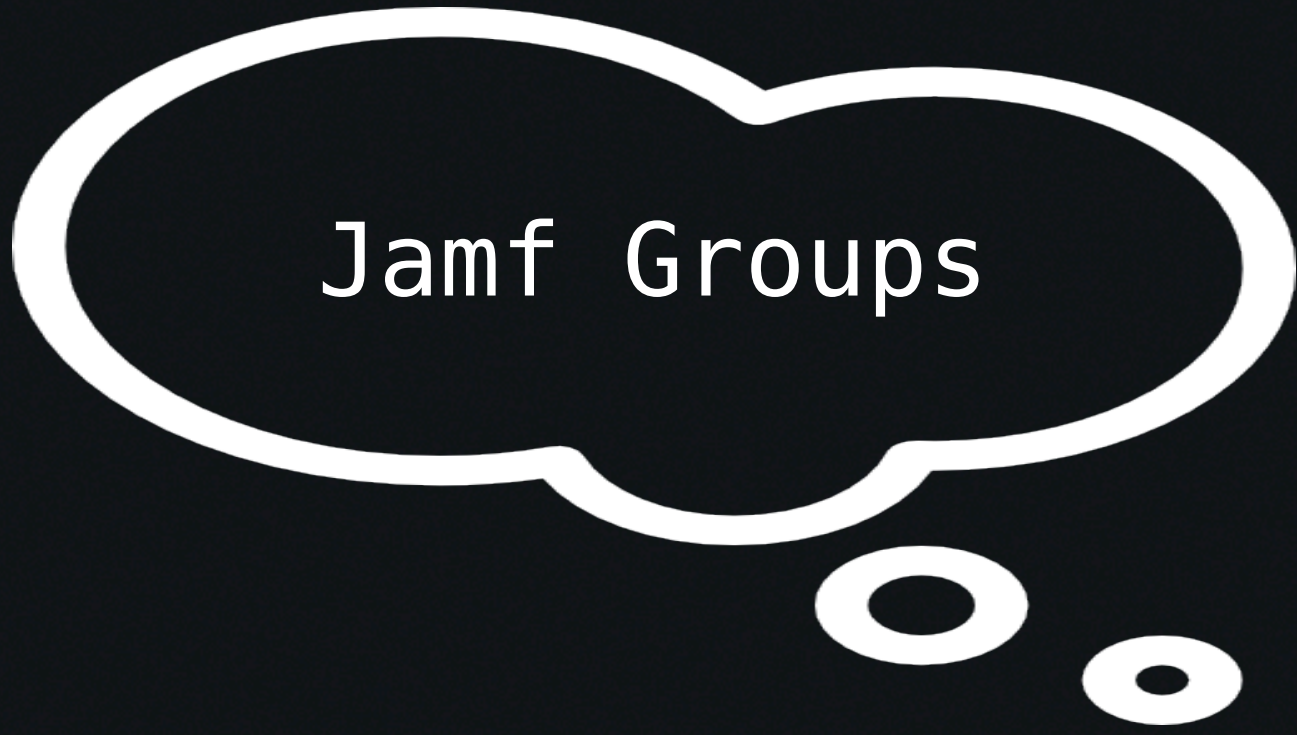
Jamf Policies

Jamf API





Jamf Groups



Computers : Smart Computer Groups

← GitHub Desktop installed

Computer Group Criteria Reports

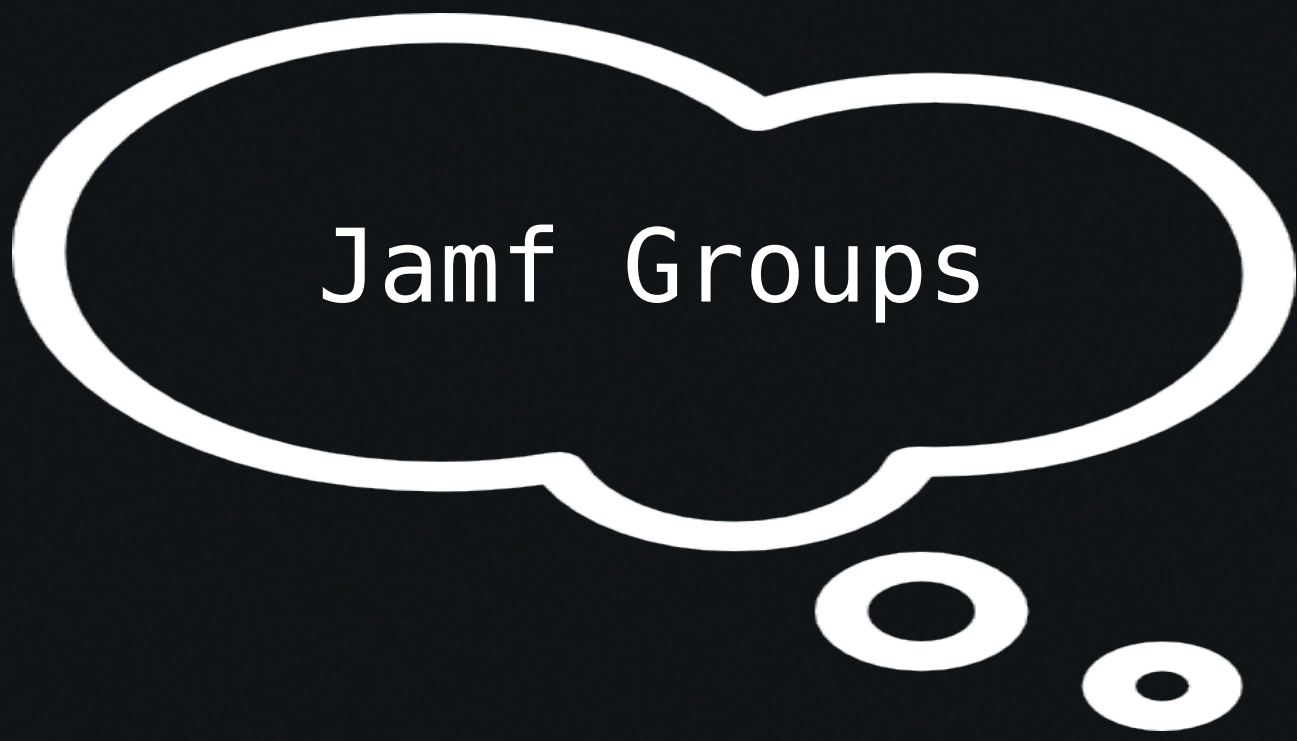
AND/OR	CRITERIA	OPERATOR	VALUE			
	Application Title	is	GitHub Desktop.app	...		Delete

Computers : Smart Computer Groups

← GitHub Desktop current version installed

Computer Group Criteria Reports

AND/OR	CRITERIA	OPERATOR	VALUE			
	Application Title	is	GitHub Desktop.app	...		Delete
and	(matches regex	^\d{2,}.*[4-9].* 3\d{2,}.* 3\.[5-9].			Delete
or)	matches regex	^\$)	Delete



Computers : Smart Computer Groups

← GitHub Desktop installed

Computer Group **Criteria** Reports

AND/OR	CRITERIA	OPERATOR	VALUE
	Application Title	is	GitHub Desktop.app

Computers : Smart Computer Groups

← GitHub Desktop current version installed

Computer Group **Criteria** Reports

AND/OR	CRITERIA	OPERATOR
	Application Title	is
and	(Application Version
		matches regex
or		Application Version
		matches regex

Computers : Smart Computer Groups

← GitHub Desktop test version installed

Computer Group **Criteria** Reports

AND/OR	CRITERIA	OPERATOR	VALUE
	Application Title	is	GitHub Desktop.app
and	(Application Version	matches regex
			^(\d{2},)*[4-9].* 3\d{2},.* 3.[5-9].
or		Application Version	matches regex
			^\$
)			
and		Computer Group	member of
			Testing

State of Installation



Jamf Groups

State of Installation

Jamf Groups

App state

Smart Group scoping

State of Installation



Jamf Groups

App state	Smart Group scoping		
not installed	installed	current version installed	test version installed

State of Installation



Jamf Groups

App state	Smart Group scoping		
not installed	installed	current version installed	test version installed
OLD installed	installed	current version installed	test version installed

State of Installation



Jamf Groups

App state	Smart Group scoping		
not installed	installed	current version installed	test version installed
OLD installed	installed	current version installed	test version installed
PRD installed	installed	current version installed	test version installed

State of Installation

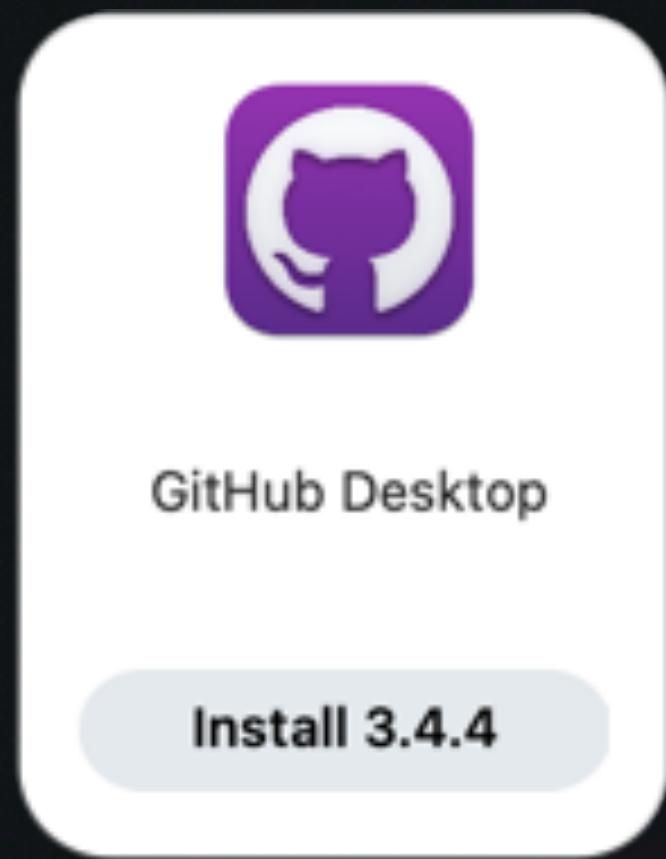


Jamf Groups

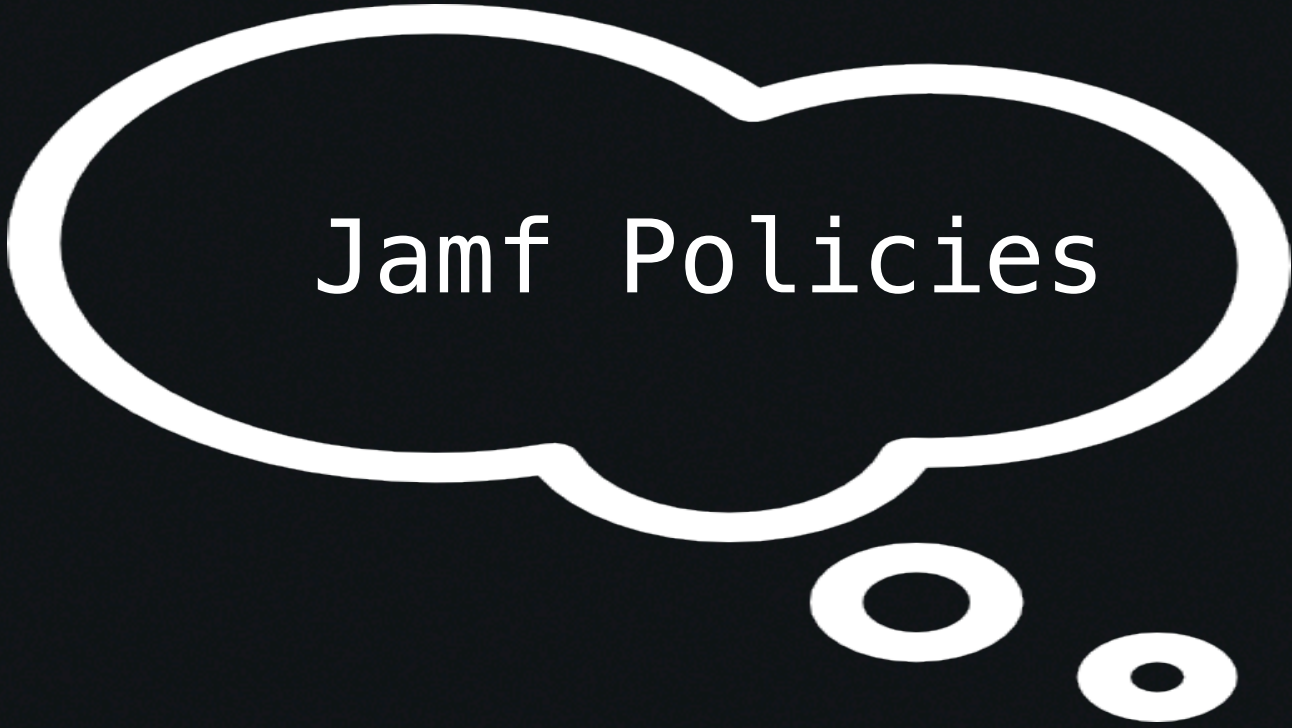
App state	Smart Group scoping		
not installed	installed	current version installed	test version installed
OLD installed	installed	current version installed	test version installed
PRD installed	installed	current version installed	test version installed
TST installed	installed	current version installed	test version installed




Jamf Policies



Install




Jamf Policies



GitHub Desktop

Install 3.4.4

Install




GitHub Desktop

Uninstall

Uninstall


Jamf Policies



GitHub Desktop

Install 3.4.4


Install



GitHub Desktop

Uninstall

Uninstall



GitHub Desktop

Update

Update

Jamf Policies



GitHub Desktop

Install 3.4.4

Install



GitHub Desktop

Uninstall

Uninstall



GitHub Desktop

Update

Update



GitHub Desktop (Testing)

Install 3.4.5

Testing

From State of Installation... to ... Policy

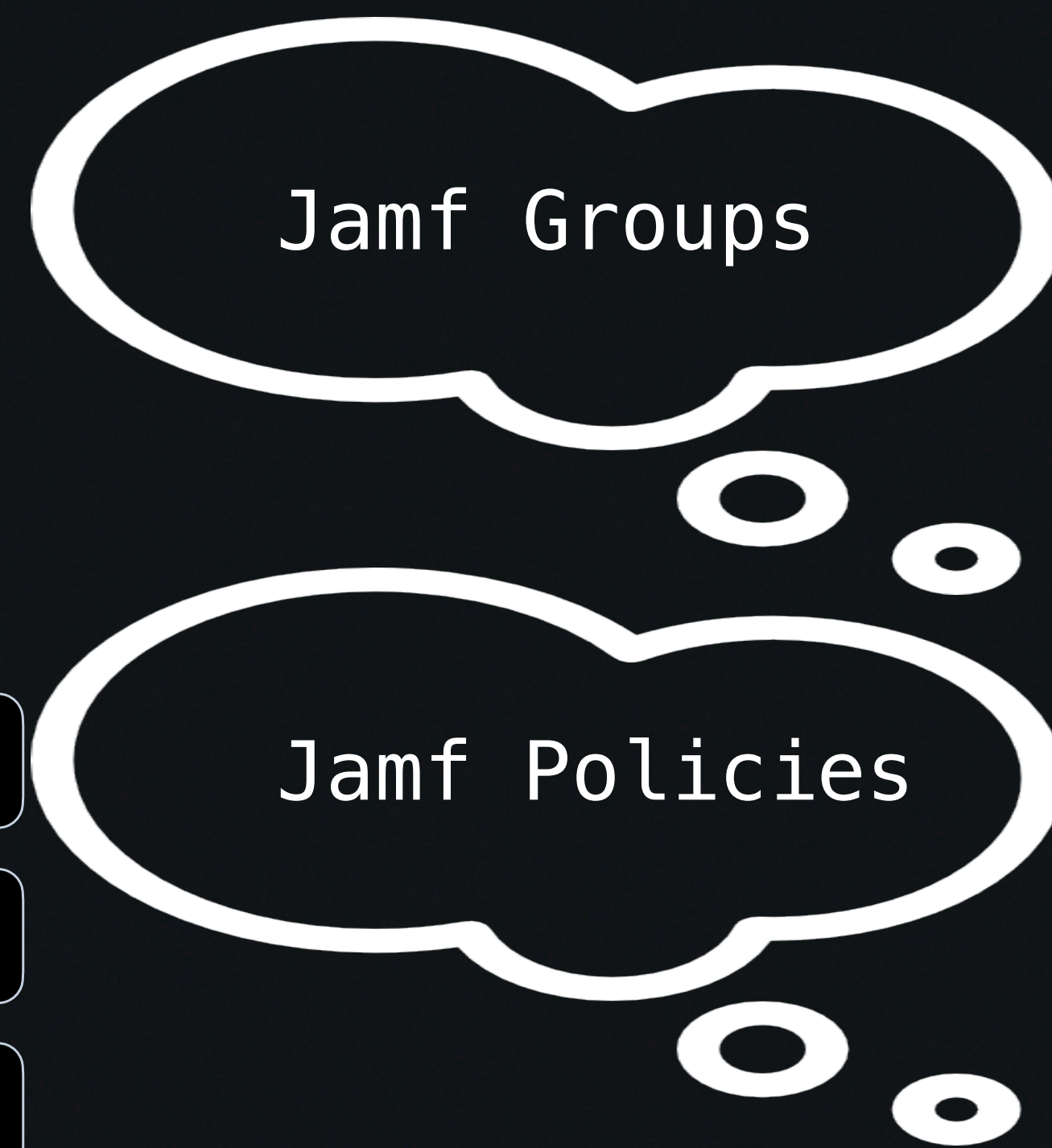


Jamf Groups



Jamf Policies

From State of Installation... to ... Policy



not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

Uninstall

Update

~~Testing~~

~~Install~~

PRD installed

Uninstall

Update

Testing

~~Install~~

OLD installed

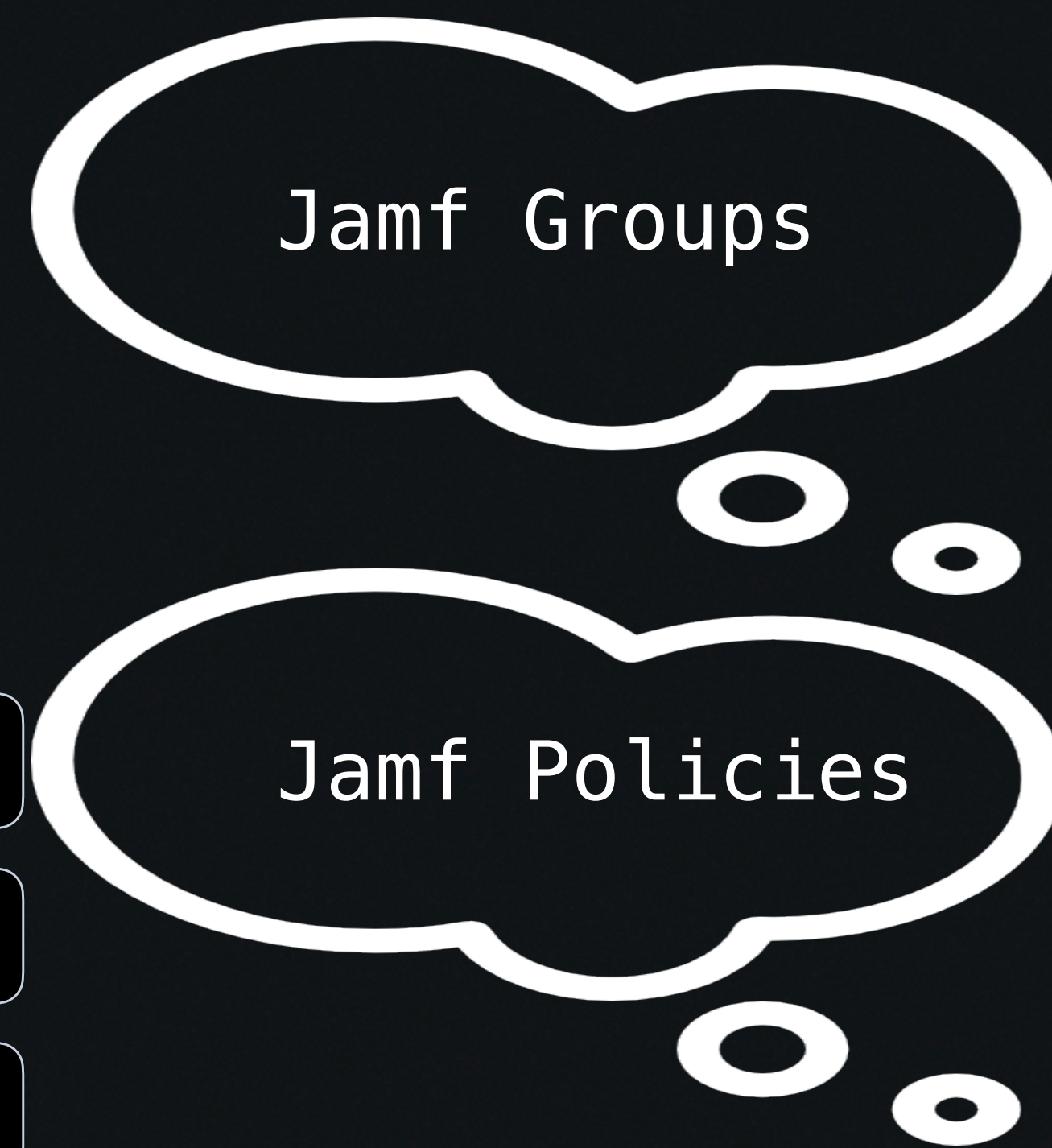
Uninstall

Update

Testing

~~Install~~

From State of Installation... to ... Policy



not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

~~Uninstall~~

~~Update~~

~~Testing~~

~~Install~~

PRD installed

Uninstall

Update

Testing

~~Install~~

OLD installed

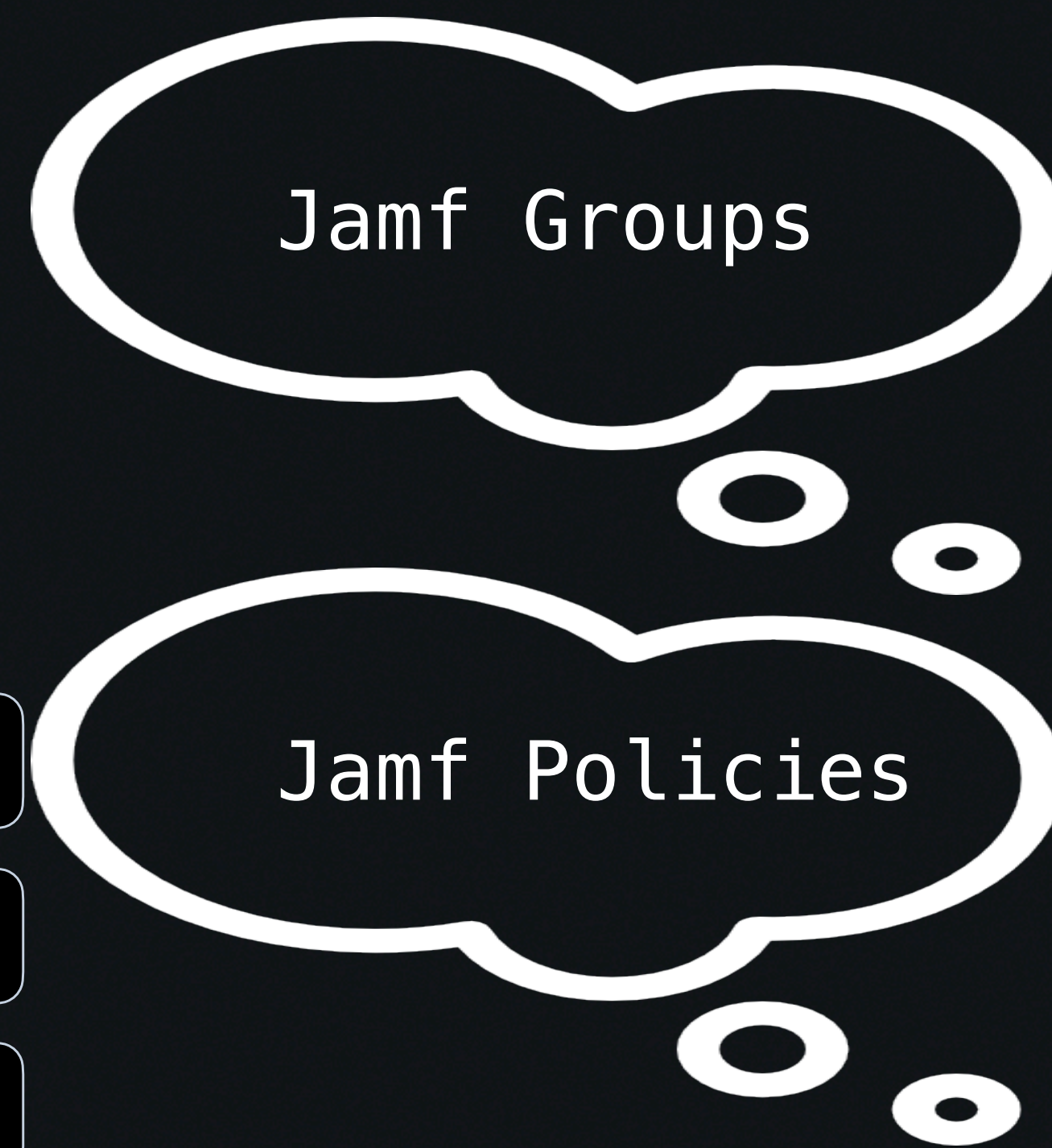
Uninstall

Update

Testing

~~Install~~

From State of Installation... to ... Policy



not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

~~Uninstall~~

~~Update~~

~~Testing~~

~~Install~~

PRD installed

Uninstall

Update

Testing

~~Install~~

OLD installed

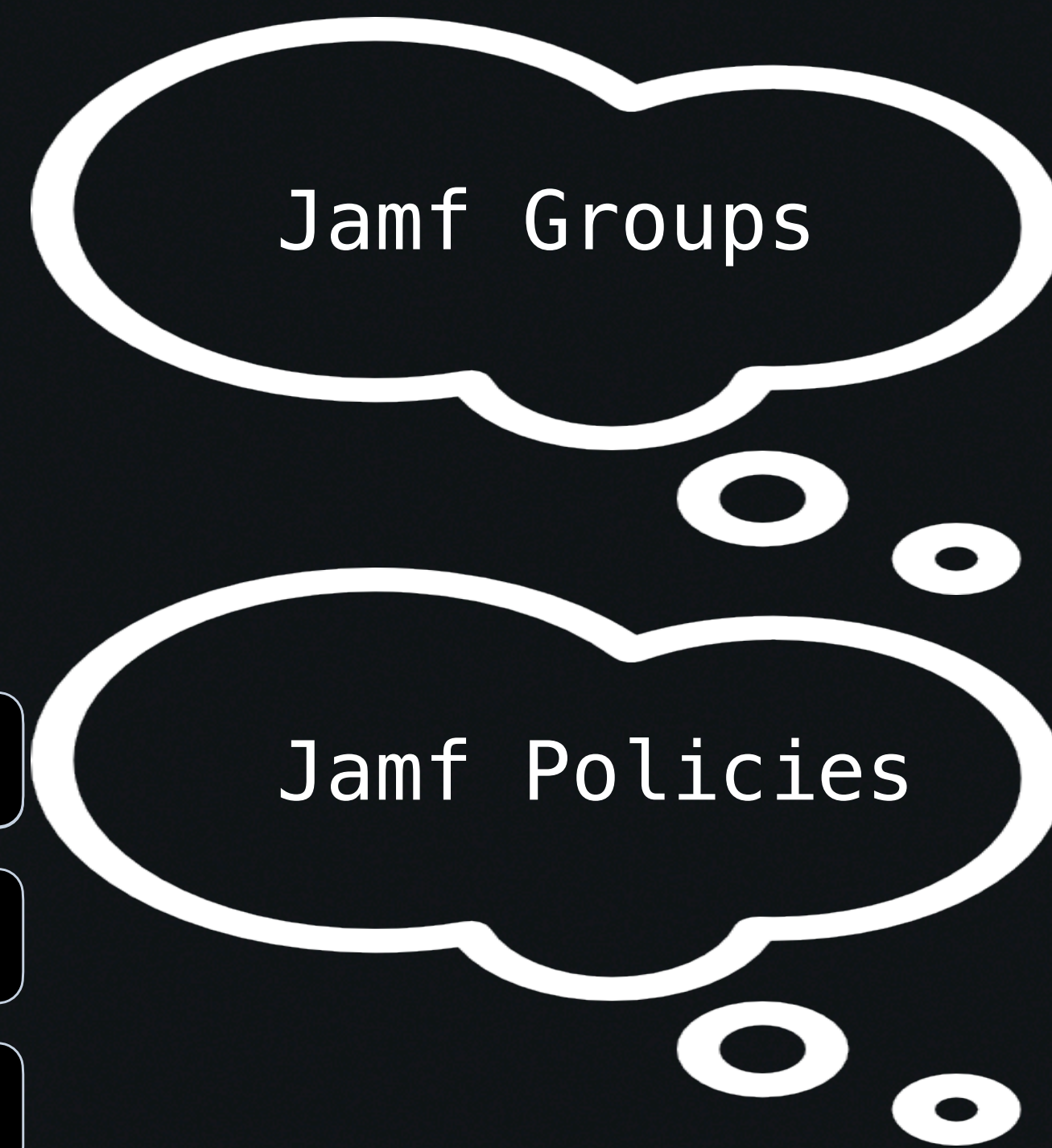
Uninstall

Update

Testing

~~Install~~

From State of Installation... to ... Policy



not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

Uninstall

~~Update~~

~~Testing~~

~~Install~~

PRD installed

Uninstall

~~Update~~

Testing

~~Install~~

OLD installed

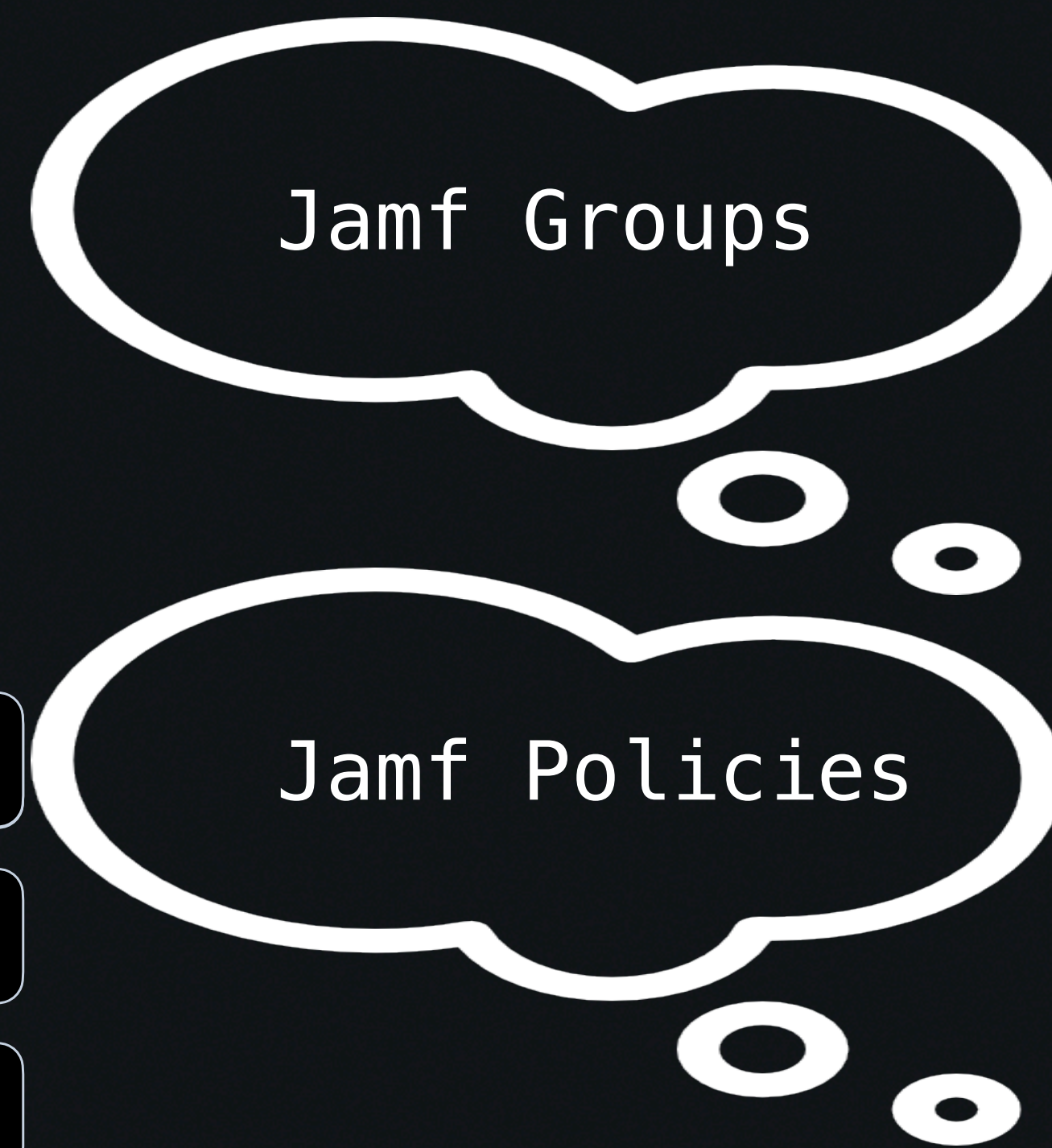
Uninstall

Update

Testing

~~Install~~

From State of Installation... to ... Policy



not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

Uninstall

~~Update~~

~~Testing~~

~~Install~~

PRD installed

Uninstall

~~Update~~

Testing

~~Install~~

OLD installed

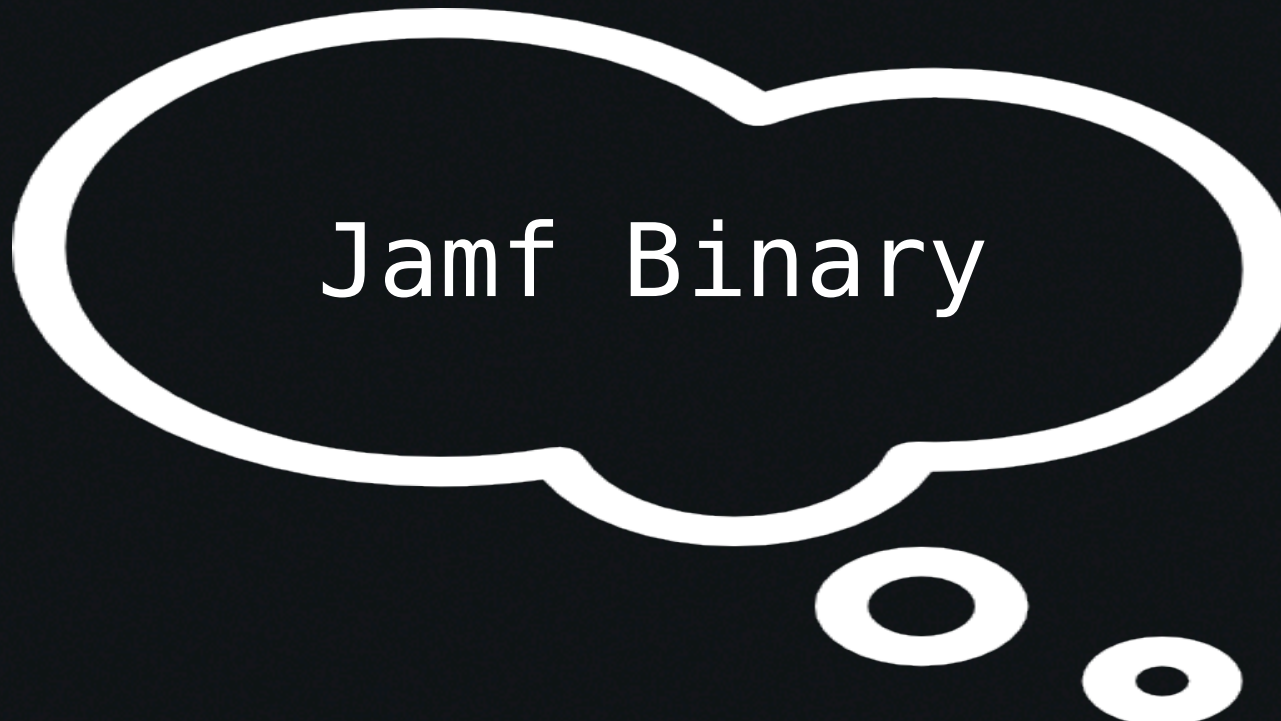
Uninstall

Update

Testing

~~Install~~

```
/usr/local/jamf/bin/jamf
```



Jamf Binary

```
/usr/local/jamf/bin/jamf
```



Jamf Binary

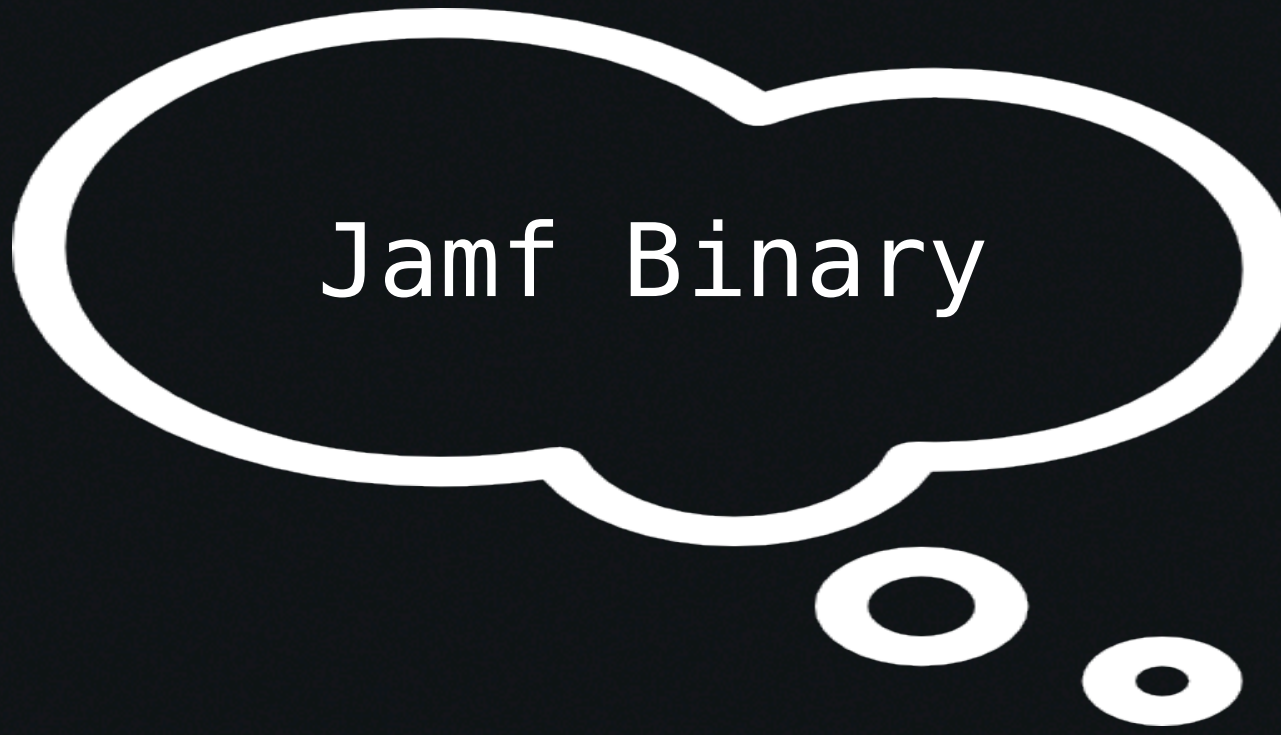
```
% jamf help
```

```
Usage: jamf verb [options]
```

```
verb is one of the following:
```

about	Displays information about the jamf binary
bind	Binds this computer to a directory service
bless	Blesses a System or a NetBoot Server
changePassword	Changes a local user's password
checkJSSConnection	Checks the availability of the JSS
createAccount	Creates a new local account on the system
createConf	Creates a configuration file that the jamf binary uses to find
createSetupDone	Ensures the Setup Assistant does not launch immediately on the
createStartupItem	Creates a startup script to contact the JSS
deleteAccount	Deletes a local account from NetInfo or the local dscl database
deletePrinter	Deletes a printer from the system
deleteSetupDone	Causes the Setup Assistant to launch on the next boot
displayMessage	Displays a message to the current user
enablePermissions	Enables permissions on a volume
enroll	Enrolls this computer with the JSS

jamf recon



Jamf Binary

jamf recon



Jamf Binary

```
% sudo jamf recon
Password:

Retrieving inventory preferences from https://ethztst0.jamfcloud.com...
Finding extension attributes...
Locating package receipts...
Locating accounts...
Locating hard drive information...
Locating applications...
Searching path: /System/Applications
Locating printers...
Searching path: /Applications
Locating hardware information (macOS 14.6.1)...
Submitting data to https://ethztst0.jamfcloud.com ...
<computer_id>94</computer_id>
%
```

jamf policy

The output of a “Successful Policy Execution”



Jamf Binary

jamf policy



Jamf Binary

The output of a “Successful Policy Execution”

```
% sudo jamf policy -id 6829

Checking for policy ID 6829...
Executing Policy GitHub Desktop
Running command jamf policy -event "GitHub Desktop-install"...
Result of command:
Checking for policies triggered by "GitHub Desktop-install" for user "test"...
Executing Policy Install GitHub Desktop
Downloading GitHub_Desktop-Universal-3.4.4.pkg...
Downloading https://id-jamf-prd.ethz.ch/JPSHare/Packages/GitHub_Desktop_Universal-3.4.5.pkg.
Verifying package integrity...
Installing GitHub_Desktop-Universal-3.4.4.pkg...
Successfully installed GitHub_Desktop-Universal-3.4.4.pkg.
Running command echo 'Installation of GitHub Desktop complete'...
Result of command:
Installation of GitHub Desktop complete
Running Recon...
Submitting log to https://ethztst0.jamfcloud.com
%
```

jamf policy

The output of a “Failed Policy Execution”



Jamf Binary

jamf policy



Jamf Binary

The output of a “Failed Policy Execution”

```
% sudo jamf policy -id 6829  
Checking for policy ID 6829...  
No policies were found for the ID 6829.
```

jamf policy



Jamf Binary

The output of a “Failed Policy Execution”

```
% sudo jamf policy -id 6829
```

```
Checking for policy ID 6829...
```

```
No policies were found for the ID 6829.
```

```
Error running script: return code was 1.
```

```
Error: This computer does not meet the OS Requirements
```

```
Installation failed. The installer reported: installer: Error - SWITCHdrive Client can't be installed on this computer.
```

```
Installation failed. The package could not be verified.
```

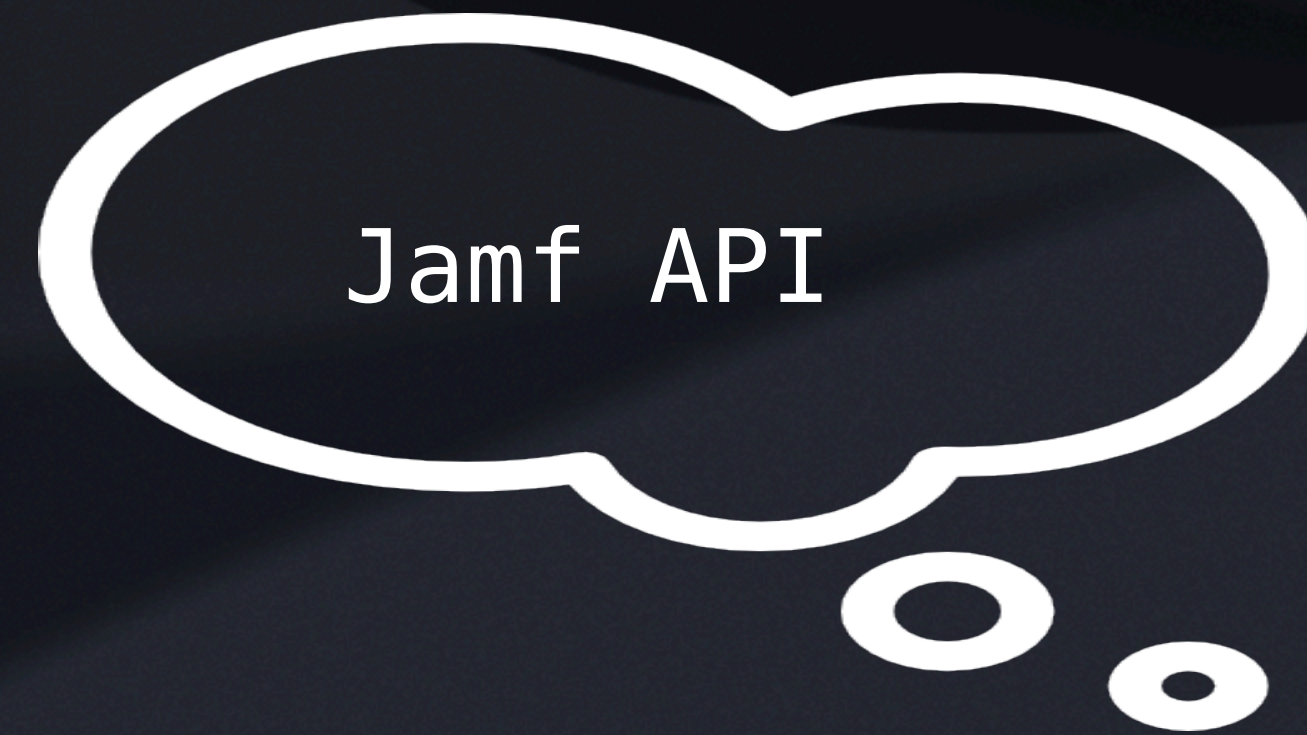
<https://developer.jamf.com/jamf-pro/reference/classic-api>



Jamf API

https://developer.jamf.com/jamf-pro/reference/classic-api

The screenshot shows the Jamf API reference page for the endpoint "Finds all policies". The page is viewed in a browser at developer.jamf.com. The navigation bar includes the Jamf logo, version "v11.9.0", and links for Home, Guides, Recipes, API Reference, and Changelog. A search bar is located on the right. The left sidebar lists various API endpoints under the "policies" category, with "Finds all policies" selected. The main content area displays the endpoint name, a GET method, and the URL: https://yourServer.jamfcloud.com/JSSResource/policies. Below this, a "RESPONSE" section shows a 200 OK status. A "LANGUAGE" selector is available, with options for Shell, Swift, Ruby, and Python. A "CURL REQUEST" section provides a curl command: curl --request GET \ --url https://yourserver.jamfcloud.com/JSSResour \ --header 'accept: application/json'. At the bottom, there are tabs for "RESPONSE" and "EXAMPLES", with the "RESPONSE" tab active, showing "Choose an example:" and "application/json" with a 200 status indicator.



Putting it into code - example

Get Policy ID

```
curl_args=(
  --location
  --request GET
  --header "Authorization: Bearer $token"
  --header "Accept: application/xml"
  --output "/tmp/output-policies.txt"
  "${jss_url}/JSSResource/policies"
)
curl "${curl_args[@]}"

policy_id=$(
  xmllint --xpath "//policy[name='${app_name} (Testing)']/id/text()" \
  "$tmp_folder/output-policies.txt"
)
```

Putting it into code - example

Get Policy ID

```
curl_args=(
  --location
  --request GET
  --header "Authorization: Bearer $token"
  --header "Accept: application/xml"
  --output "/tmp/output-policies.txt"
  "${jss_url}/JSSResource/policies"
)
curl "${curl_args[@]}"

policy_id=$(
  xmllint --xpath "//policy[name='${app_name} (Testing)']/id/text()" \
  "$tmp_folder/output-policies.txt"
)
```

Putting it into code - example

Get Policy ID

```
curl_args=(  
  --location  
  --request GET  
  --header "Authorization: Bearer $token"  
  --header "Accept: application/xml"  
  --output "/tmp/output-policies.txt"  
  "${jss_url}/JSSResource/policies"  
)  
curl "${curl_args[@]}"
```

```
policy_id=$(  
  xmllint --xpath "//policy[name='${app_name} (Testing)']/id/text()" \  
  "$tmp_folder/output-policies.txt"  
)
```

https://developer.jamf.com/jamf-pro/reference/classic-api

The screenshot shows the Jamf API reference page for the endpoint "Finds computer groups by name". The page is part of the Jamf v11.9.0 documentation. The main content area displays the endpoint details, including the HTTP method (GET), the URL template, and the path parameter "name" which is a required string. The response is shown as a 200 OK status. A curl request is provided for the endpoint. The page also includes a sidebar with a list of other API endpoints and a language selector for the curl request.

jamf API References Documentation

v11.9.0 Home Guides Recipes <> API Reference << Changelog Search

computergroups

- Finds all computer groups GET
- Finds computer groups by ID GET
- Updates an existing computer group by ID PUT
- Creates a new computer group by ID POST
- Deletes a computer group by ID DELETE
- Finds computer groups by name GET**
- Updates an existing computer group by name PUT
- Deletes a computer group by name DELETE

computerhardwaresoftwarereports

computerhistory

computerinventorycollection

computerinvitations

computermanagement

Finds computer groups by name

GET https://yourServer.jamfcloud.com/JSSResource/computergroups/name/{name}

PATH PARAMS

name string required
Name to filter by

RESPONSE

200 OK

Updated about 1 year ago

← Creates a new computer group by ID Updates an existing computer group by name →

LANGUAGE

Shell Swift Ruby Python

CURL REQUEST

```
1 curl --request GET \  
2 --url https://yourserver.jamfcloud.com/JSSResour \  
3 --header 'accept: application/json'
```

RESPONSE EXAMPLES

Choose an example:

application/json 200

Jamf API

Putting it into code - example

Get current group membership

```
curl_args=(
    --location
    --request GET
    --header "Authorization: Bearer $token"
    --header "Accept: application/xml"
    --output "$tmp_folder/output-computergroups.txt"
    "${jss_url}/JSSResource/computergroups/name/${app_name_encoded}%20installed"
)

group_installed_count=$(
    curl "${curl_args[@]}" | \
    xmllint --xpath "//computer_group/computers/computer/id/text()" - 2> /dev/null | \
    grep -xc "${computer_id}"
)
```

Putting it into code - example

Get current group membership

```
curl_args=(
    --location
    --request GET
    --header "Authorization: Bearer $token"
    --header "Accept: application/xml"
    --output "$tmp_folder/output-computergroups.txt"
    "${jss_url}/JSSResource/computergroups/name/${app_name_encoded}%20installed"
)

group_installed_count=$(
    curl "${curl_args[@]}" | \
    xmllint --xpath "//computer_group/computers/computer/id/text()" - 2> /dev/null | \
    grep -xc "${computer_id}"
)
```

Putting it into code - example

Get current group membership

```
curl_args=(
  --location
  --request GET
  --header "Authorization: Bearer $token"
  --header "Accept: application/xml"
  --output "$tmp_folder/output-computergroups.txt"
  "${jss_url}/JSSResource/computergroups/name/${app_name_encoded}%20installed"
)

group_installed_count=$(
  curl "${curl_args[@]}" | \
  xmllint --xpath "//computer_group/computers/computer/id/text()" - 2> /dev/null | \
  grep -xc "${computer_id}"
)
```



Jamf Groups

Jamf Binary

Jamf Policies

Jamf API

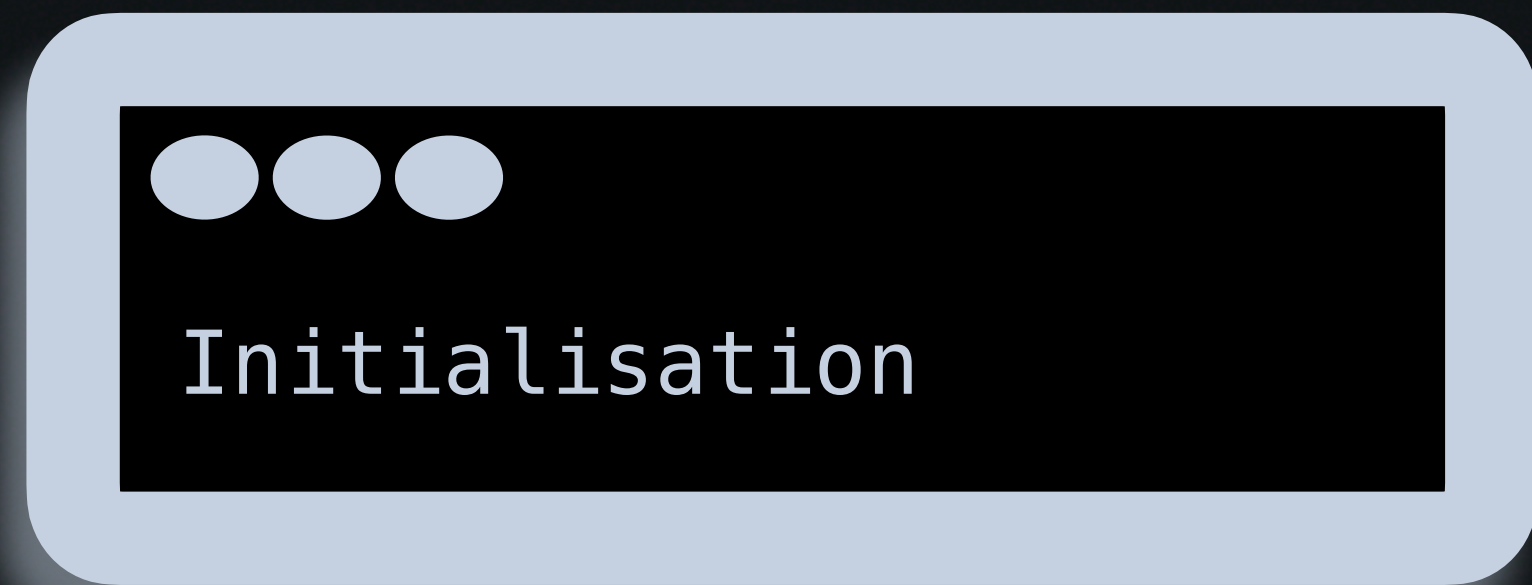


Putting it into code

Using Scripts

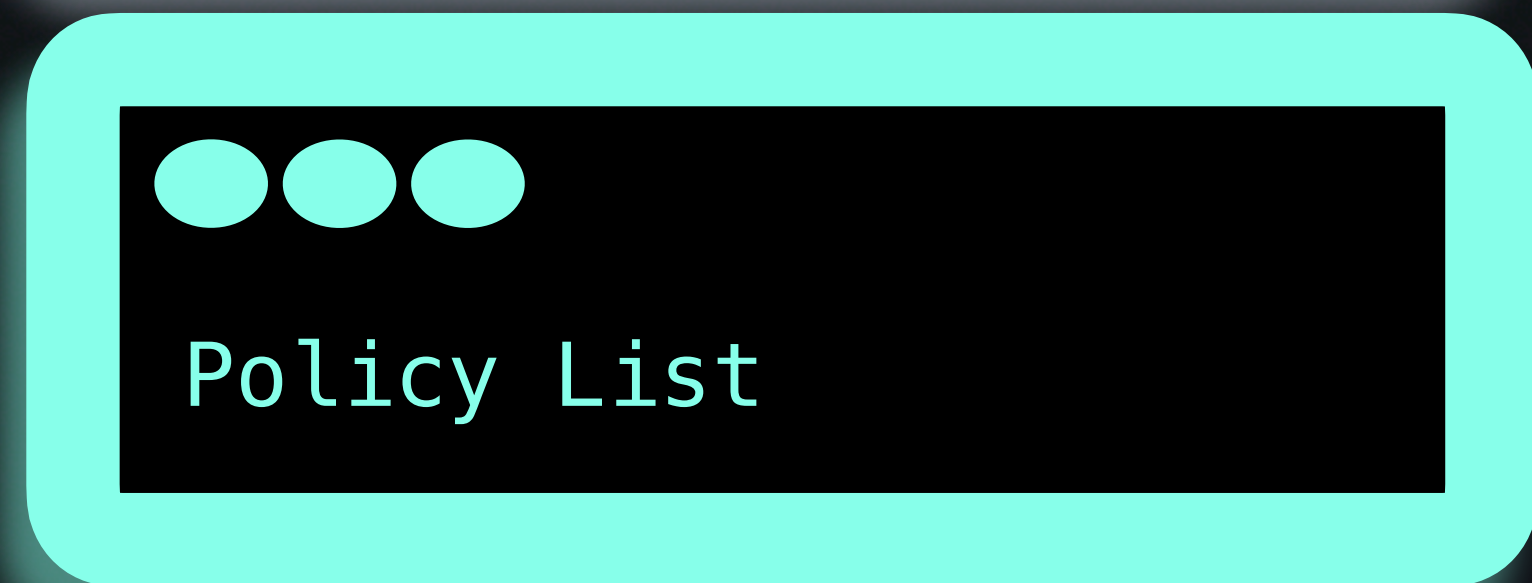
Putting it into code

Using Scripts



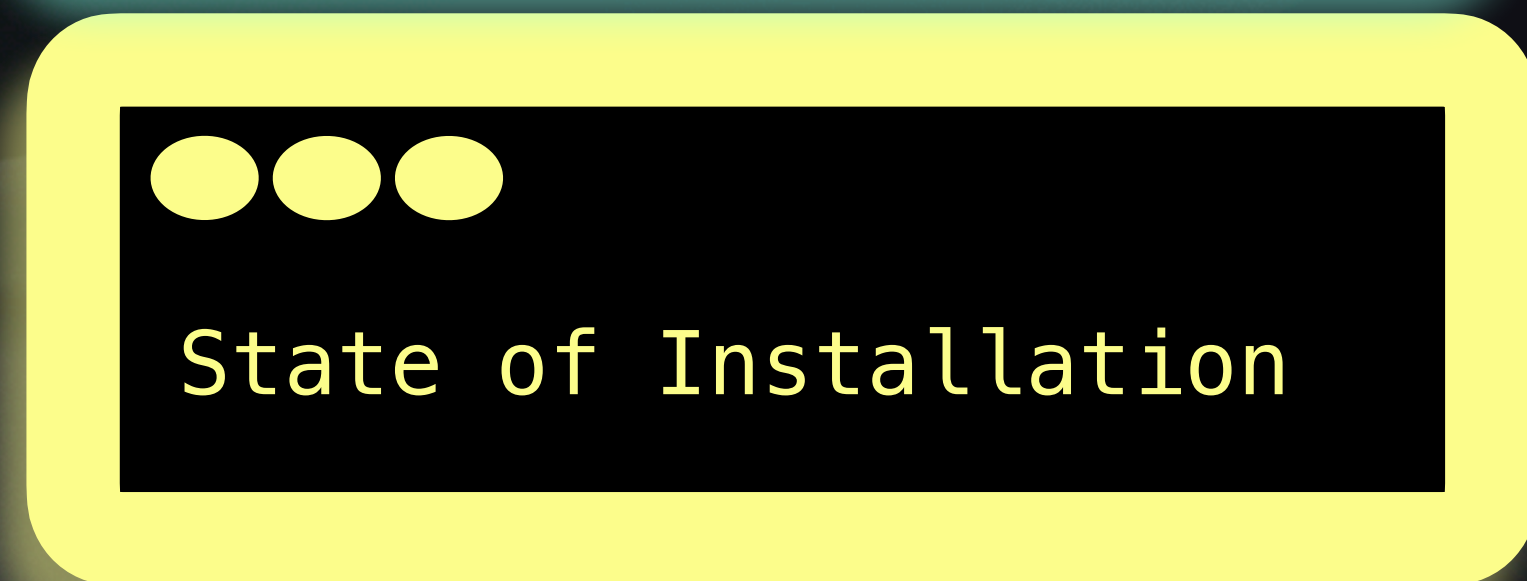
Putting it into code

Using Scripts



Putting it into code

Using Scripts



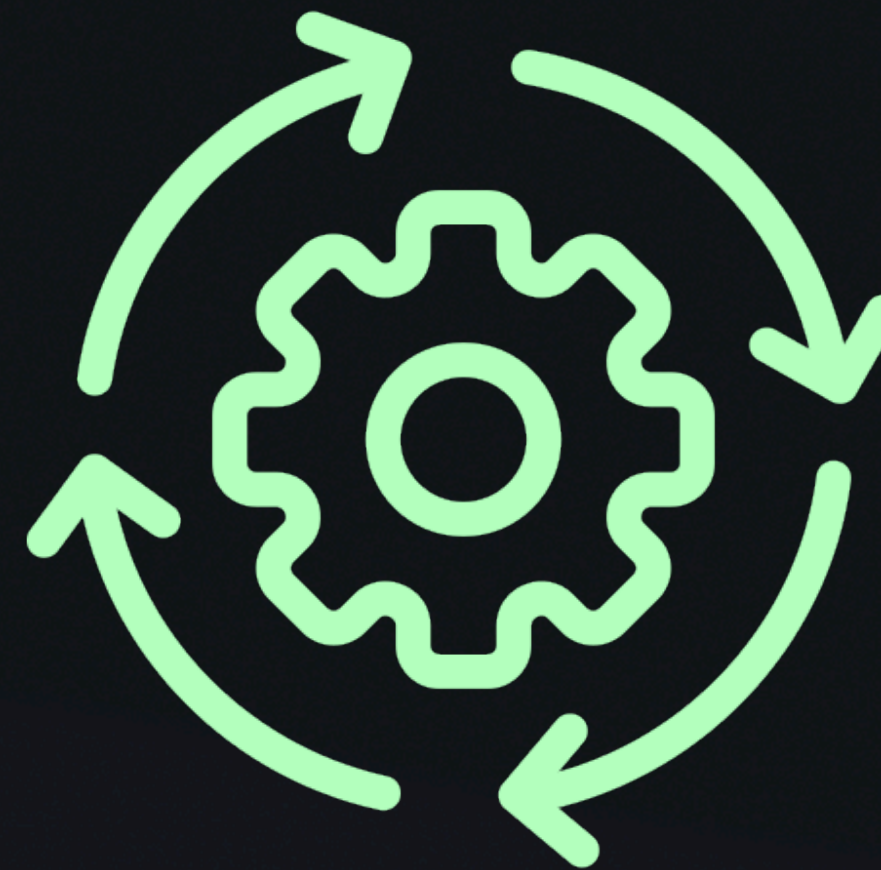
Putting it into code

Using Scripts



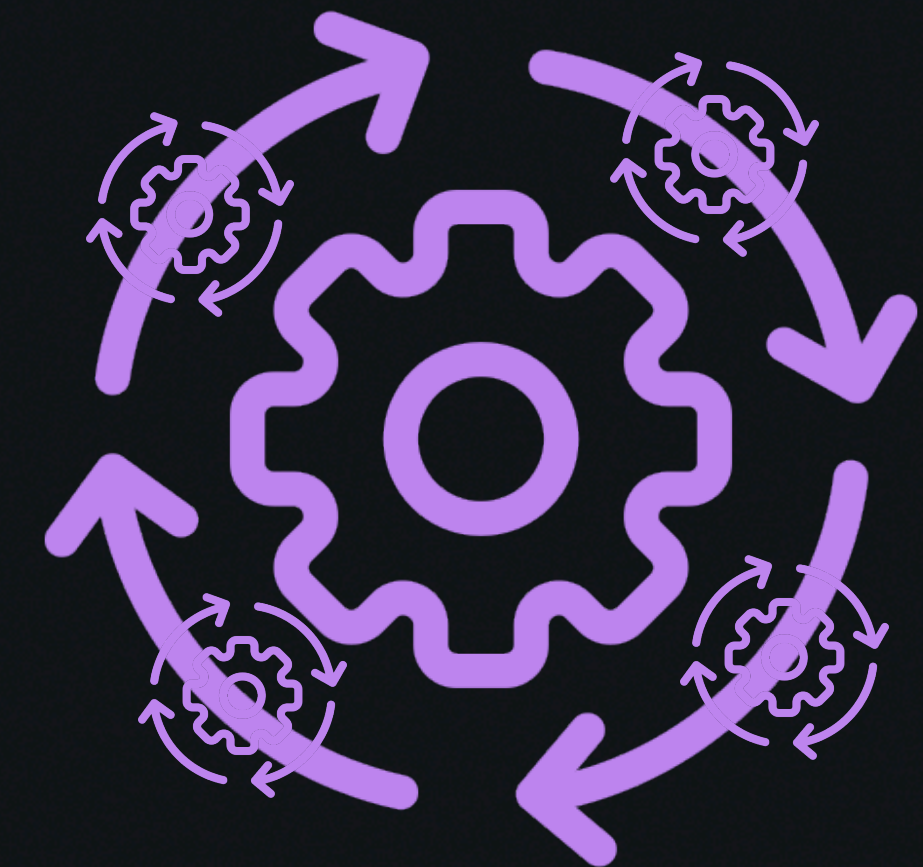
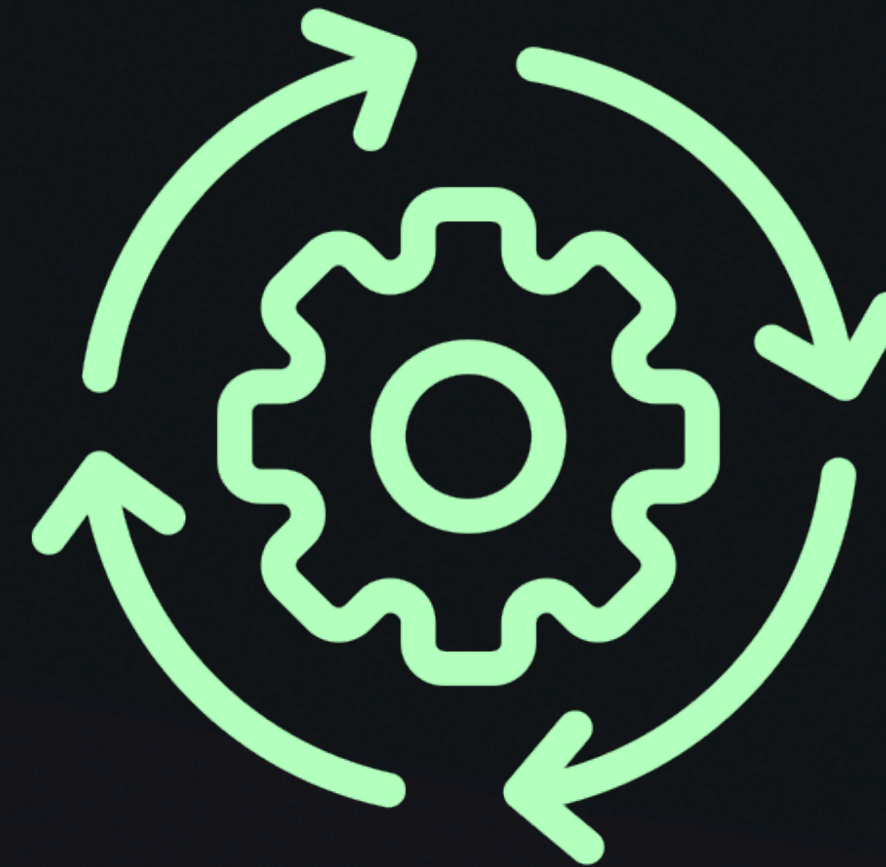
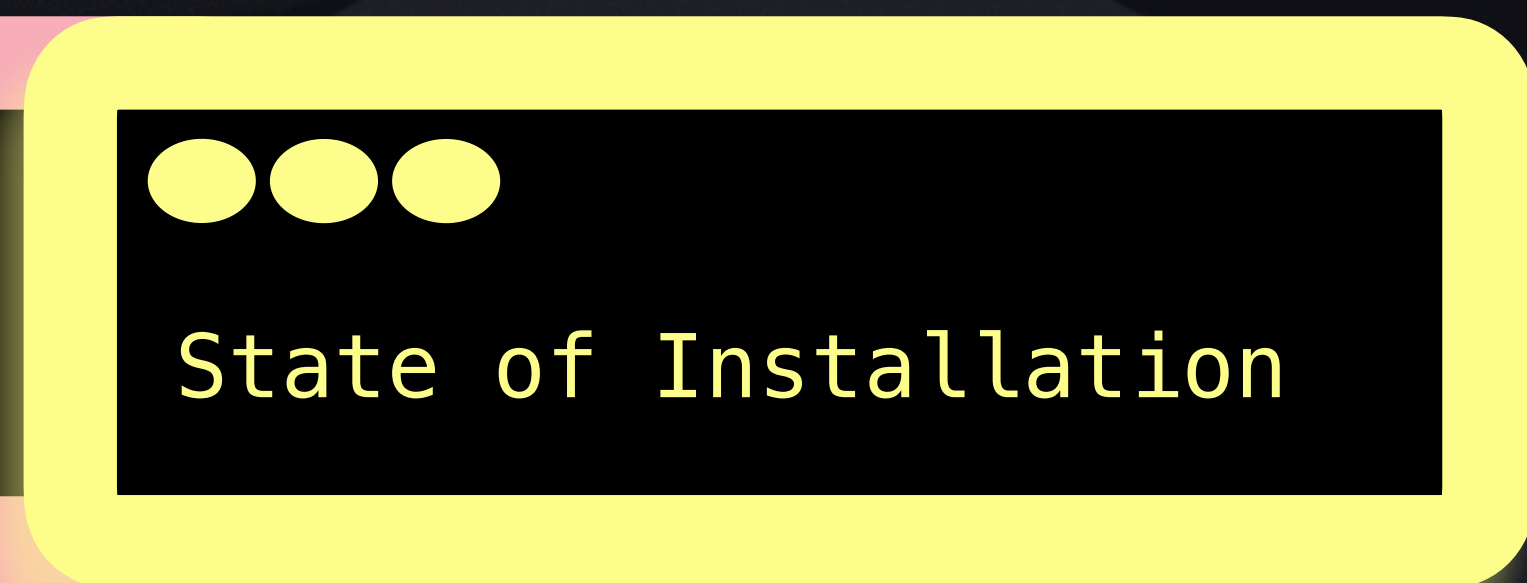
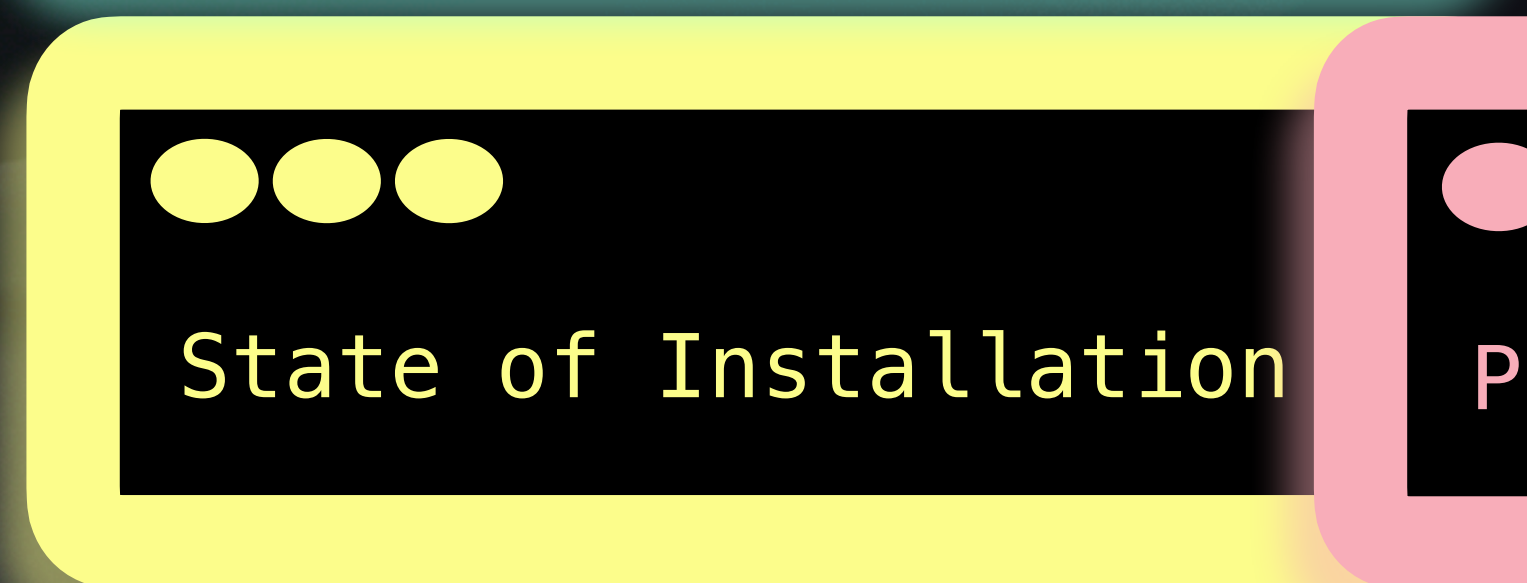
Putting it into code

Using Scripts

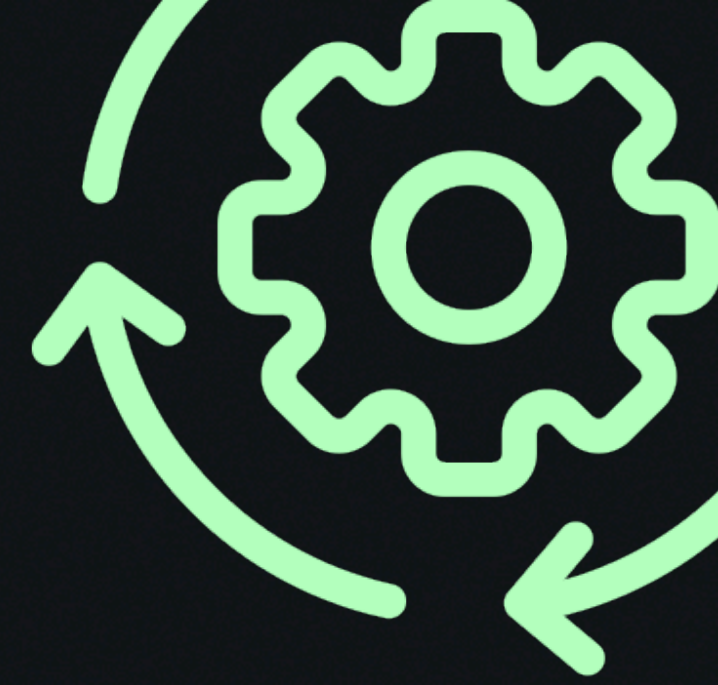


Putting it into code

Using Scripts

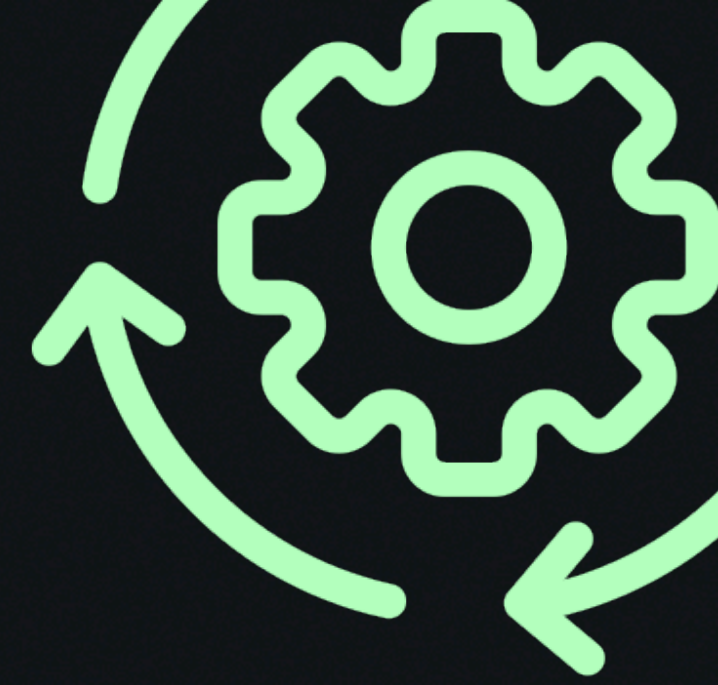


Results of tests - each single scenario



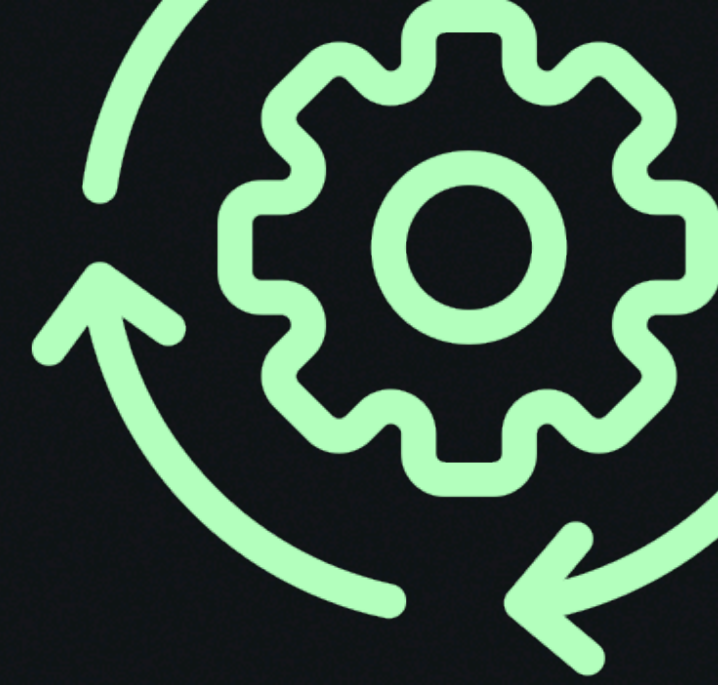
Results of tests - each single scenario

PASS or FAIL?



Results of tests - each single scenario

PASS or FAIL?



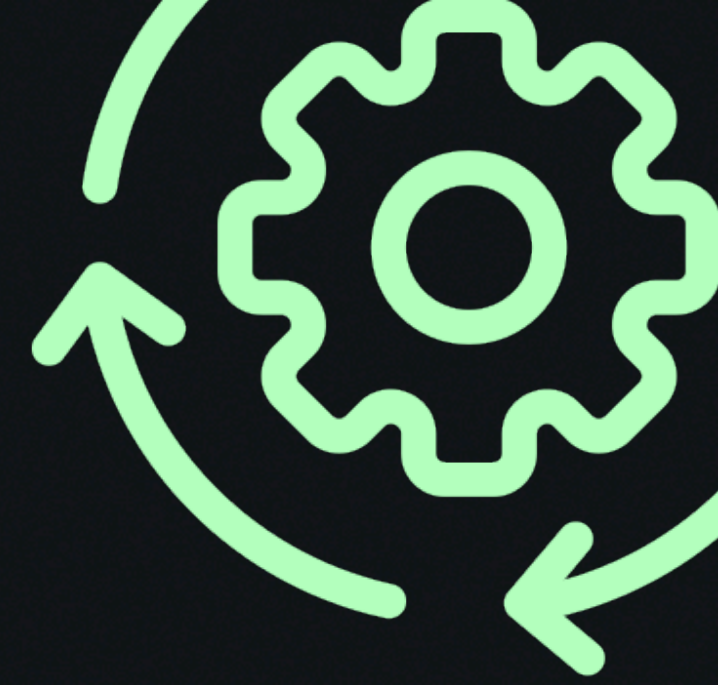
not installed

(Testing)

PRD installed

OLD installed

Results of tests - each single scenario



PASS or FAIL?

not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

Uninstall

Update

Testing

Install

PRD installed

Uninstall

Update

Testing

Install

OLD installed

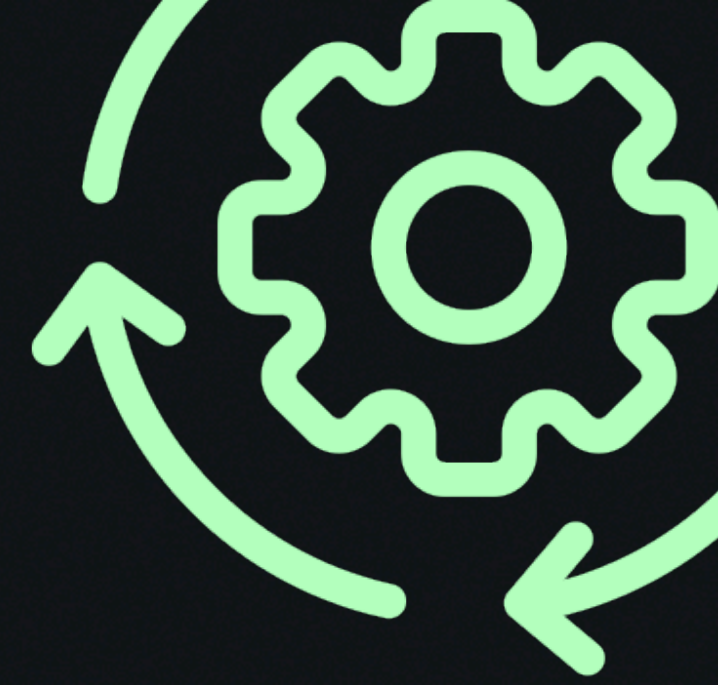
Uninstall

Update

Testing

Install

Results of tests - each single scenario



PASS or FAIL?

not installed

~~Uninstall~~

~~Update~~

Testing

Install

(Testing)

Uninstall

Update

Testing

Install

PRD installed

Uninstall

Update

Testing

Install

OLD installed

Uninstall

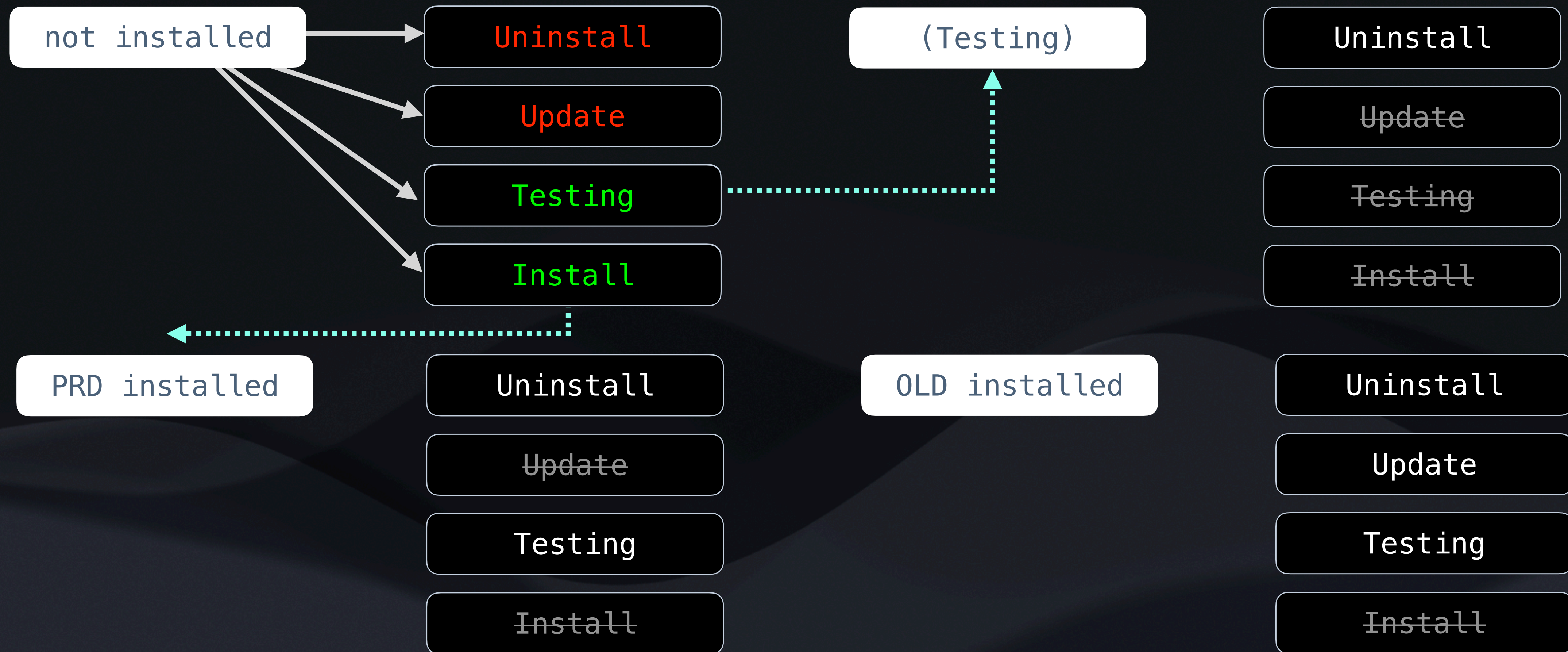
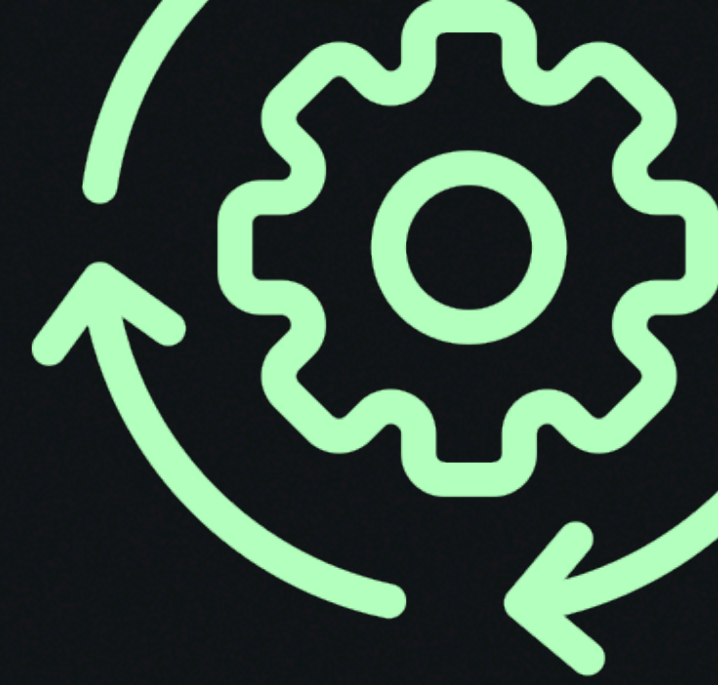
Update

Testing

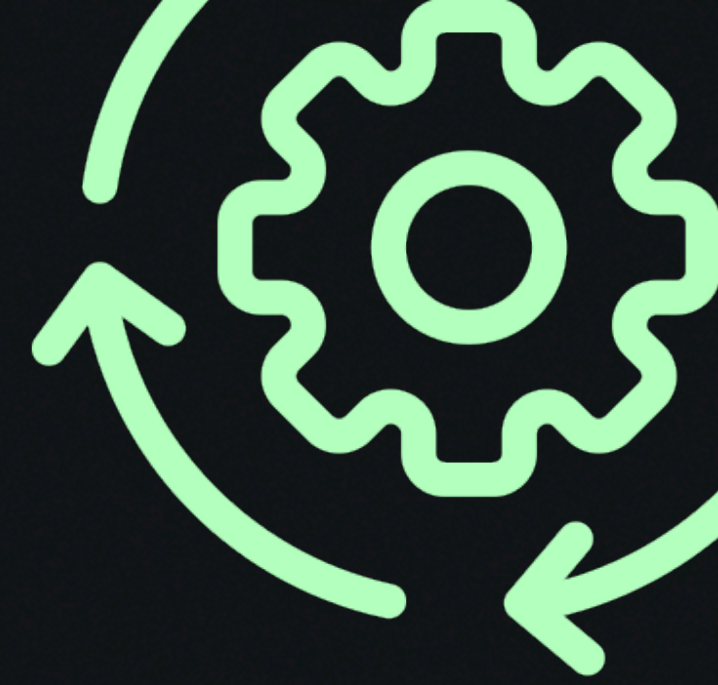
Install

Results of tests - each single scenario

PASS or FAIL?



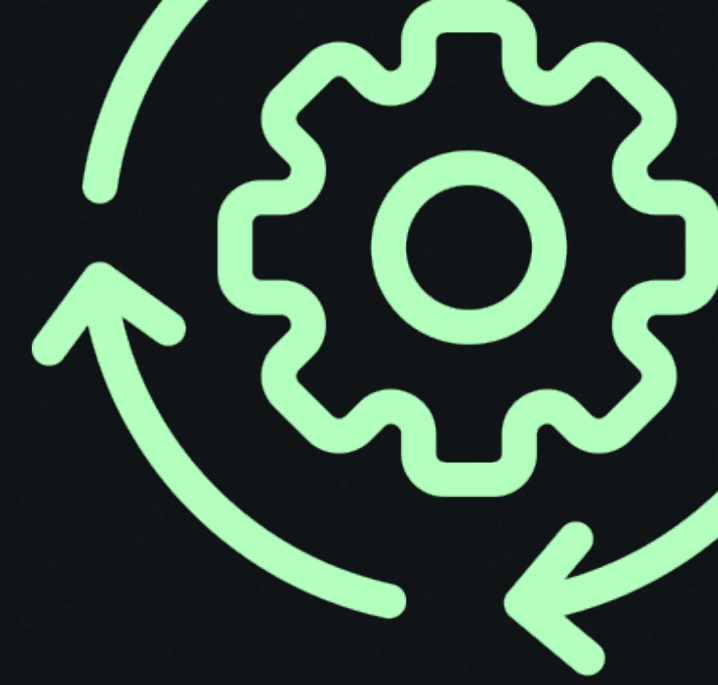
Results of tests - each single scenario



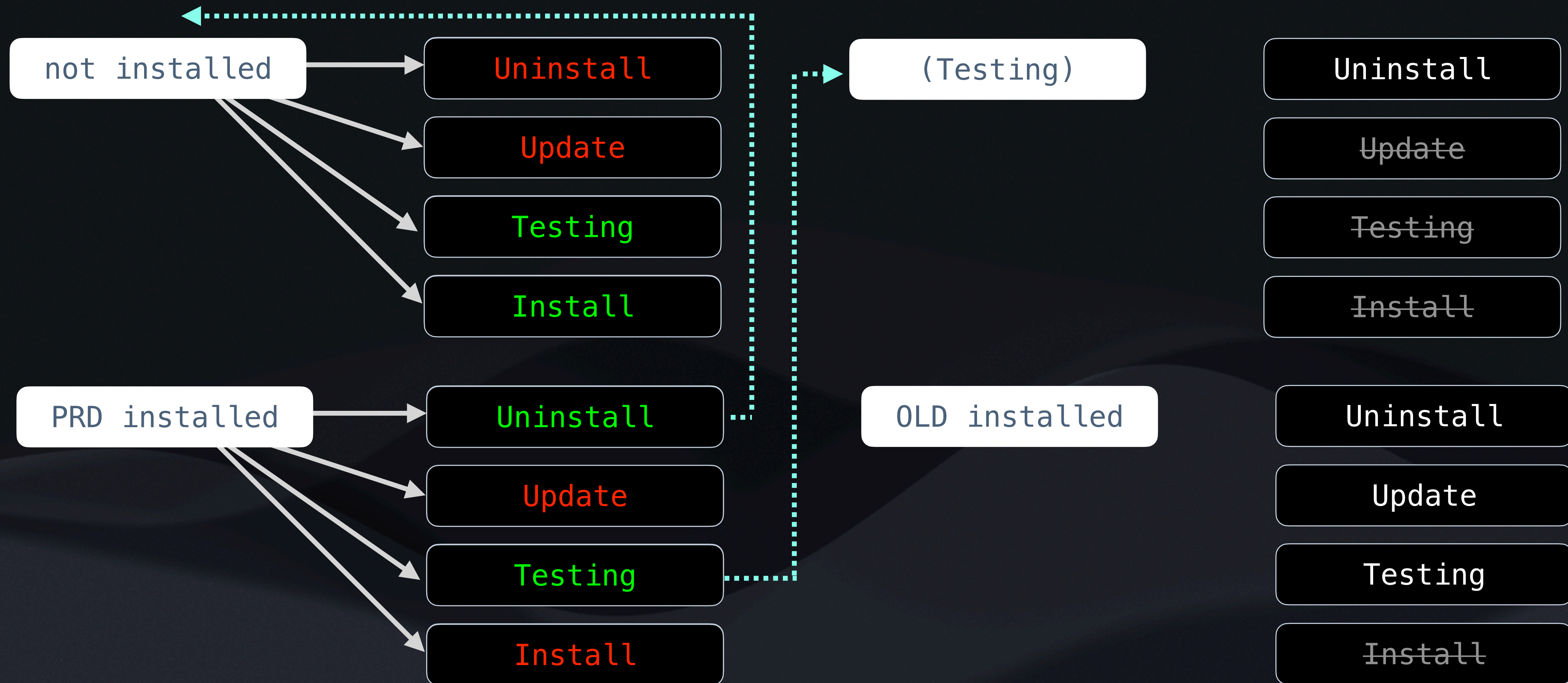
PASS or FAIL?



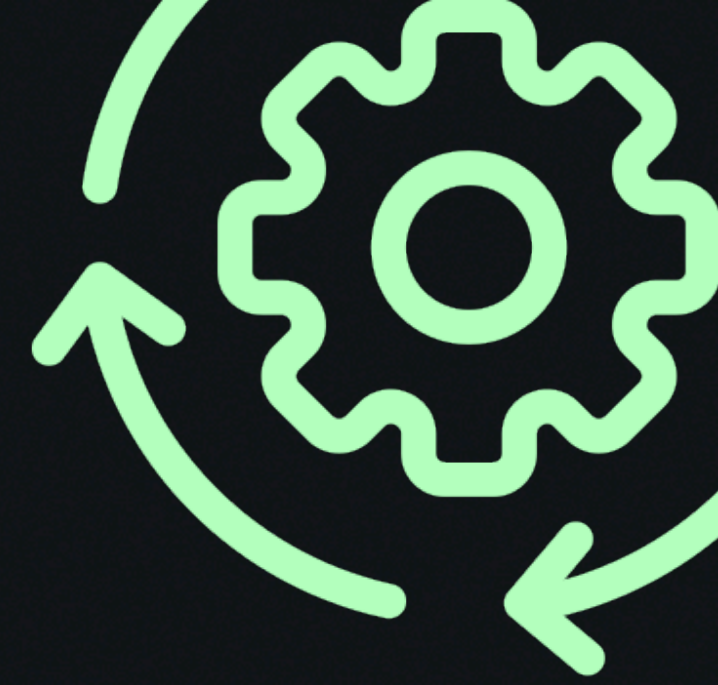
Results of tests - each single scenario



PASS or FAIL?



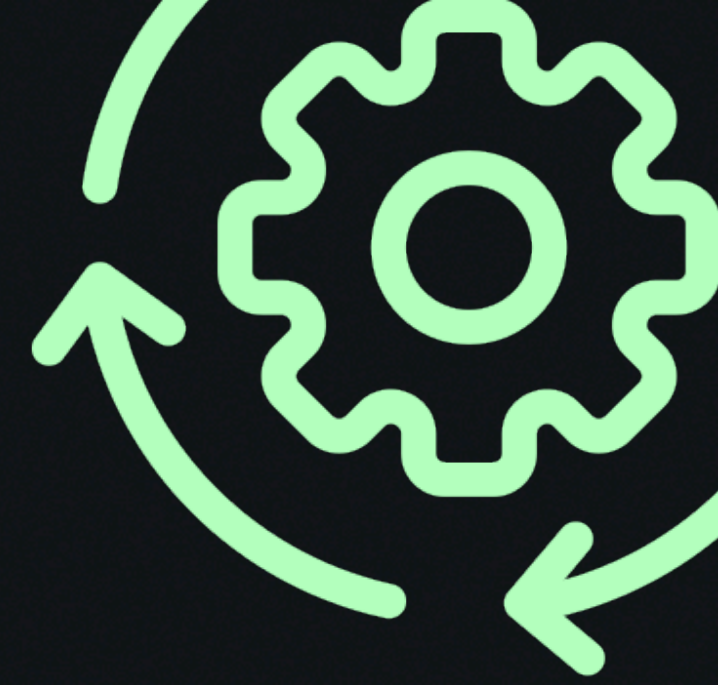
Results of tests - each single scenario



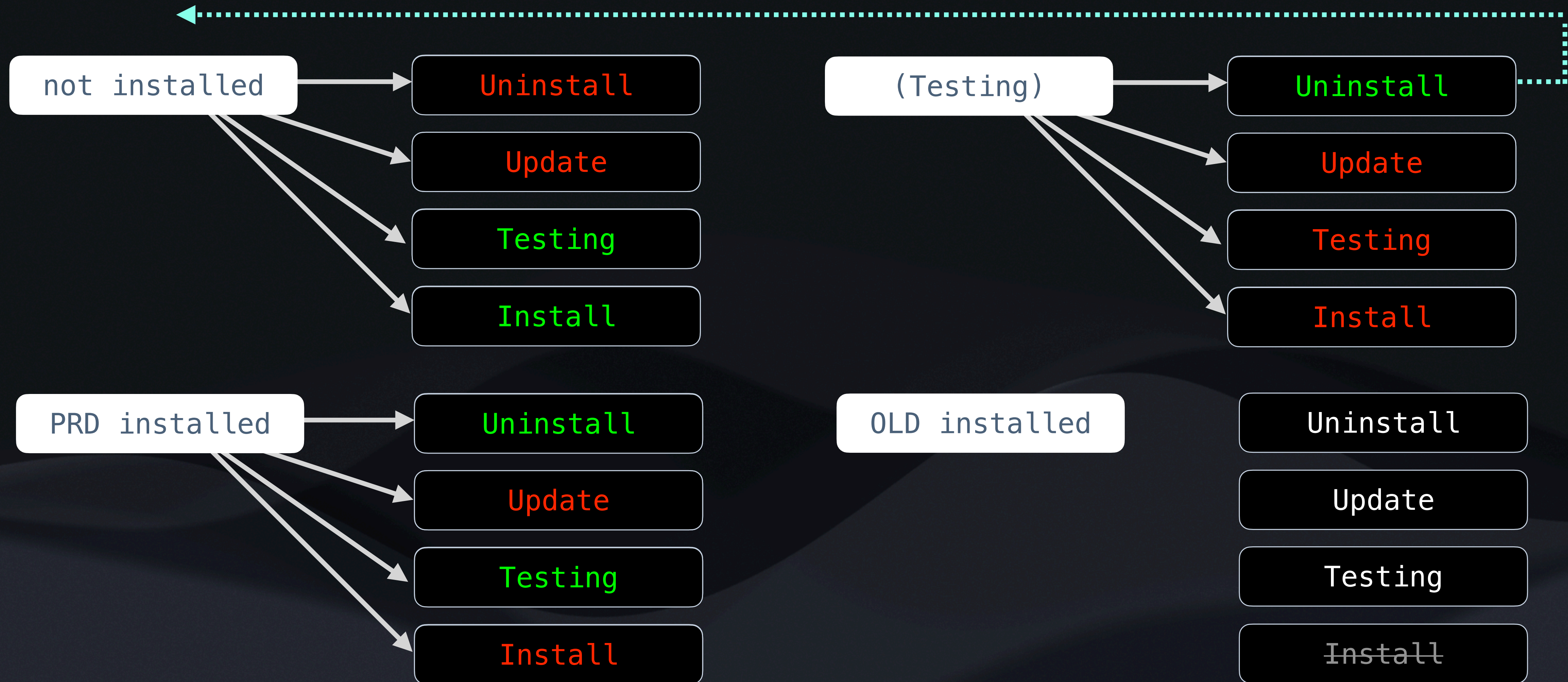
PASS or FAIL?



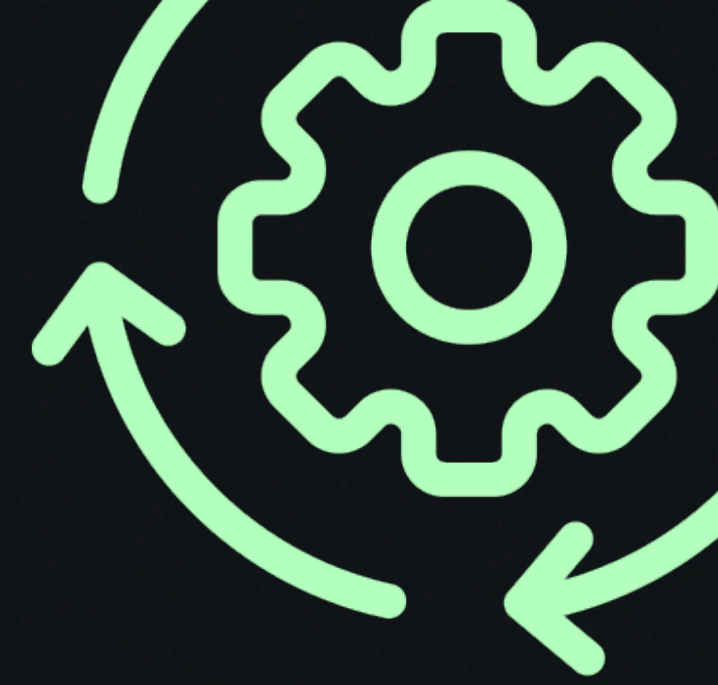
Results of tests - each single scenario



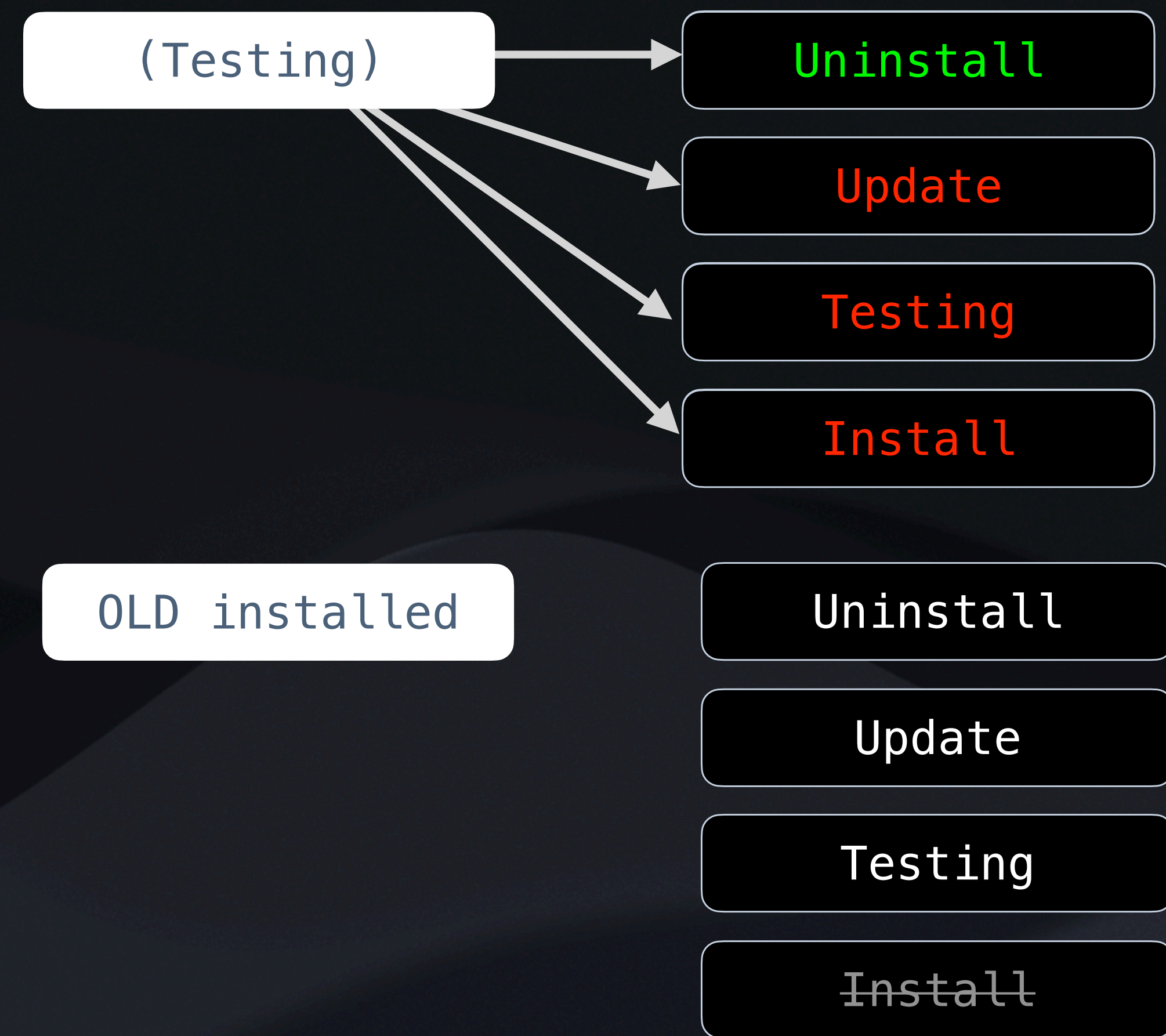
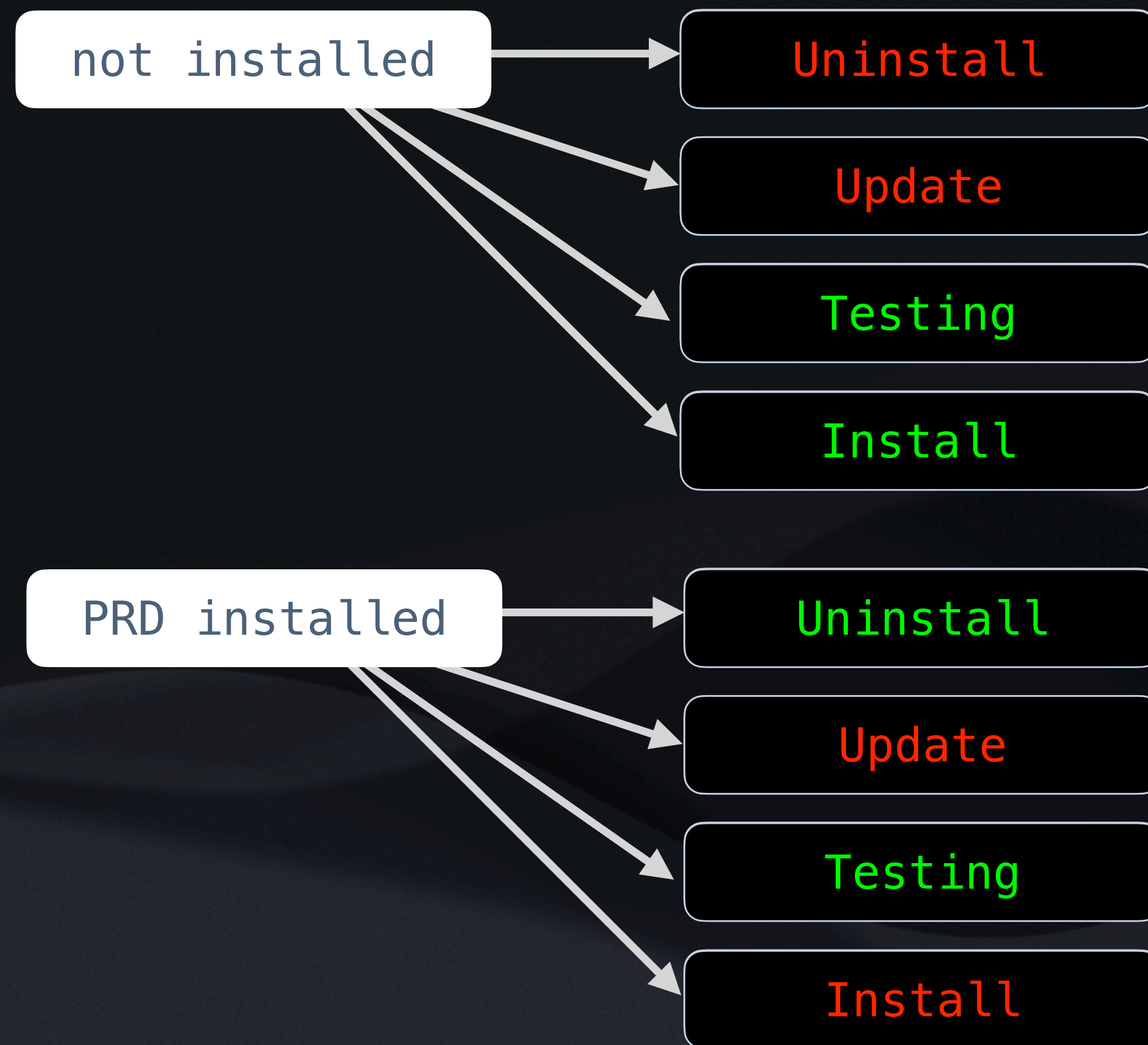
PASS or FAIL?



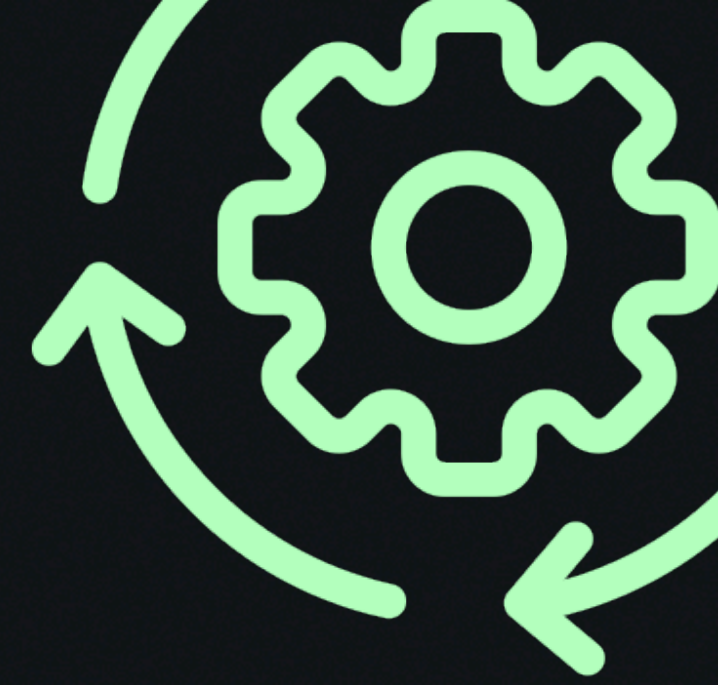
Results of tests - each single scenario



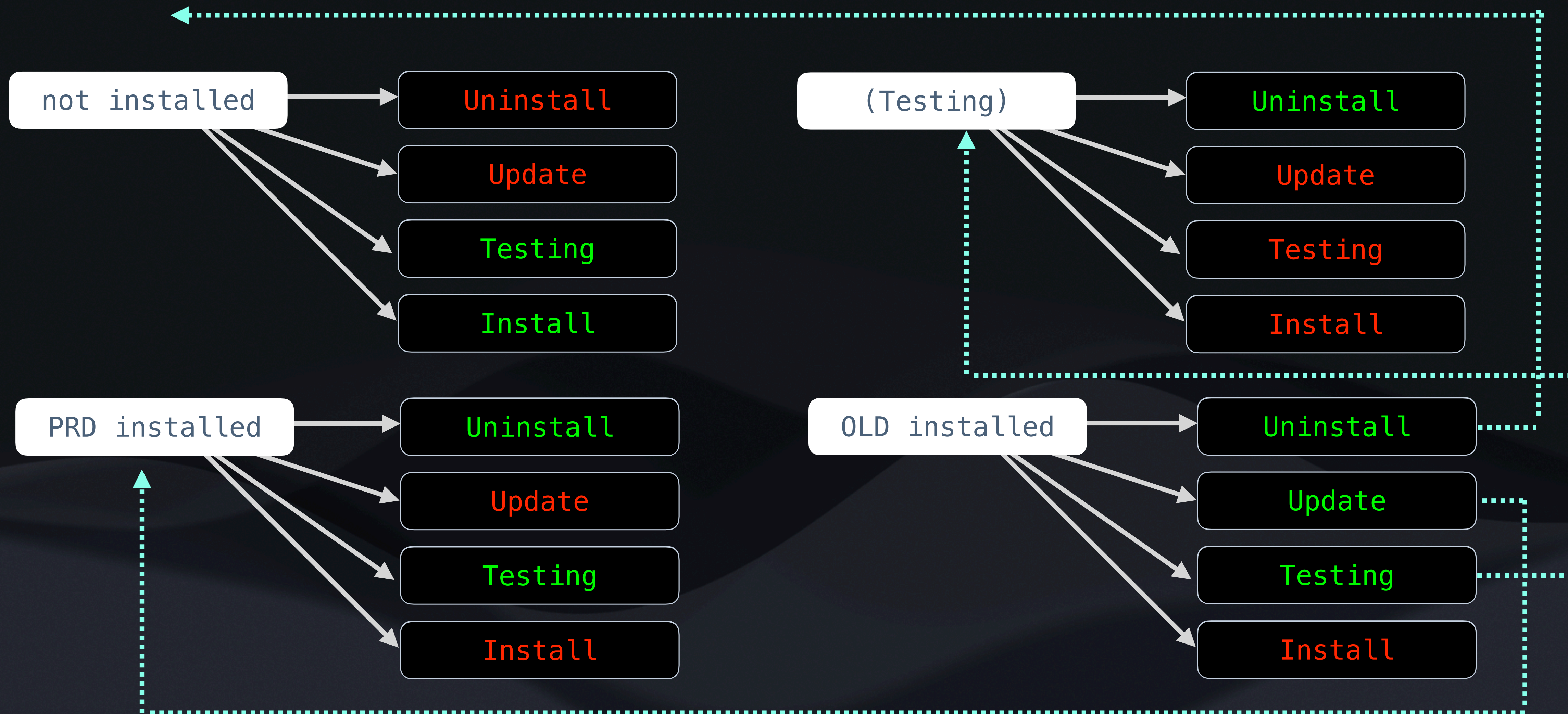
PASS or FAIL?



Results of tests - each single scenario



PASS or FAIL?

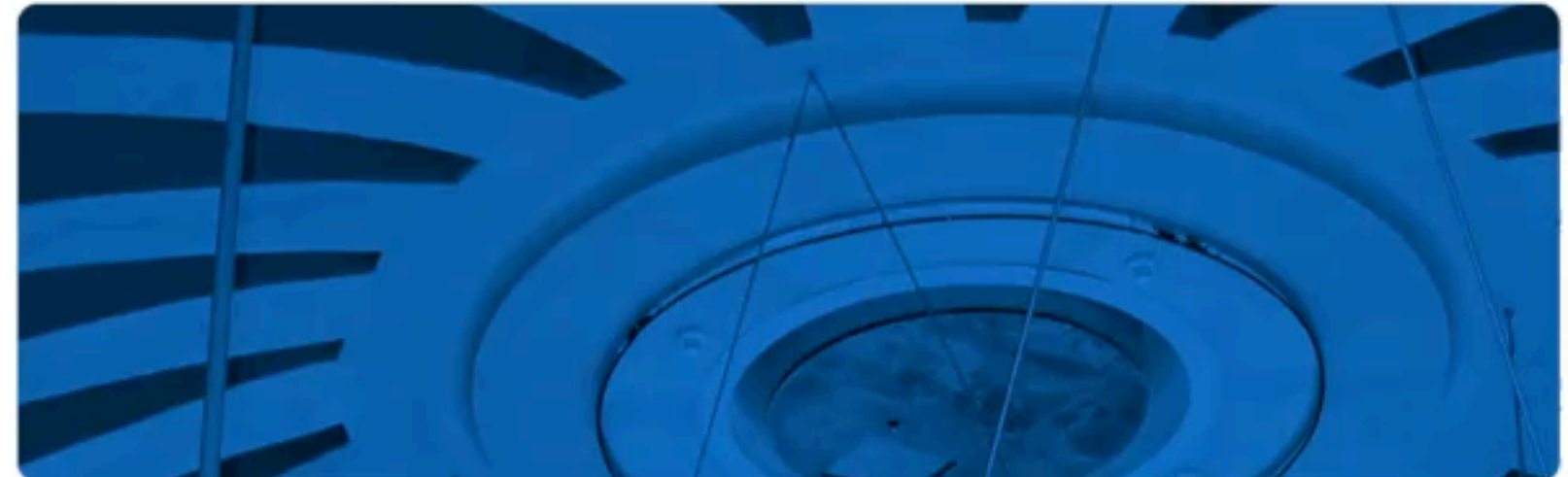


```
test@LifesTooShort Desktop %
```

ETH Self Service
Cloud Sandbox

Search

- Home
- Browse
- Notifications
- History



Browse 15 Results

- All
- Creativity
- Mac App Store Apps
- Tools
- Connect & Remote
- Development & Comp...
- Productivity
- Unin...

app-installation-tester.sh



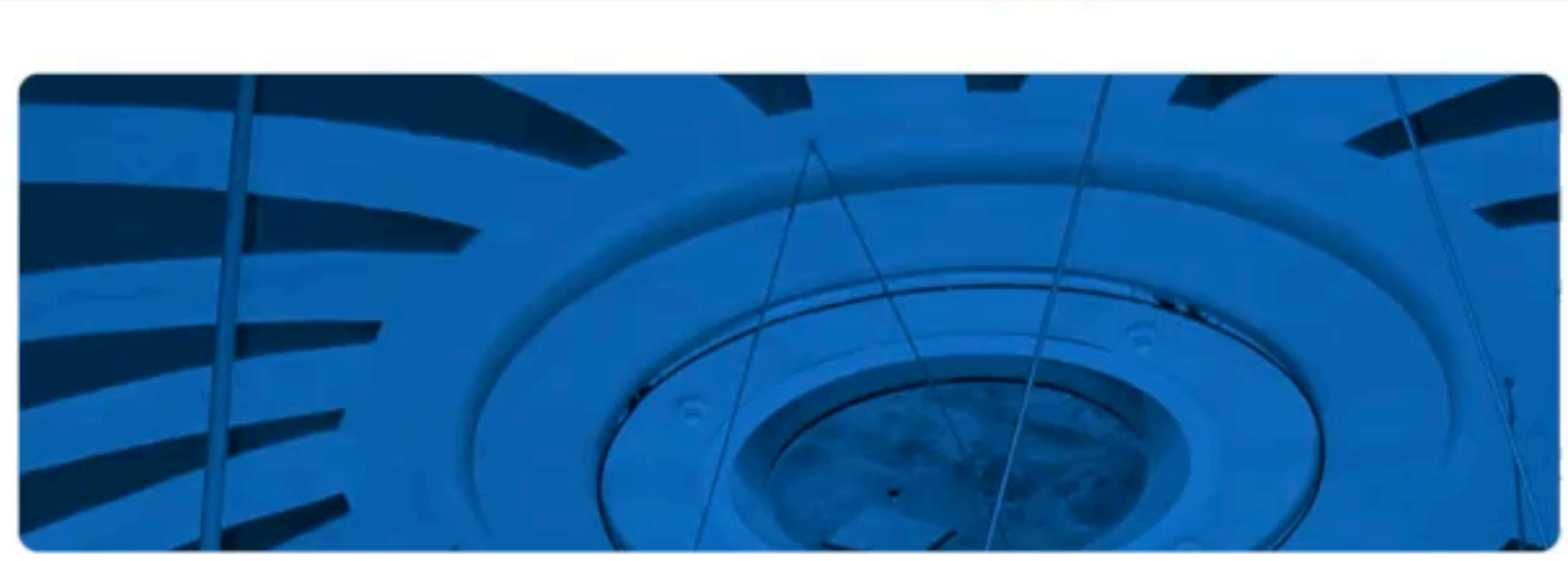
Desktop -- zsh -- 103x41

test@LifesTooShort Desktop %

ETH Self Service
Cloud Sandbox

Search

- Home
- Browse
- Notifications
- History



Browse

15 Results

- All
- Creativity
- Mac App Store Apps
- Tools
- Connect & Remote
- Development & Comp...
- Productivity
- Unin...

SHELL

app-installation-tester.sh



```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✅
[main] Repetition = done

test@LifesTooShort Desktop % █
```

```
test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✓
[main] Repetition = done

test@LifesTooShort Desktop %
```



```
[test@LifesTooShort Desktop % sudo ./app_installation_tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✅
[main] Repetition = done

test@LifesTooShort Desktop %
```

```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✓
[main] Repetition = done

test@LifesTooShort Desktop %
```

```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs for app related policies
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try installing 101 version of GitHub Desktop - should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✅
[main] Repetition = done

test@LifesTooShort Desktop %
```

```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✓
[main] Repetition = done

test@LifesTooShort Desktop % █
```

```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✅
[main] Repetition = done

test@LifesTooShort Desktop % █
```

```
[test@LifesTooShort Desktop % sudo ./app-installation-tester.sh --app "GitHub Desktop" --option t
[initialise_variables] Initialisation of variables
[initialise_variables] Initial Recon running
[initialise_variables] Initial Recon done
[check_token] No token found. Grabbing a new one
[get_token] Token for auto_tester on https://ethztst01.jamfcloud.com/ written to /tmp/jamf_auto_api_
token.txt
[initialise_variables] Initialisation done
[main] Application 1: GitHub Desktop
[main] Option 1: t

--> GitHub Desktop
[main] Starting Testing Process for Application 1 GitHub Desktop of 1
[get_app_policy_list] policy IDs for app related policies
[get_app_policy_list] policy IDs obtained
[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop not installed
[main] Try Installing TST Version of GitHub Desktop -> should result in [PASS]

---> Step 1/1 GitHub Desktop Testing
[test_execute] Start Test 1 of 1 Testing for GitHub Desktop
[execute_content] Executing Policy ID 3594

[get_current_state] Checking current group membership of this computer
[get_current_state] Current state: GitHub Desktop TST installed

EXPECTED_1 : GitHub Desktop:not installed:Testing:TST installed:PASS
RESULT_1   : GitHub Desktop:not installed:Testing:TST installed:PASS
STEP_1     : Check

[test_execute] Test GitHub Desktop [t] completed successfully ✓
[main] Repetition = done

test@LifesTooShort Desktop %
```


```
test@LifesTooShort Desktop %
```

ETH Self Service
Cloud Sandbox

Search: github

- Home
- Browse
- Notifications
- History

Search results for "github"



Uninstall GitHub Desktop

Uninstall



```
test@LifesTooShort Desktop %
```

ETH Self Service
Cloud Sandbox

Search results for "github"

github

- Home
- Browse
- Notifications
- History

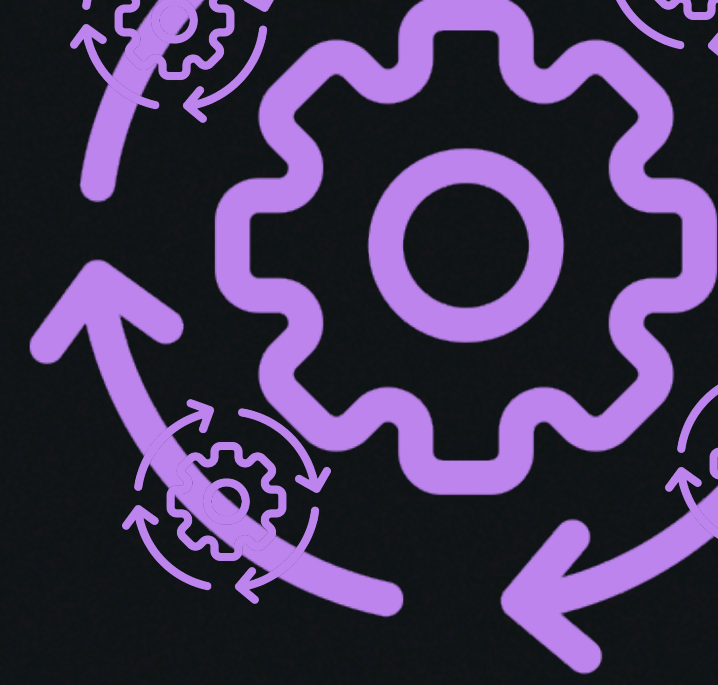
Uninstall GitHub Desktop

Uninstall

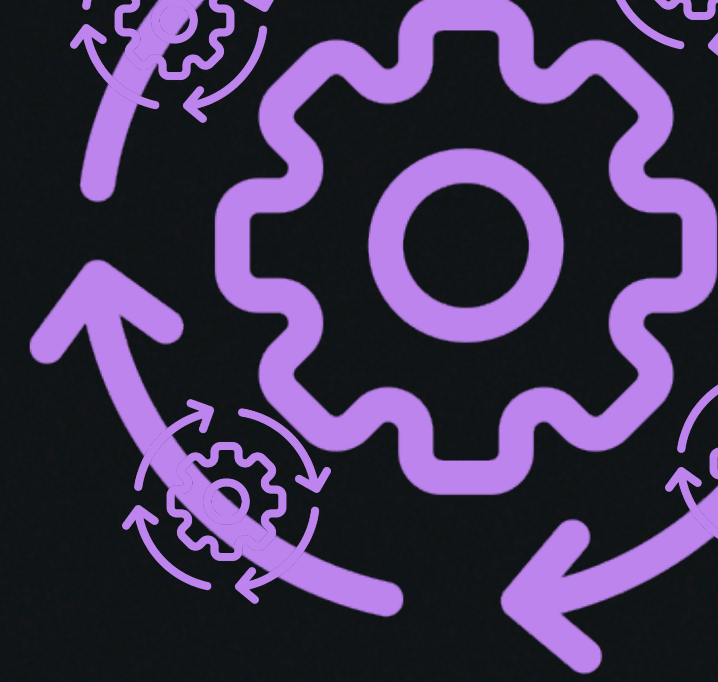
app-installation-tester.sh



Order of tests - testing workflow



Order of tests - testing workflow



not installed

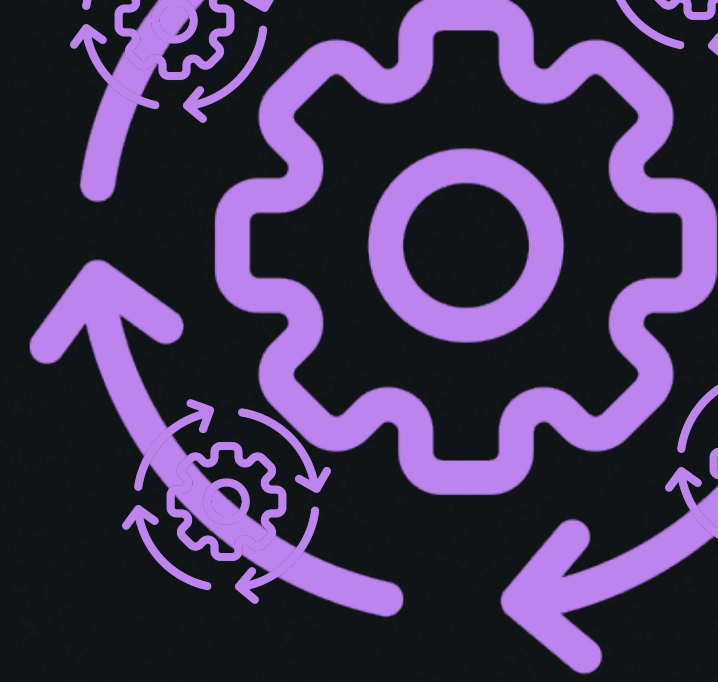
Uninstall

Update

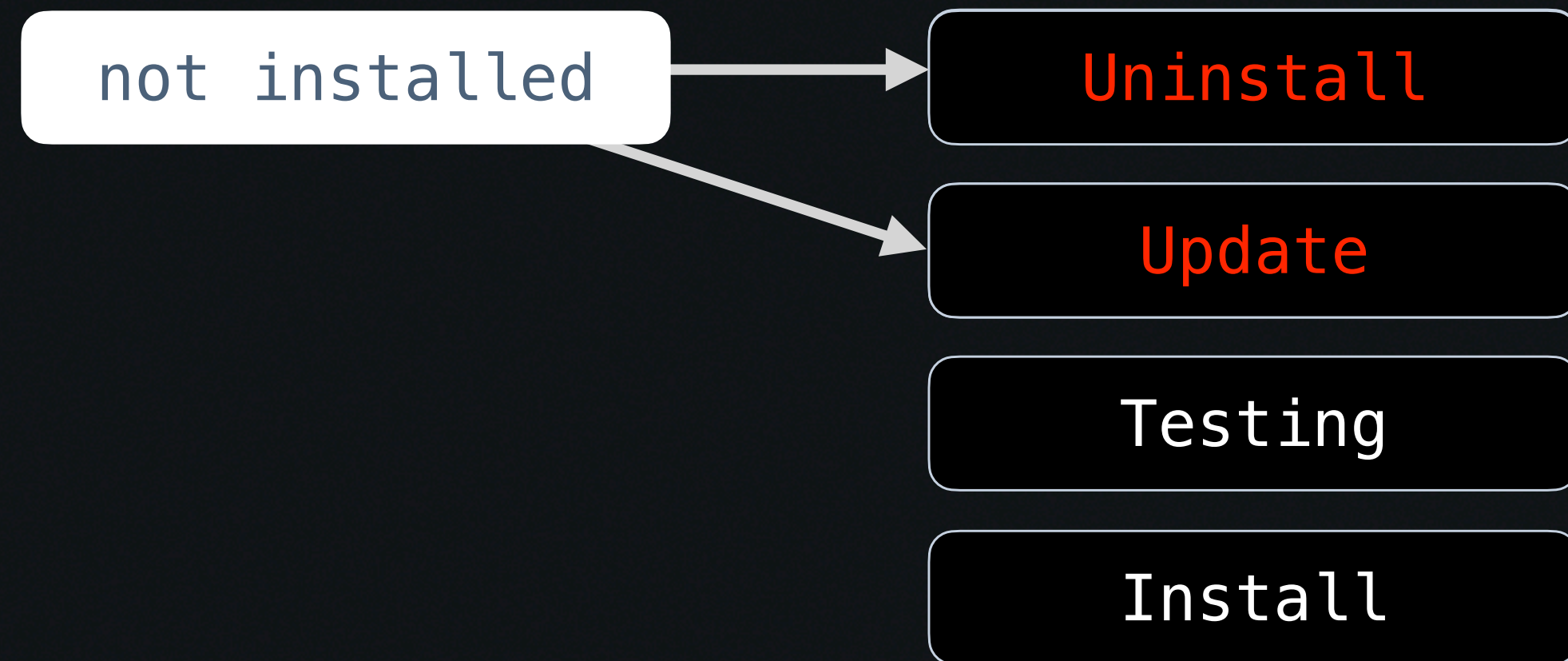
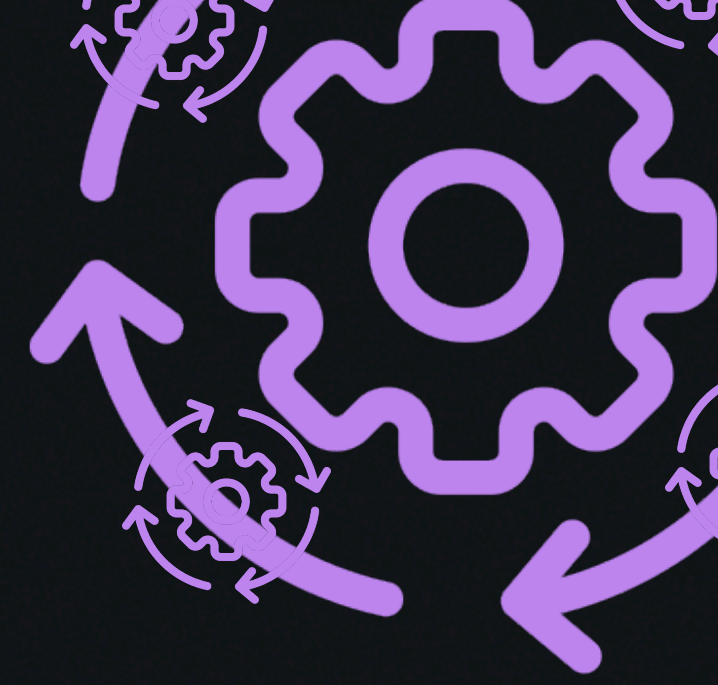
Testing

Install

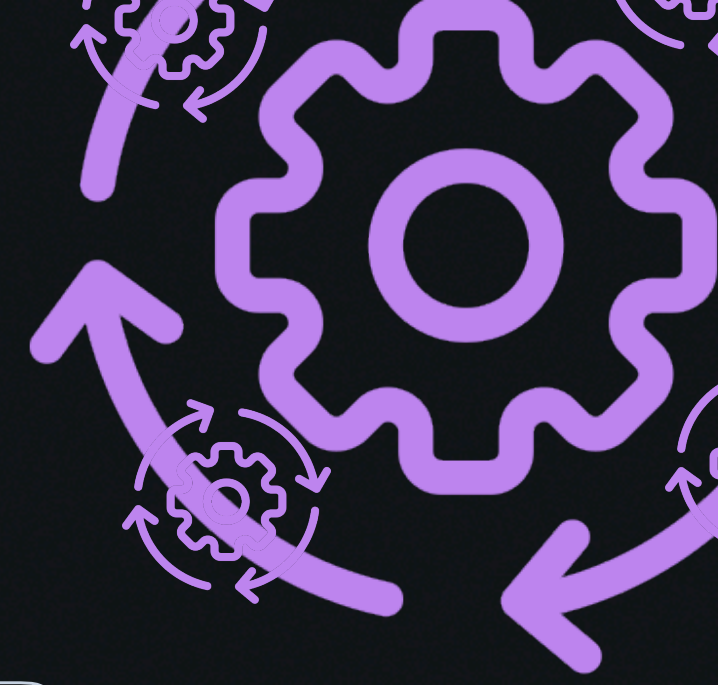
Order of tests - testing workflow



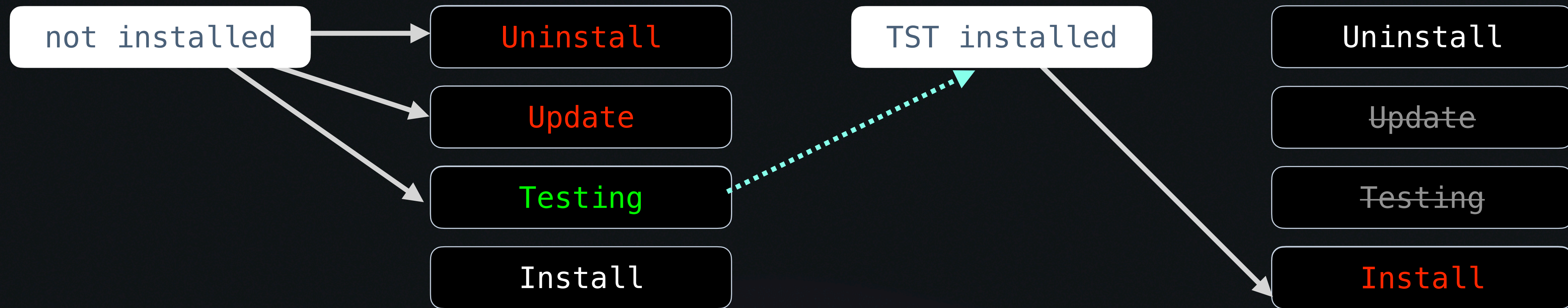
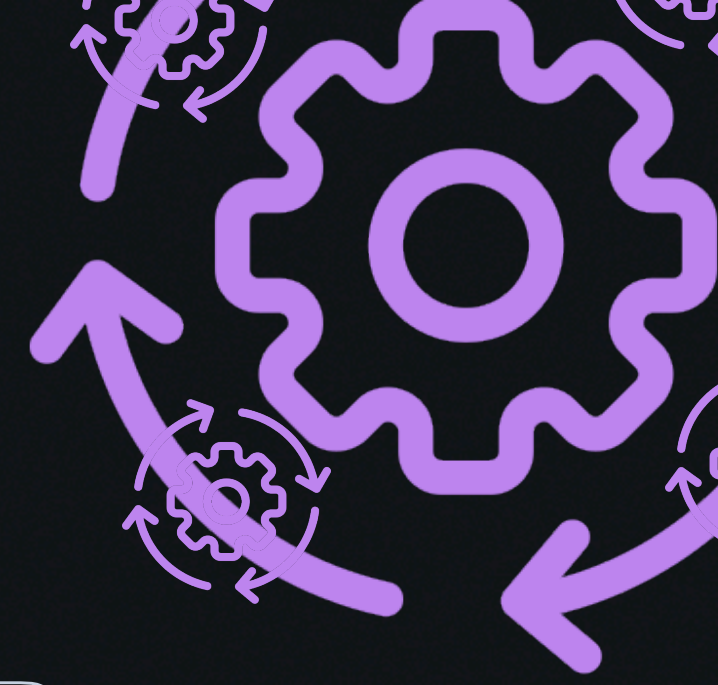
Order of tests - testing workflow



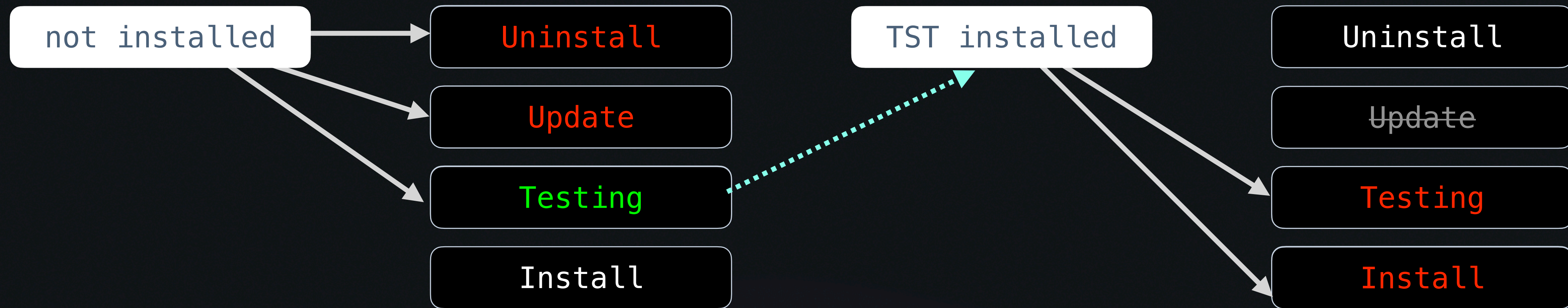
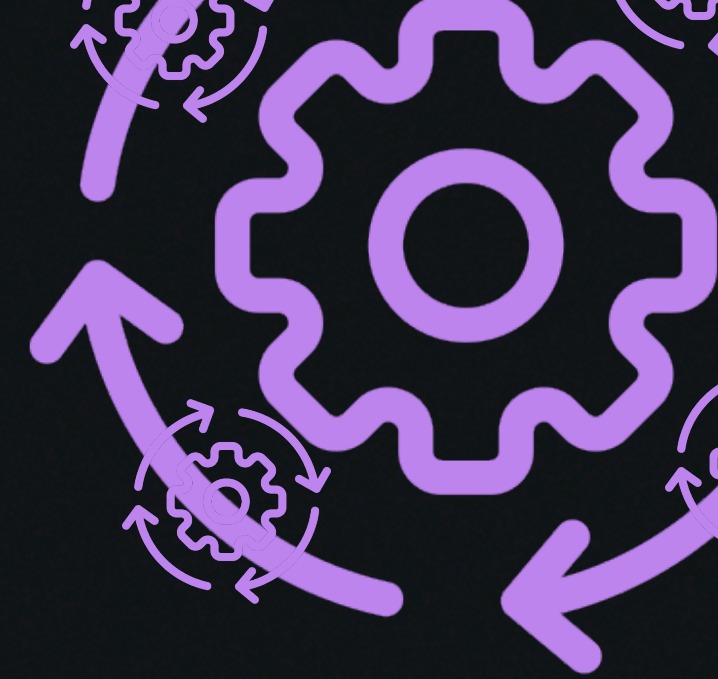
Order of tests - testing workflow



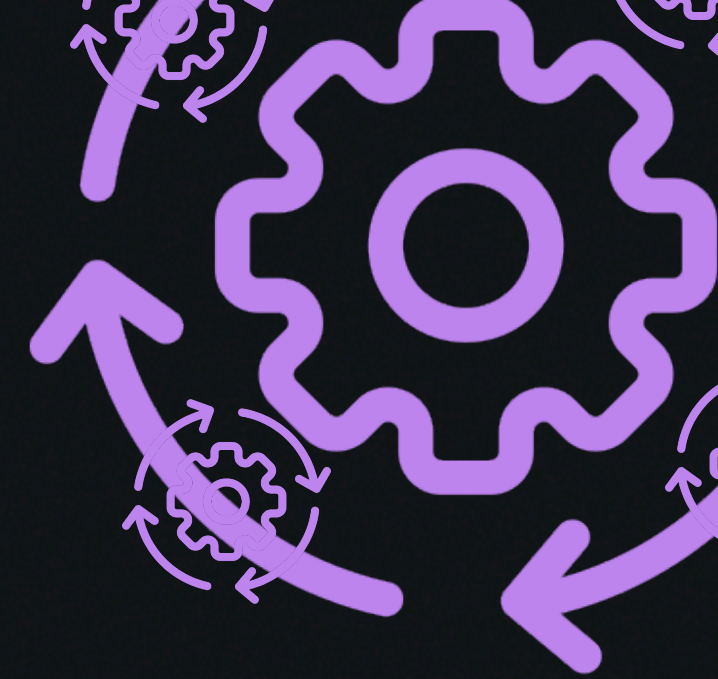
Order of tests - testing workflow



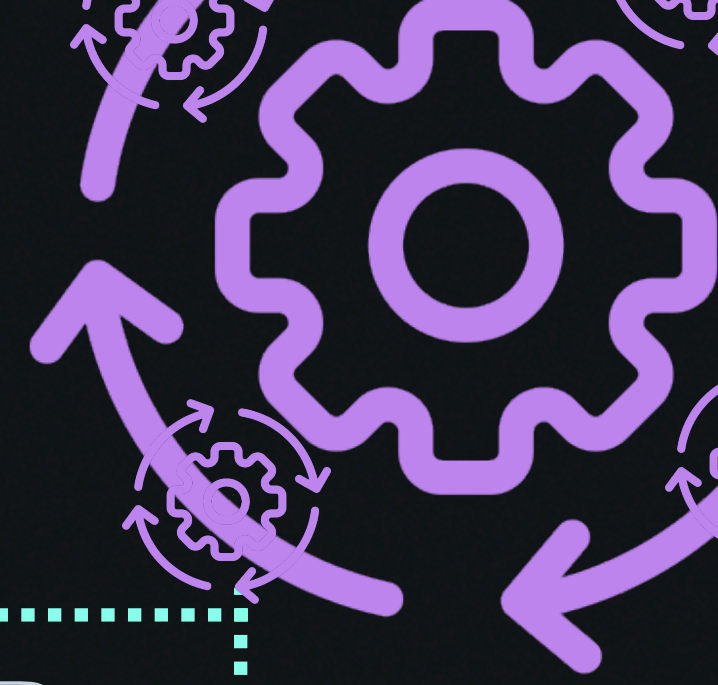
Order of tests - testing workflow



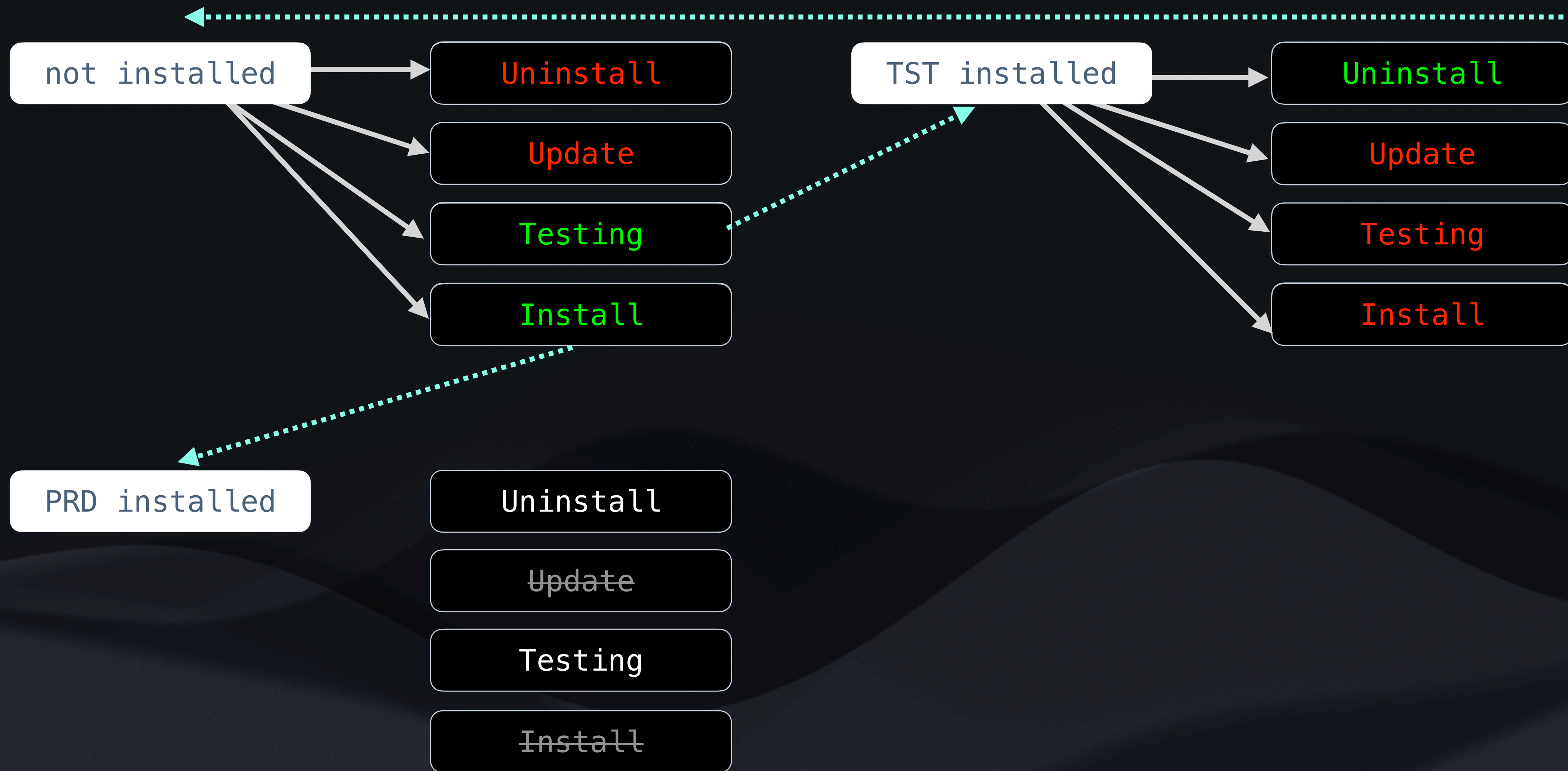
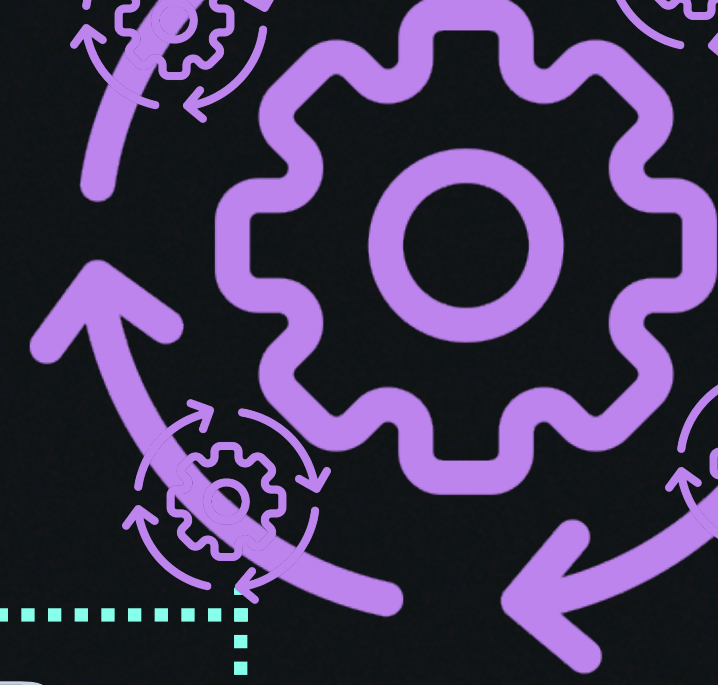
Order of tests - testing workflow



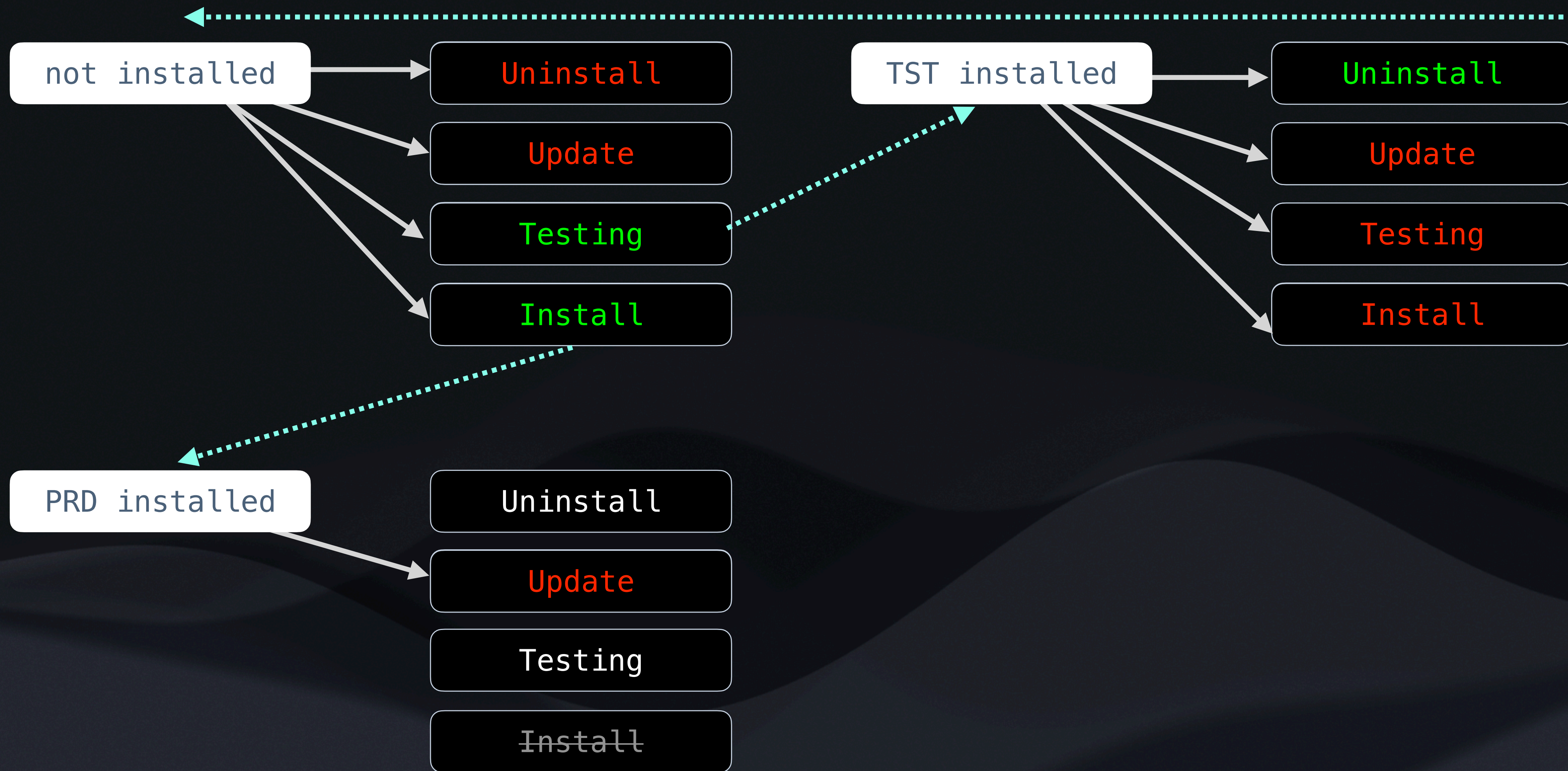
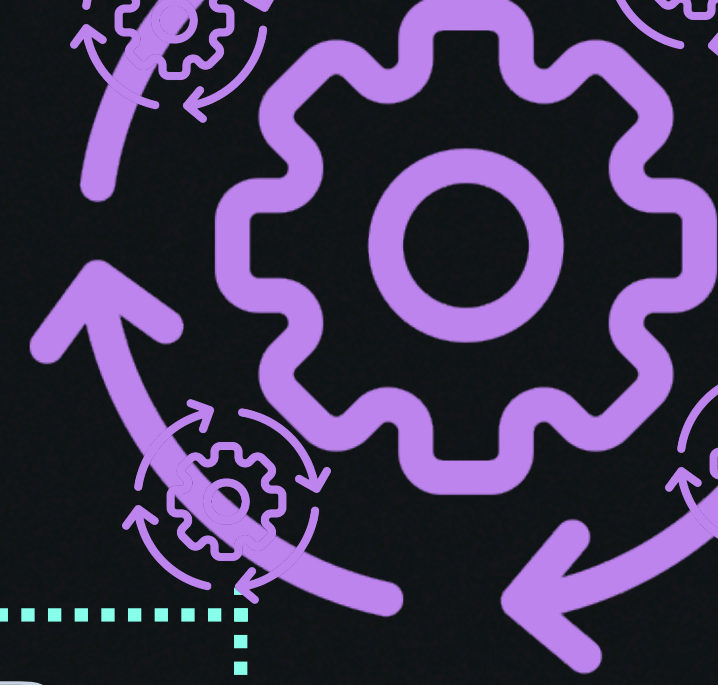
Order of tests - testing workflow



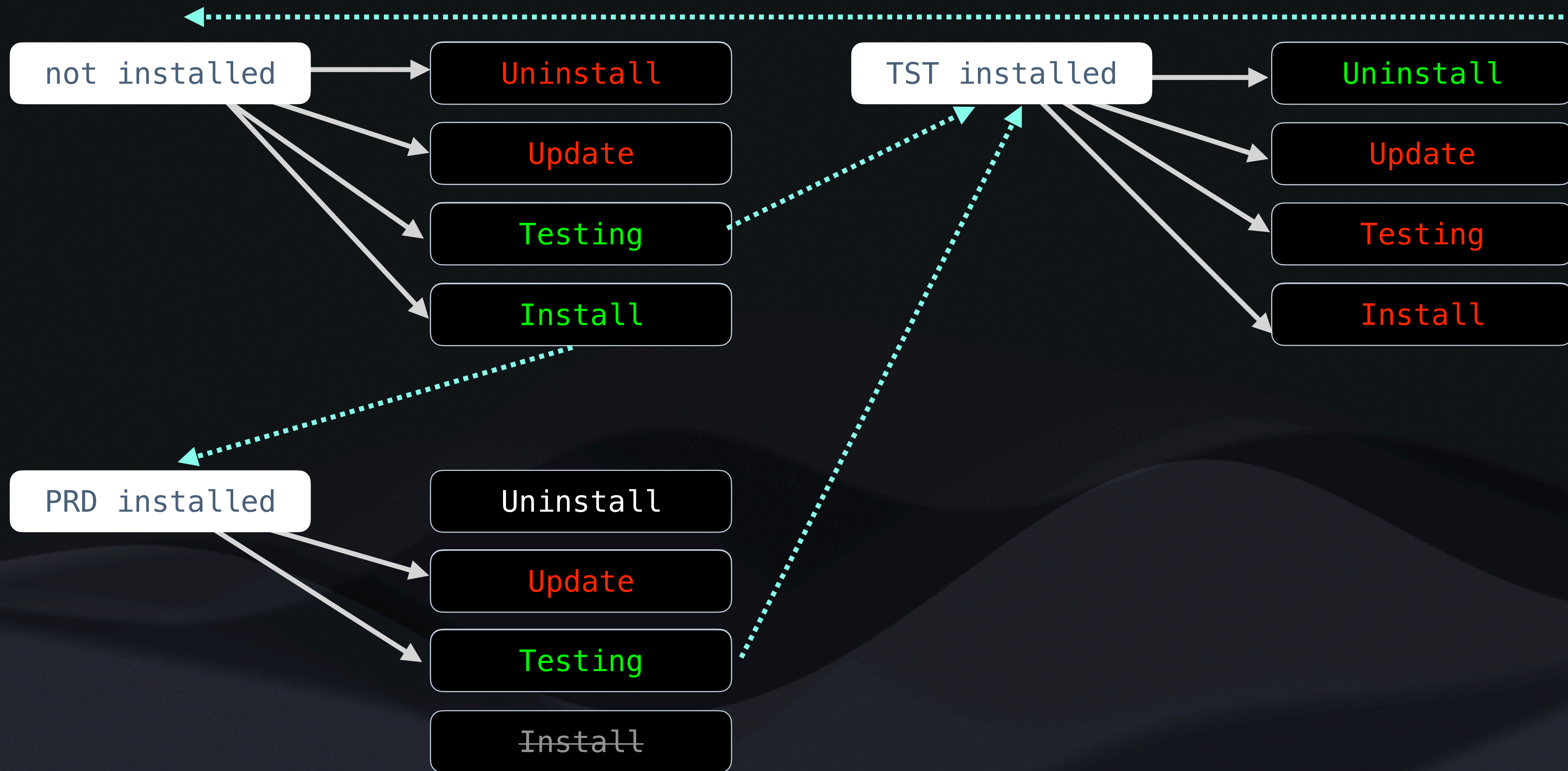
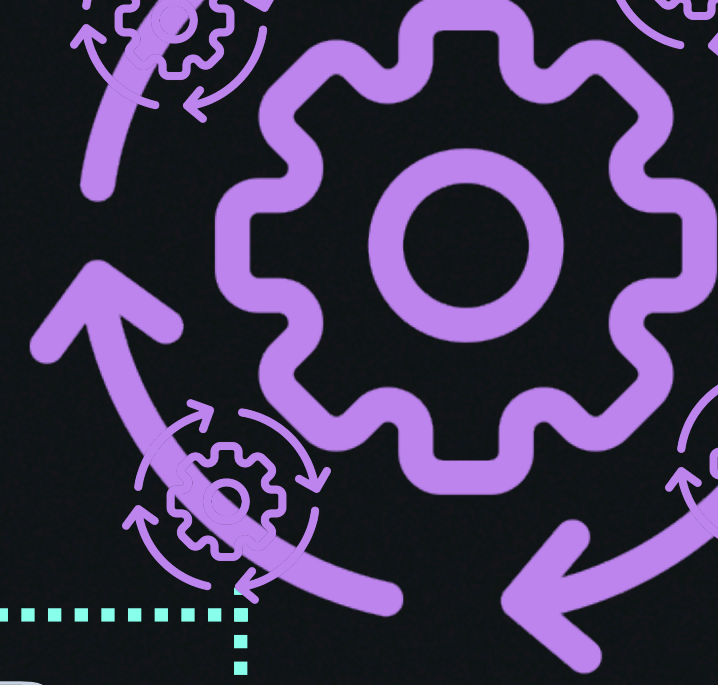
Order of tests - testing workflow



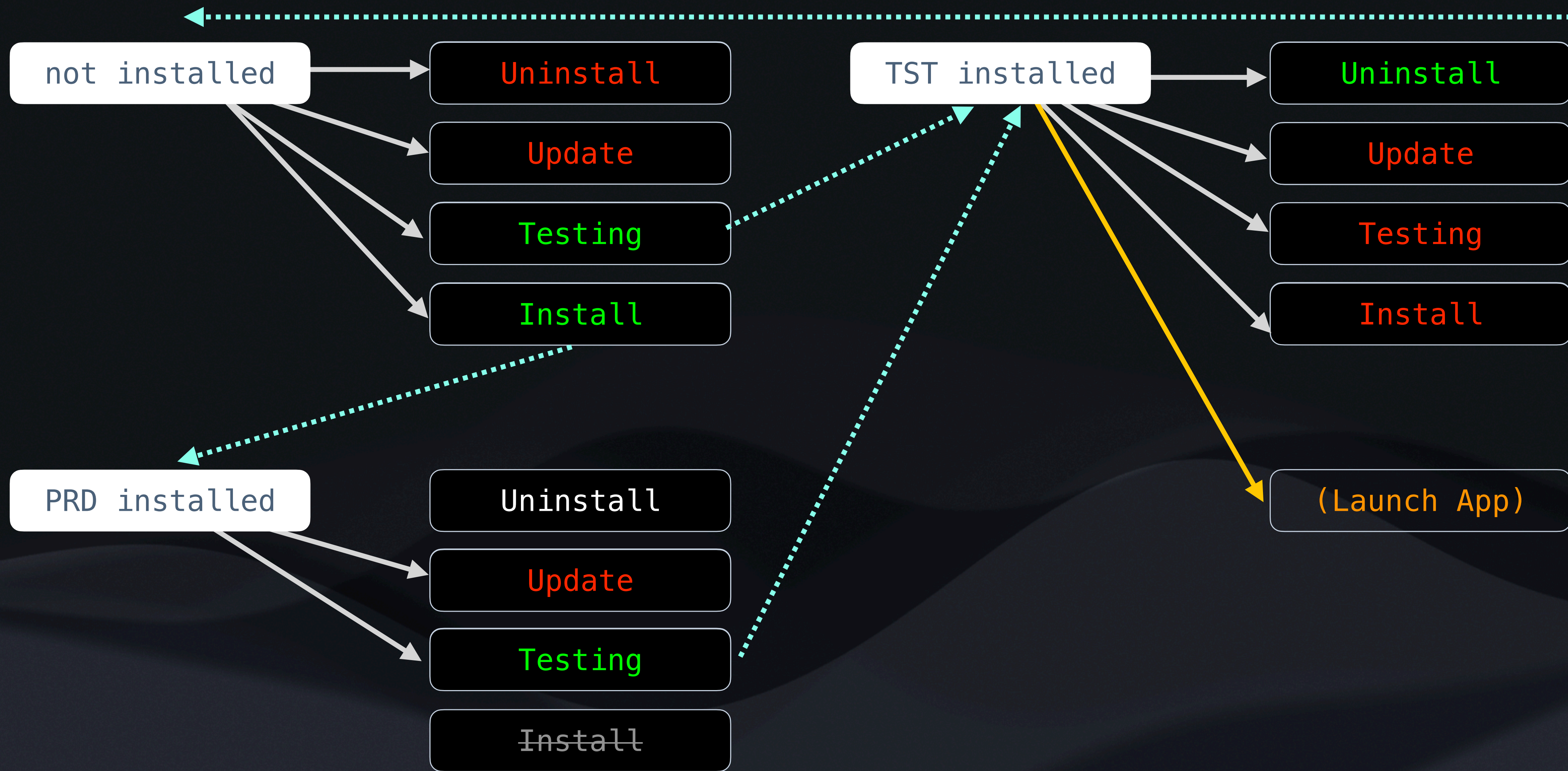
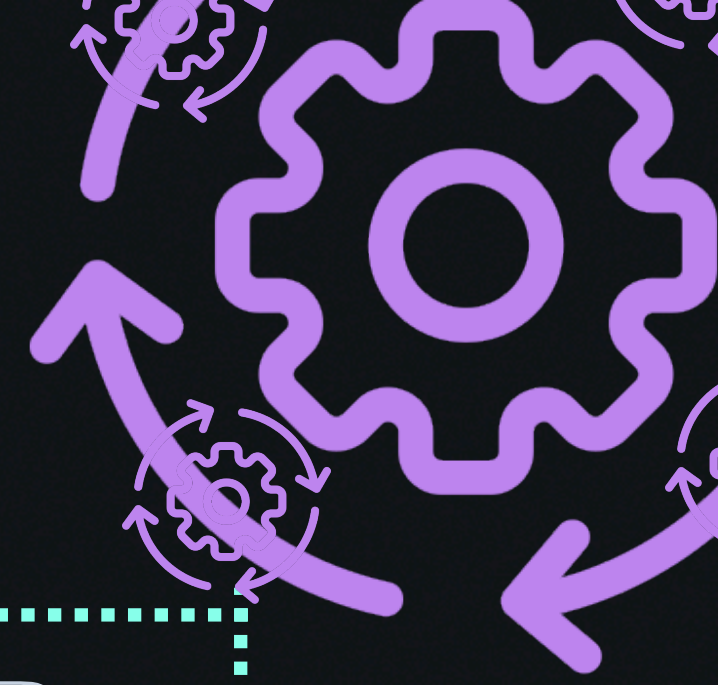
Order of tests - testing workflow



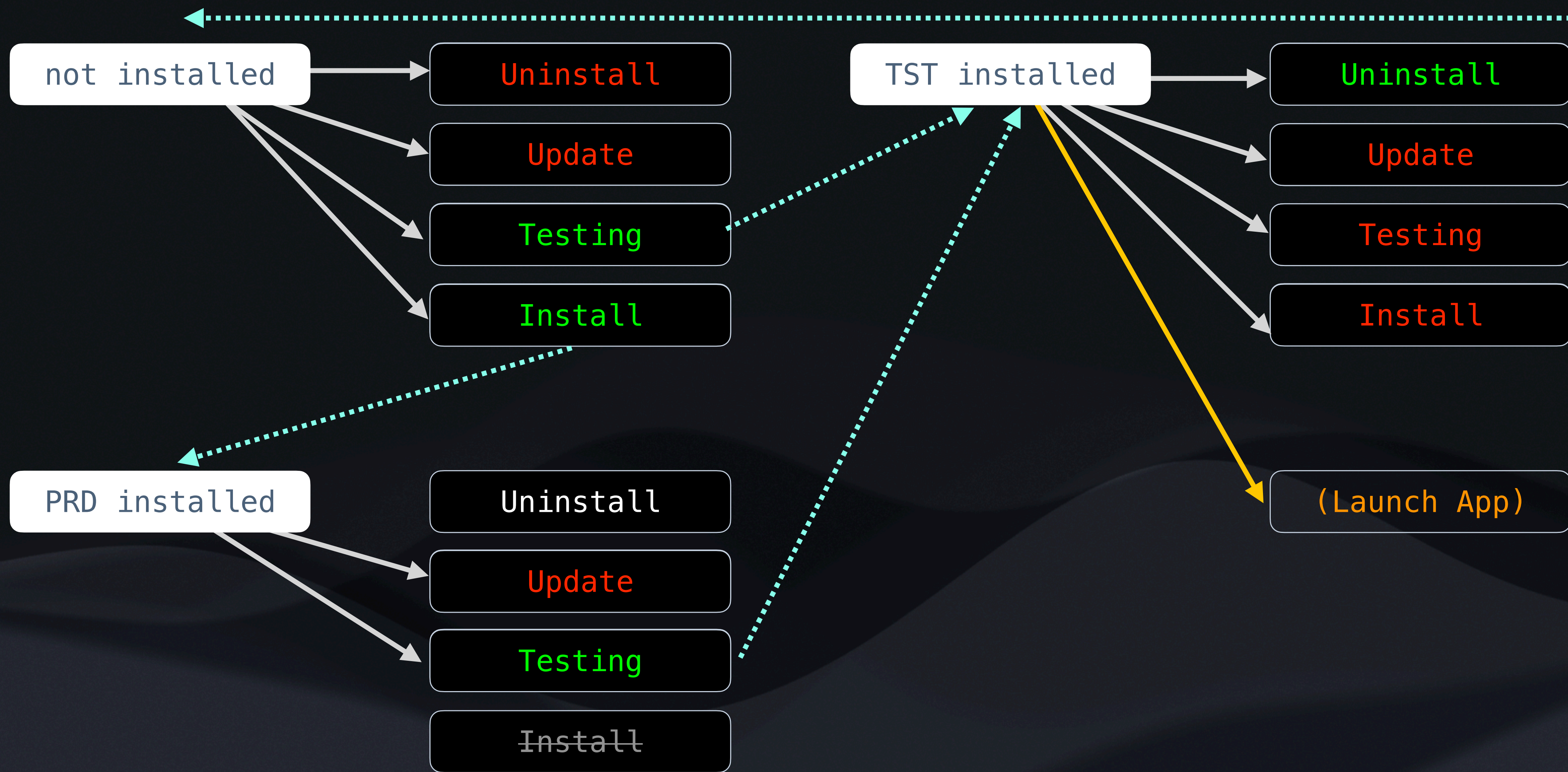
Order of tests - testing workflow



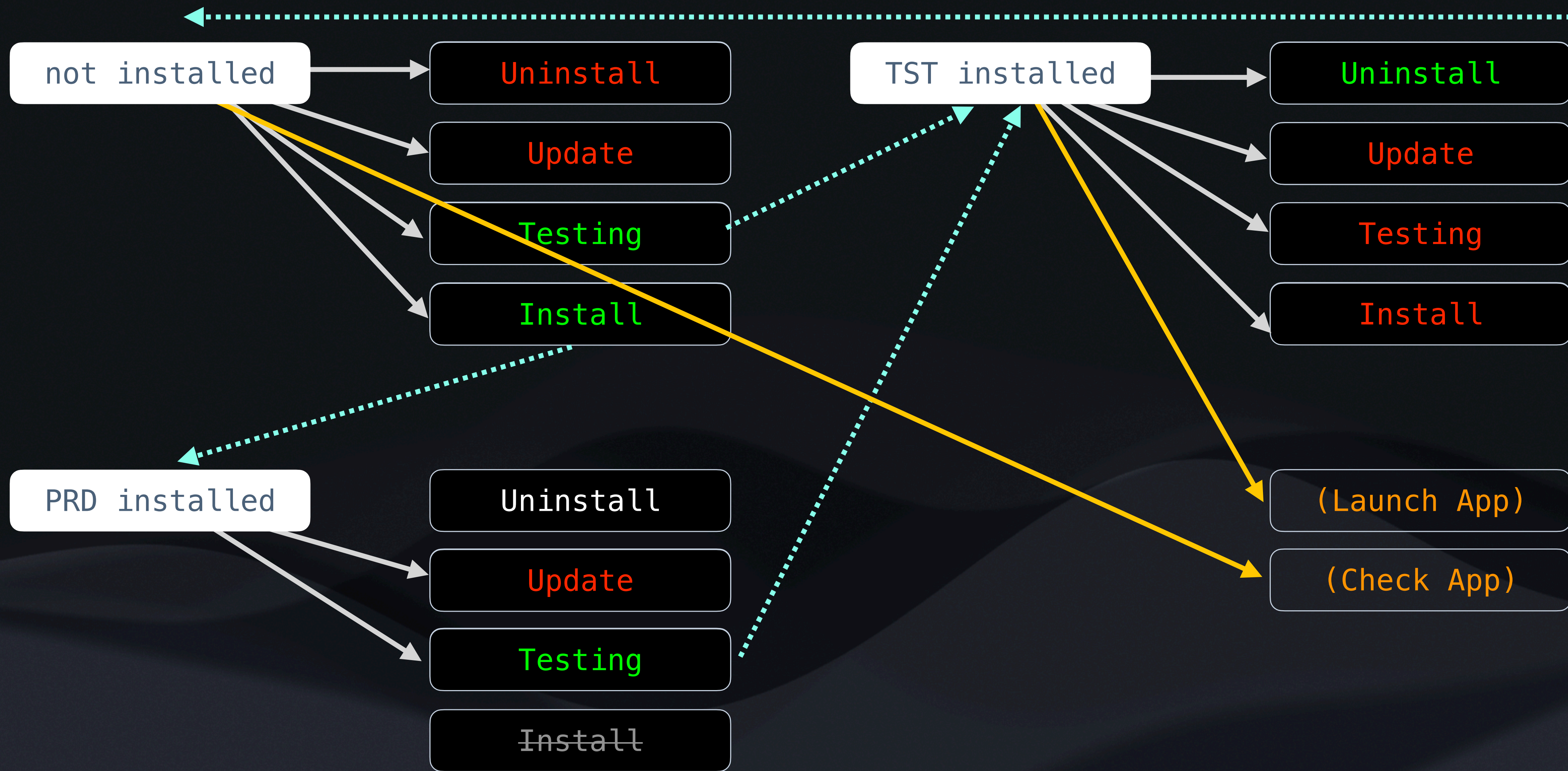
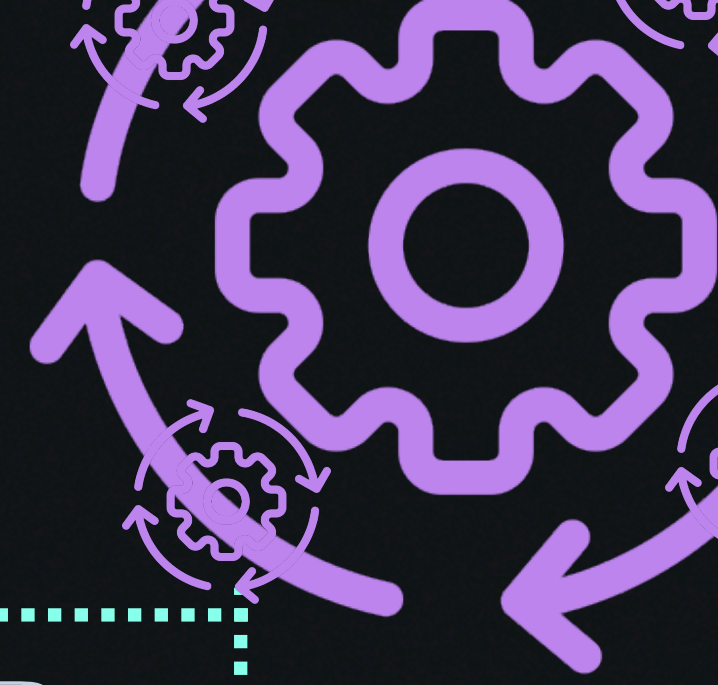
Order of tests - testing workflow



Order of tests - testing workflow



Order of tests - testing workflow



test@LifesTooShort Desktop %

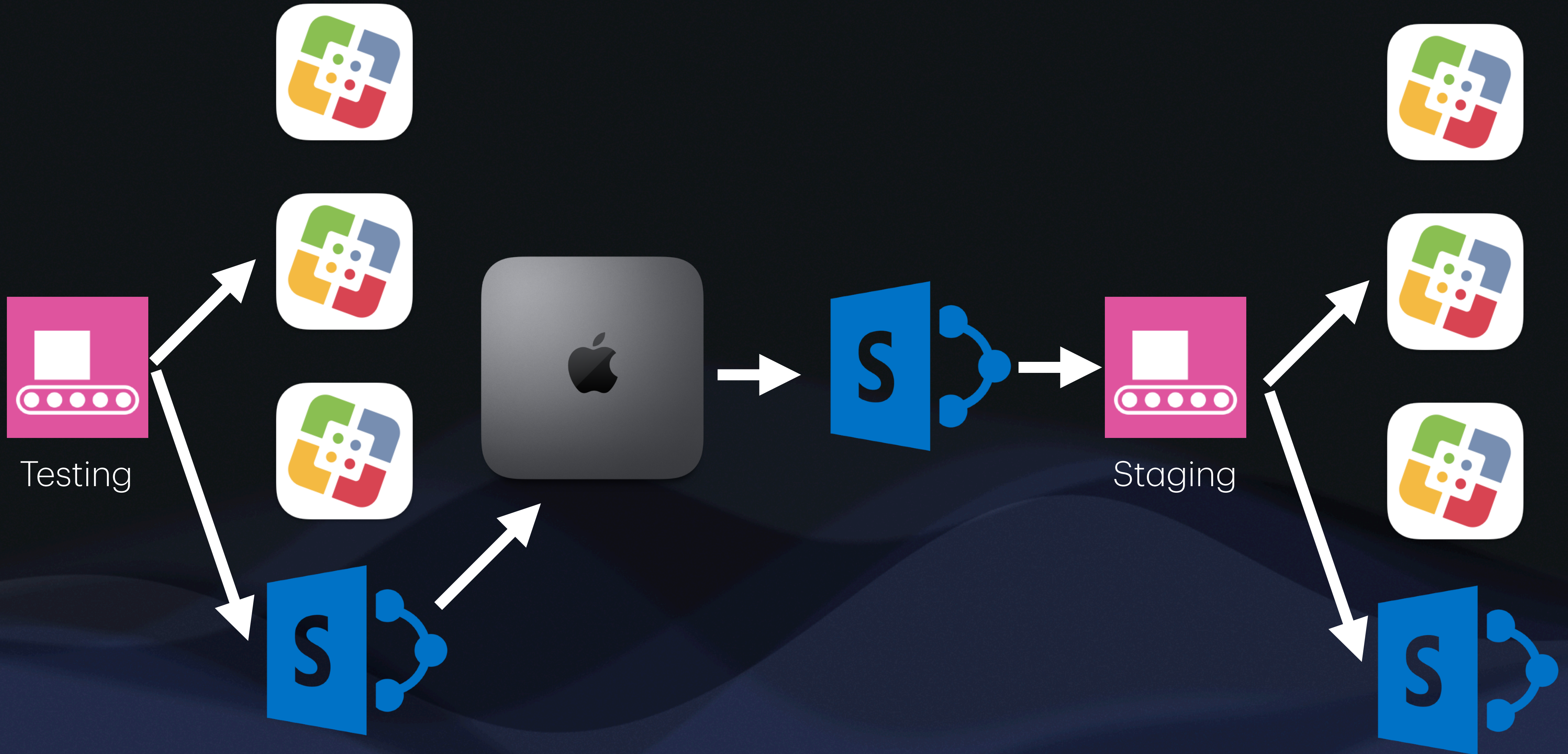
app-installation-tester.sh

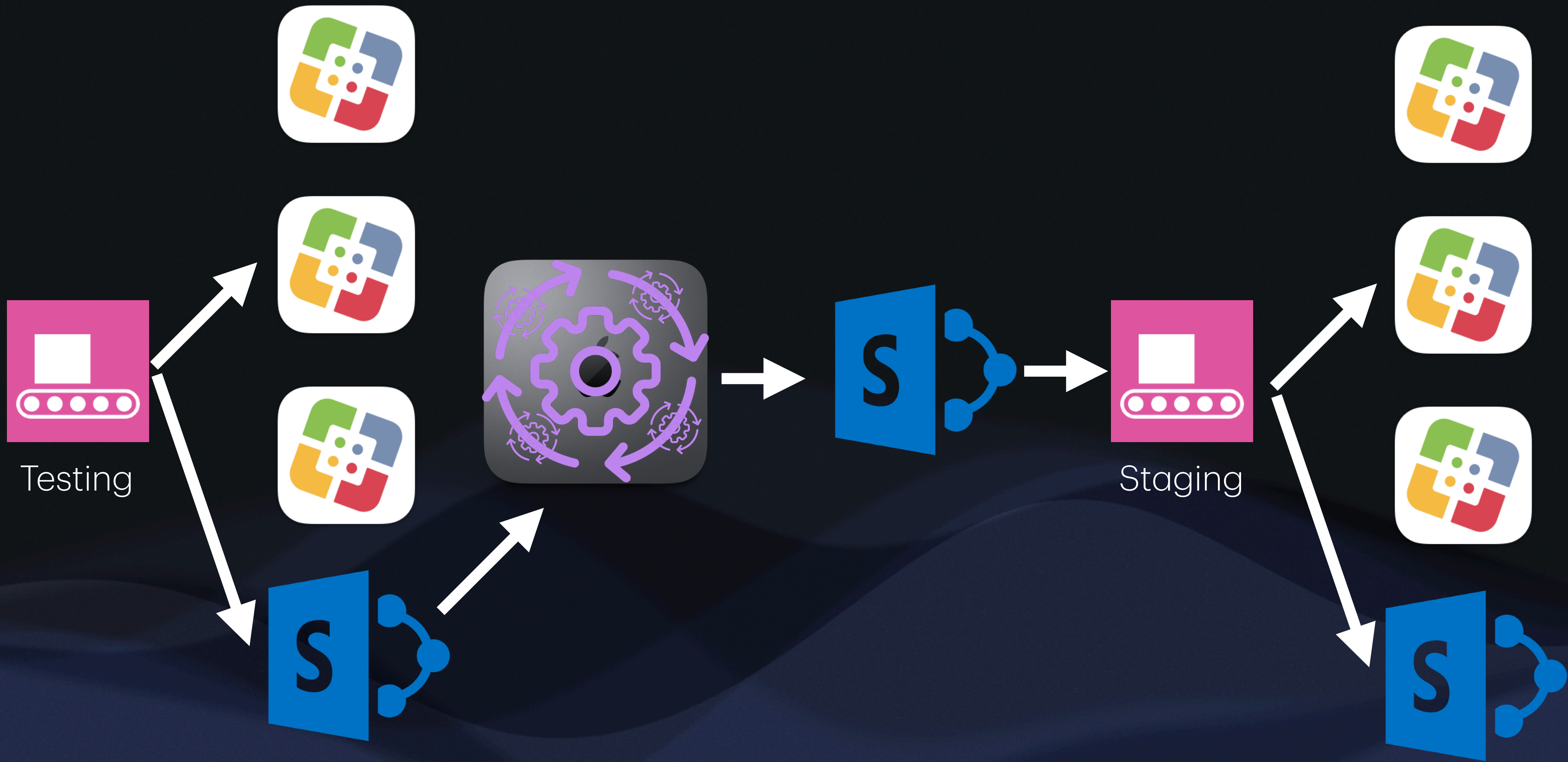


test@LifesTooShort Desktop %

app-installation-tester.sh







Positive Effects

Positive Effects

- 90 of 200 apps now auto-staging

Positive Effects

- 90 of 200 apps now auto-staging
- Apps released to production faster

Positive Effects

- 90 of 200 apps now auto-staging
- Apps released to production faster
- Close to zero complaints

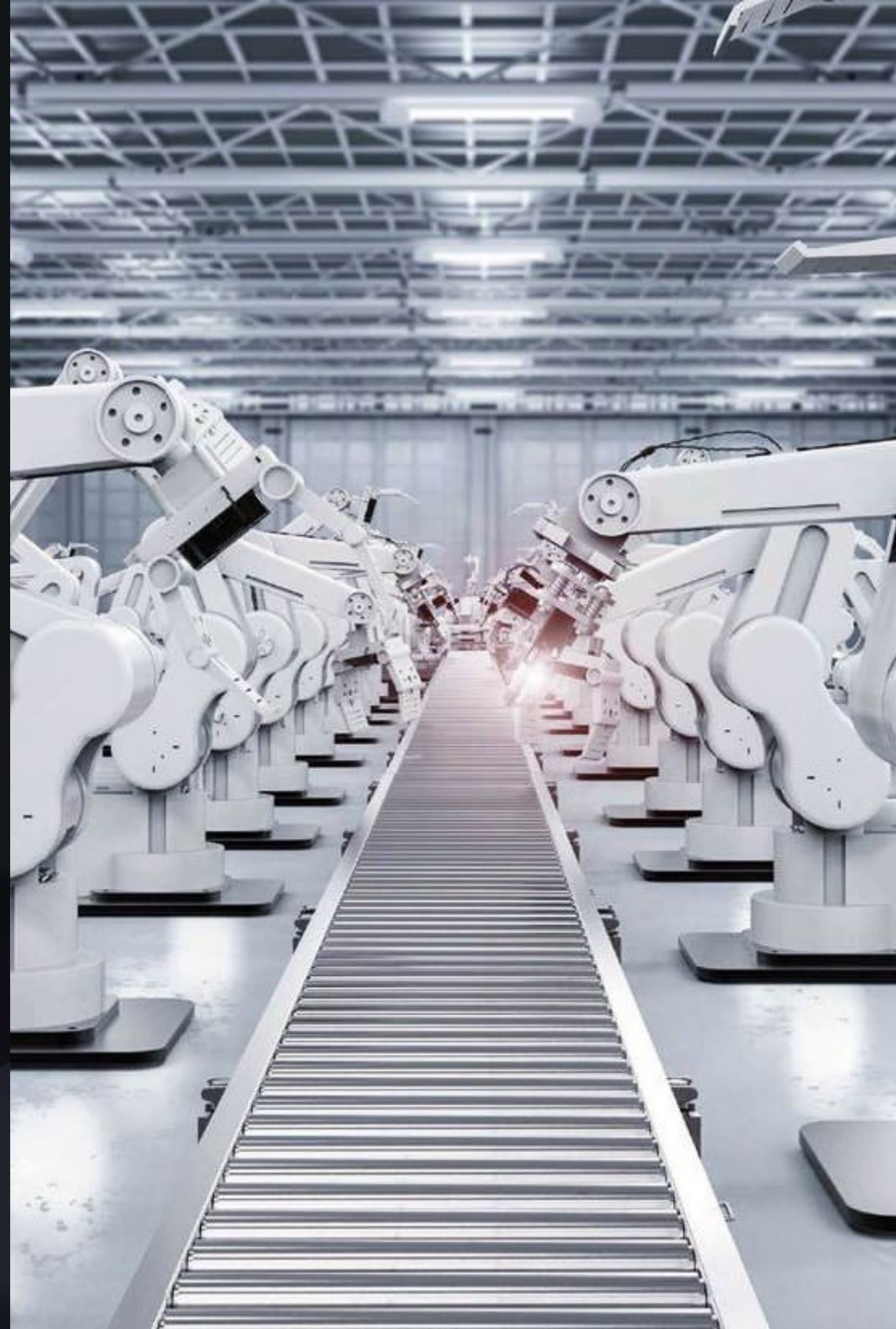
Positive Effects

- 90 of 200 apps now auto-staging
- Apps released to production faster
- Close to zero complaints
- Higher reliability means confidence to enforce automatic updates for all users for core titles

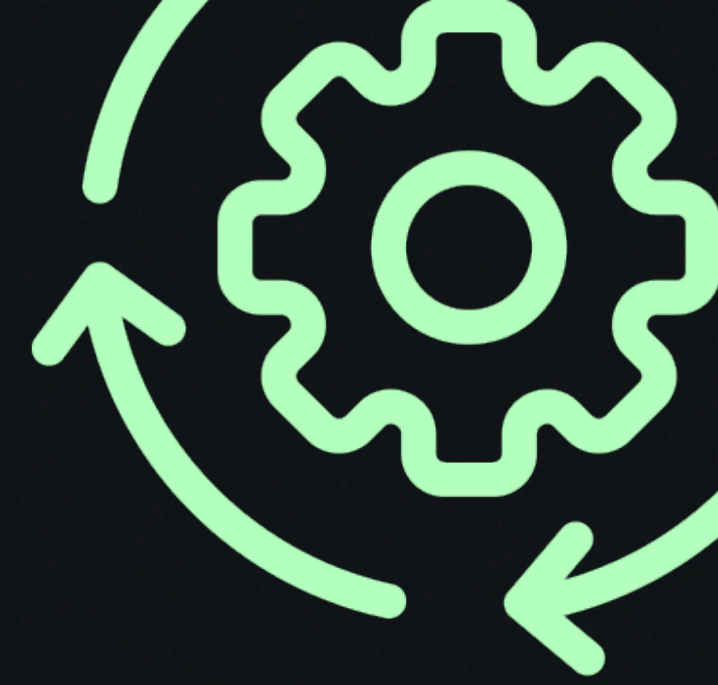
Positive Effects

- 90 of 200 apps now auto-staging
- Apps released to production faster
- Close to zero complaints
- Higher reliability means confidence to enforce automatic updates for all users for core titles
- Simplifying Jamf content

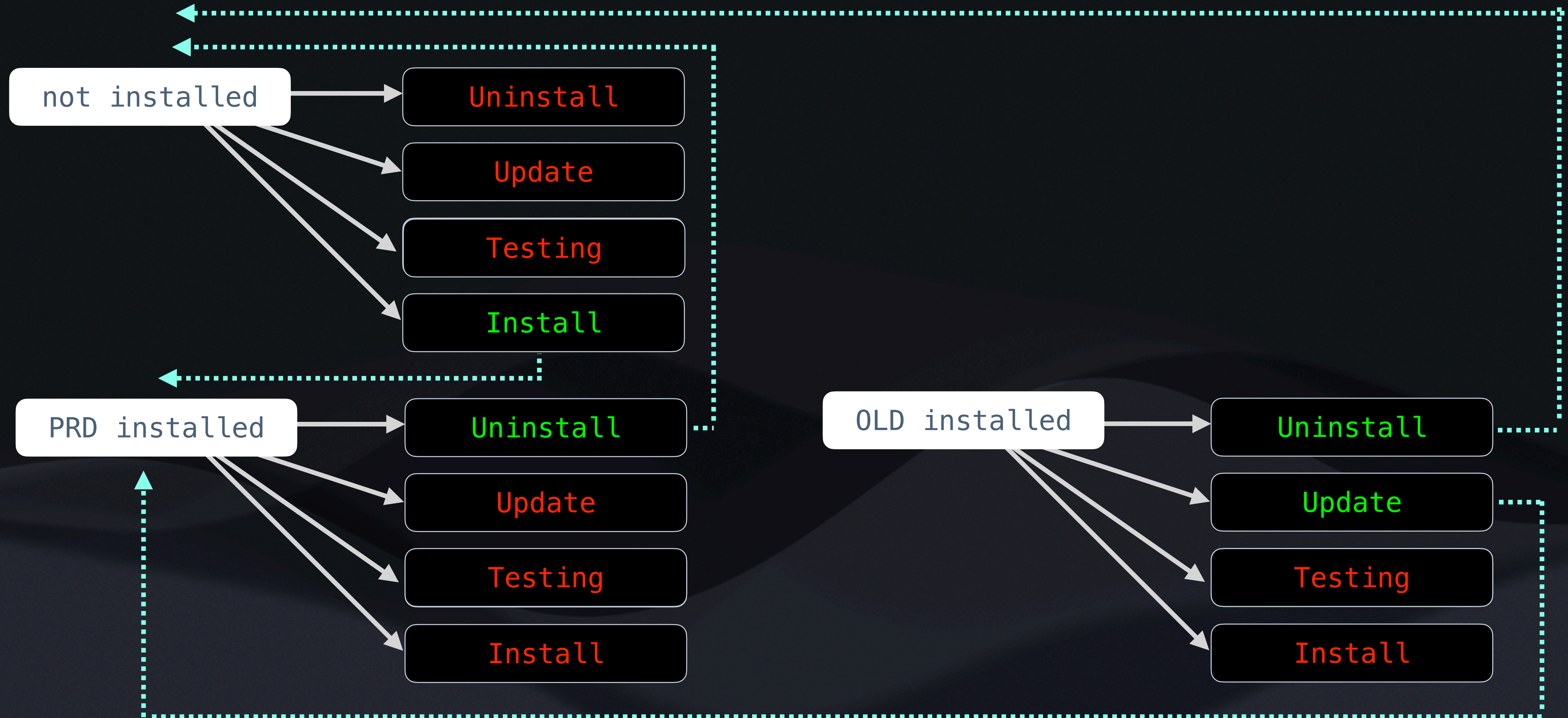
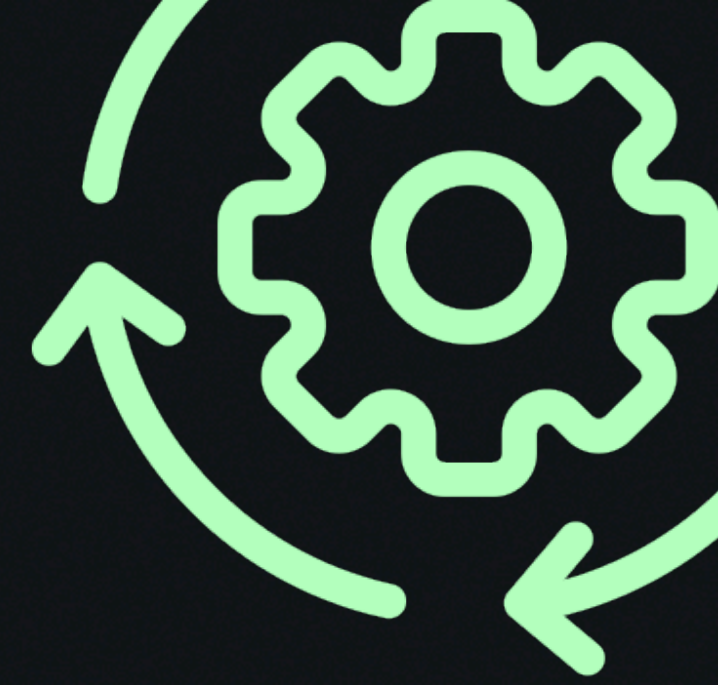
What Next?



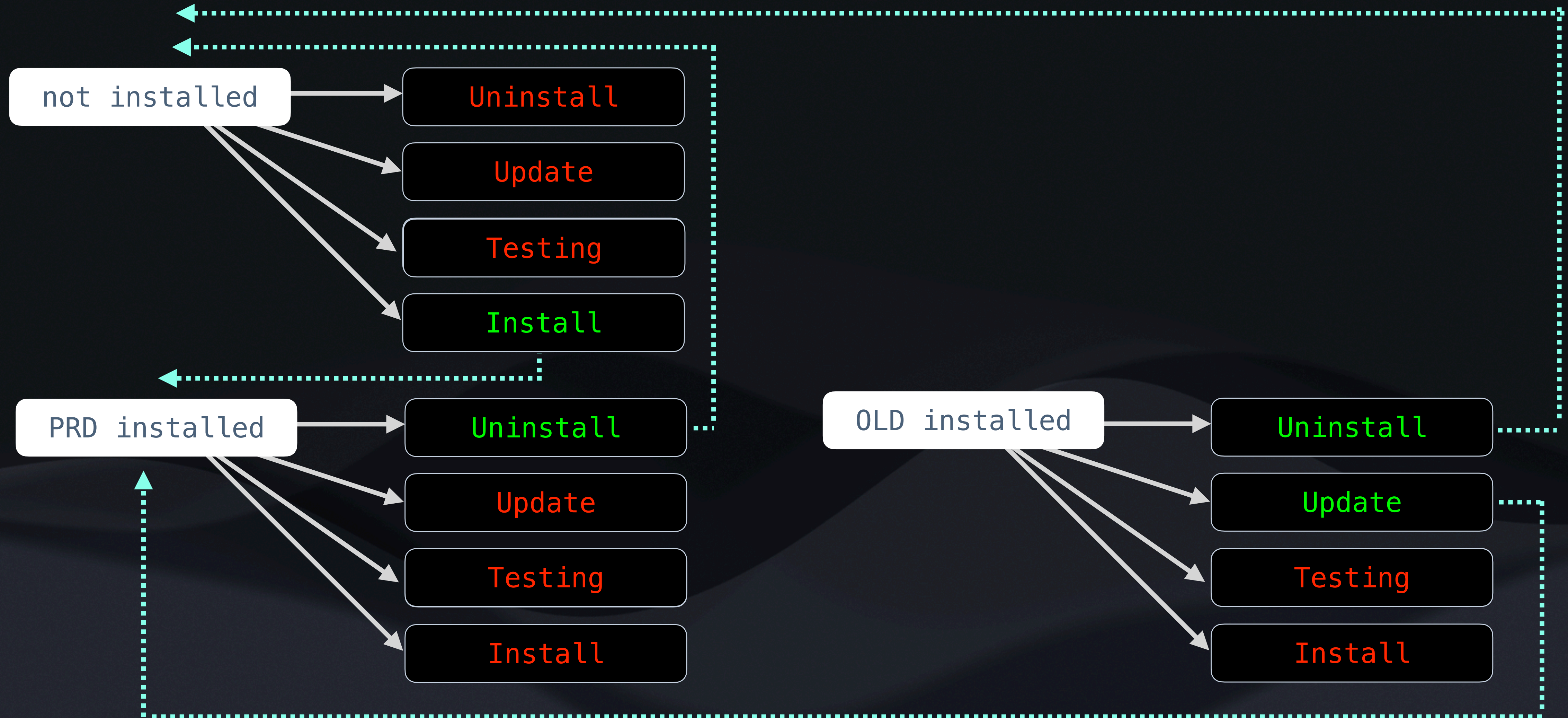
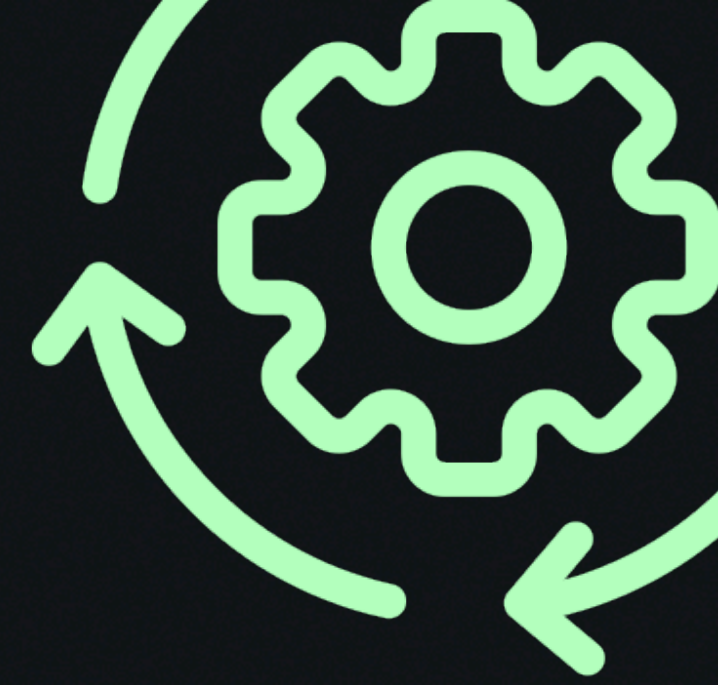
Results of tests - future staging scenario



Results of tests - future staging scenario



Results of tests - future staging scenario



Functionality testing?

Functionality testing?

- Agents - check process
- Binaries/Frameworks - run simple command
- PPC profiles - check TCC database
- System Extensions - check with `systemextensionsctl`
- Beyond that - bespoke scripts for individual apps

Thank You!
Merci vielmal!



Thank You!
Merci vielmal!

