

Creating DevOps RunBook Automation with an AI bot

Doug Sillars
 **@dougsillars**
Head of Developer Relations



**RunBook =
Documentation**

Documentation



Documentation



Documentation



Code samples

GET /repos/{owner}/{repo}/traffic/views

cURL

JavaScript

GitHub CLI



```
curl \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer <YOUR-TOKEN>" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  https://api.github.com/repos/OWNER/REPO/traff:
```

Documentation



HTTP/2 403

"message": "Resource not accessible
by personal access token"

Documentation

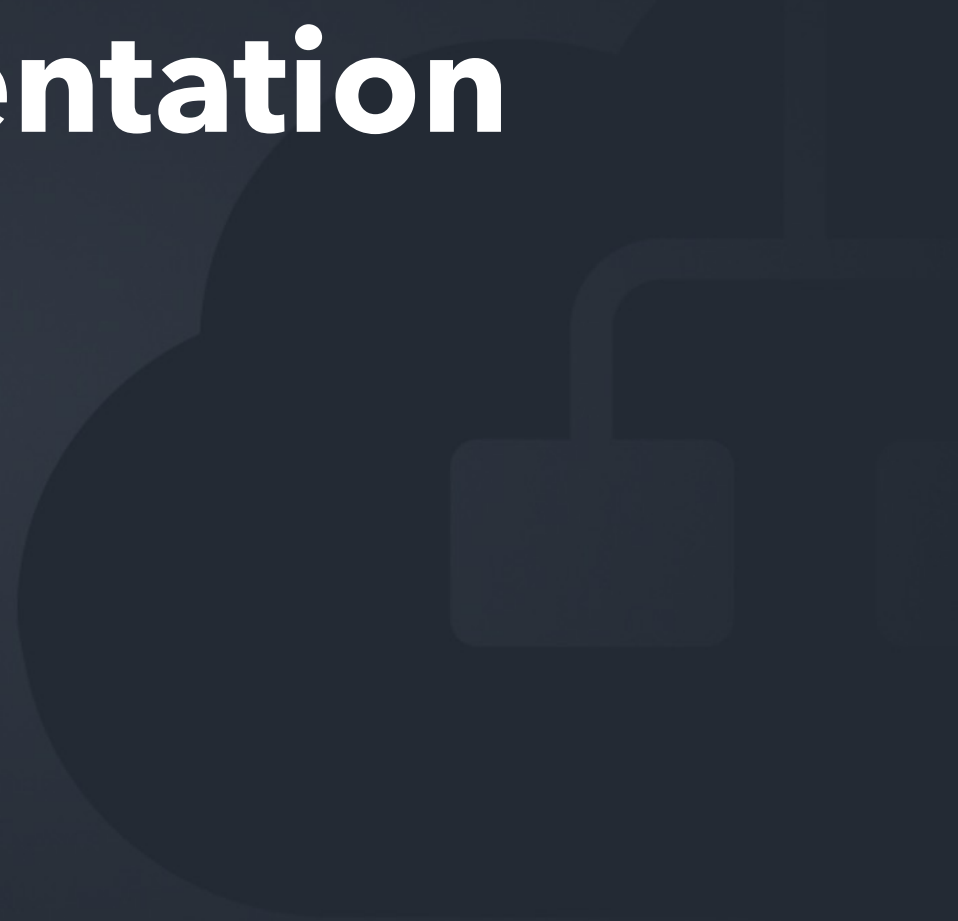


HTTP/2 403

"message": "Resource not accessible by personal access token"

```
https://api.github.com/repos/unskript/Awesc
{
  "count": 891,
  "uniques": 57,
  "views": [
    {
      "timestamp": "2023-01-06T00:00:00Z",
      "count": 2,
      "uniques": 1
    },
    {
      "timestamp": "2023-01-07T00:00:00Z",
      "count": 7,
```

Documentation



Documentation

RunBooks
Checklists

Documentation

RunBooks
Checklists

**Application Support and
Operations takes up to
55% of Developer time**

Documentation Advantages

<https://zwischenzugs.com/2017/04/04/things-i-learned-managing-site-reliability-for-some-of-the-worlds-busiest-gambling-sites/>

Documentation Advantages

- Faster resolution of Issues

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding

<https://zwischenzugs.com/2017/04/04/things-i-learned-managing-site-reliability-for-some-of-the-worlds-busiest-gambling-sites/>

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding
- Better training

<https://zwischenzugs.com/2017/04/04/things-i-learned-managing-site-reliability-for-some-of-the-worlds-busiest-gambling-sites/>

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding
- Better training
- Better Discipline

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding
- Better training
- Better Discipline
- Automation

<https://zwischenzugs.com/2017/04/04/things-i-learned-managing-site-reliability-for-some-of-the-worlds-busiest-gambling-sites/>

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding
- Better training
- Better Discipline
- Automation

7 month kick-off
10% ongoing

Documentation Getting Started

<https://blog.danslimmon.com/2019/07/15/do-nothing-scripting-the-key-to-gradual-automation/>

Documentation Getting Started

Do Nothing Scripting

<https://blog.danslimmon.com/2019/07/15/do-nothing-scripting-the-key-to-gradual-automation/>

Documentation Getting Started

Do Nothing Scripting

- Create steps in the code

Documentation Getting Started

Do Nothing Scripting

- Create steps in the code
- Add Automations as needed

Documentation Getting Started

Do Nothing Scripting

- Create steps in the code
- Add Automations as needed

Make it Accessible to the team



UNSKRIPT, INC.

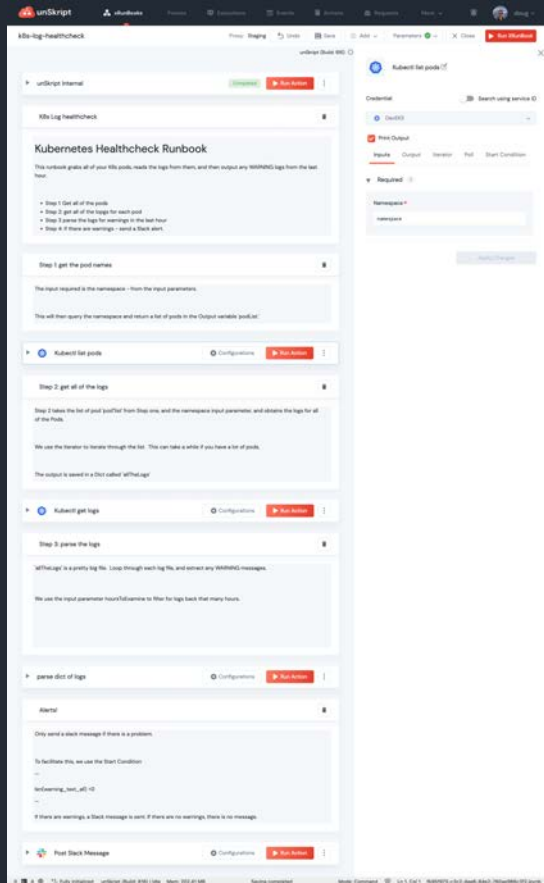


Open Source RunBook Automation

<https://github.com/unskript/Awesome-CloudOps-Automation>

Open Source RunBook Automation

Based on Jupyter Notebooks

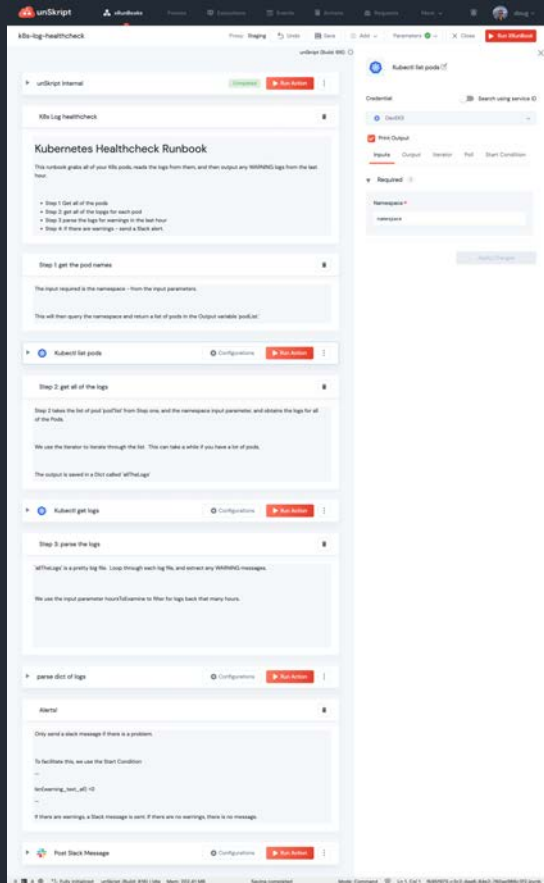


<https://github.com/unskript/Awesome-CloudOps-Automation>

Open Source RunBook Automation

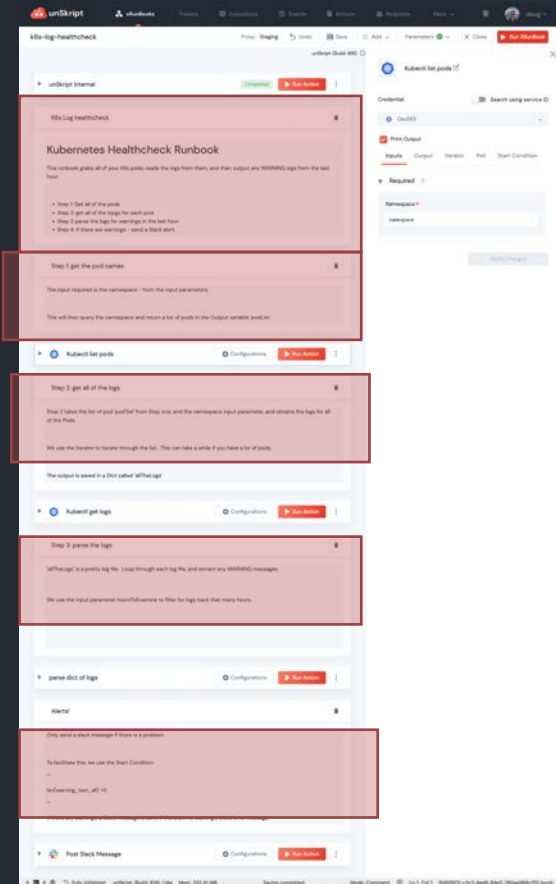
Based on Jupyter Notebooks

- Online/easily shared amongst teams



<https://github.com/unskript/Awesome-CloudOps-Automation>

Open Source RunBook Automation

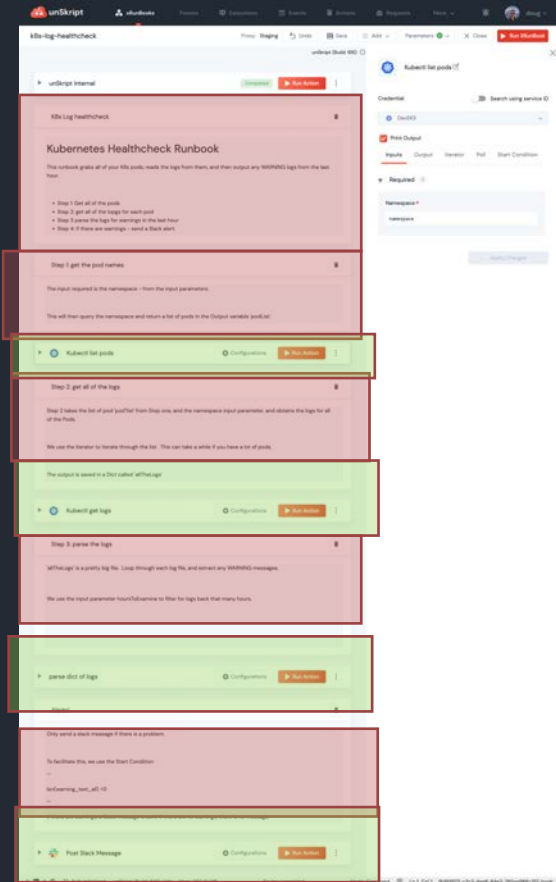


Based on Jupyter Notebooks

- Online/easily shared amongst teams
- Text/Markdown fields

<https://github.com/unskript/Awesome-CloudOps-Automation>

Open Source RunBook Automation



Based on Jupyter Notebooks

- Online/easily shared amongst teams
- Text/Markdown fields
- Automation fields (Python)

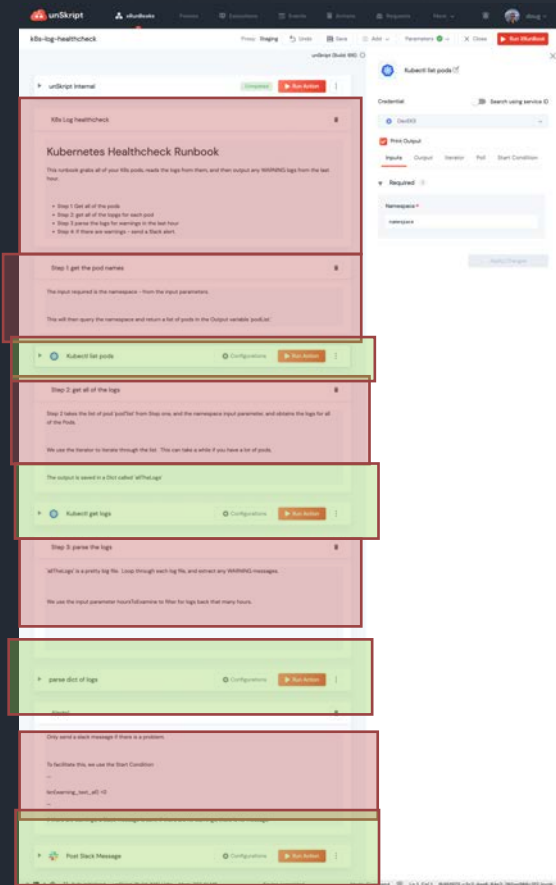
<https://github.com/unskript/Awesome-CloudOps-Automation>

Documentation Advantages

- Faster resolution of Issues
- Simpler Escalations
- Easier onboarding
- Better training
- Better Discipline
- Automation

7 month kick-off
10% ongoing

Open Source RunBook Automation



Based on Jupyter Notebooks

- Online/easily shared amongst teams
- Text/Markdown fields
- Automation fields (Python)
- Hundreds of pre-built Actions
 - AWS, GCP, K8s, DBs, Jira & more!

Example RunBook

Kubernetes Healthcheck Runbook

▶  Kubectl list pods ⚙️ Configurations ▶ Run Action ⋮

▶  Kubectl get logs ⚙️ Configurations ▶ Run Action ⋮

▶ parse dict of logs ⚙️ Configurations ▶ Run Action ⋮

▶  Post Slack Message ⚙️ Configurations ▶ Run Action ⋮

Example RunBook

Kubernetes Healthcheck Runbook

▶  Kubectl list pods  Configurations  Run Action 

▶  Kubectl get logs  Configurations  Run Action 


▶ parse dict of logs  Configurations  Run Action 

▶  Post Slack Message  Configurations  Run Action 



Example RunBook

Kubernetes Healthcheck Runbook

▶  Kubectl list pods

 Configurations

 Run Action



▶  Kubectl get logs

 Configurations

 Run Action



▶ parse dict of logs


 Configurations

 Run Action



▶  Post Slack Message

 Configurations

 Run Action



Example RunBook

Kubernetes Healthcheck Runbook

▶  Kubectl list pods ⚙️ Configurations ▶ Run Action ⋮

▶  Kubectl get logs ⚙️ Configurations ▶ Run Action ⋮

▶ parse dict of logs ⚙️ Configurations ▶ Run Action ⋮

▶  Post Slack Message ⚙️ Configurations ▶ Run Action ⋮



**RunBook
available in
Open Source**

Actions



Actions

Actions Library

Choose action from our library or create your own.

my Actions 24

unSkript Actions 342

Actions

If the desired Action doesn't exist

- **Extend existing Action**

Actions

If the desired Action doesn't exist

- **Extend existing Action**
- **Create a new Action**

Actions

If the desired Action doesn't exist

- **Extend existing Action**
- **Create a new Action**
 - **Connect to External Connection**

Actions

If the desired Action doesn't exist

- **Extend existing Action**
- **Create a new Action**
 - **Connect to External Connection**
 - **"Glue" Action**

Create a new Action



Create a new Action

- **Tag an EC2 Instance**
- **Check all GCP VMs and tell me if they are public or not**

Create a new Action



ChatGPT




Conversational AI

Can produce poetry, Text,
Code

<https://chat.openai.com/chat>

Tag an AWS EC2 instance

ChatGPT

 Examples	 Capabilities	 Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

can you write a python script to add a "CostCenter" tag with the value "Marketing" to an EC2 instance

ChatGPT-Jan 9 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag to
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[{'Key': tag_key, 'Value':
tag_value}])
```

Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag to
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[{'Key': tag_key, 'Value': tag_value}])
```

```
@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region:str) -> Dict:

    ec2 = handle.client('ec2')
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)
```


Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag to
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[{'Key': tag_key, 'Value': tag_value}])
```

```
@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region:str) -> Dict:

    ec2 = handle.client('ec2')
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

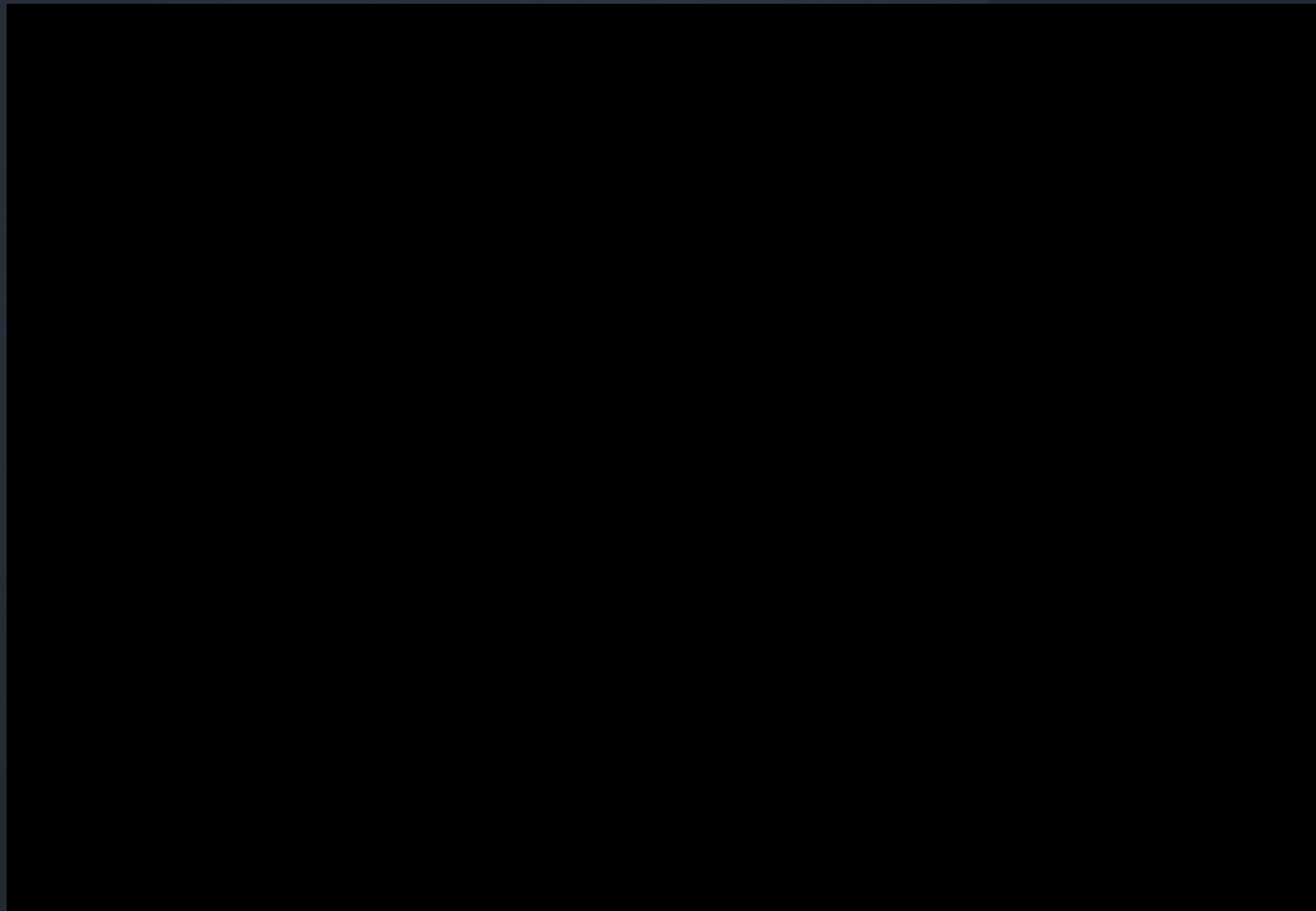
    return(res)
```

Error while executing: You must specify a region.

Execution Failed: Error: You must specify a region.

Task Parameters: {'instance': 'i-07e9ba22ffdef647c', 'tag_key': 'CostCenter'}

Tag an *AWS* EC2 instance



Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[{'Key': tag_key, 'Value': tag_value}])
```

Copy code

```
@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region: str) -> Dict:

    ec2 = handle.client('ec2', region_name=region)
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)
```

Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[tag_key, tag_value])
```

Copy code

```
@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region: str) -> Dict:

    ec2 = handle.client('ec2', region_name=region)
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)
```

```
{'ResponseMetadata': {'HTTPHeaders': {'cache-control': 'no-cache, no-store',
    'content-length': '221',
    'content-type': 'text/xml;charset=UTF-8',
    'date': 'Mon, 23 Jan 2023 17:22:25 GMT',
    'server': 'AmazonEC2',
    'strict-transport-security': 'max-age=31536000;
    includeSubDomains',
    'x-amzn-requestid': 'f6f12bc2-d87c-44b9-b970-f87f2c74ef43'},
    'HTTPStatusCode': 200,
    'RequestId': 'f6f12bc2-d87c-44b9-b970-f87f2c74ef43',
```

Tag an AWS EC2 instance

```
import boto3

# Connect to the EC2 service
ec2 = boto3.client('ec2')

# The ID of the EC2 instance to add the tag
instance_id = 'i-1234567890abcdef0'

# The key and value for the tag
tag_key = 'CostCenter'
tag_value = 'Marketing'

# Add the tag to the instance
ec2.create_tags(Resources=[instance_id], Tags=[{'Key': tag_key, 'Value': tag_value}])
```

Copy code

```
@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region: str) -> Dict:

    ec2 = handle_client('ec2', region_name=region)
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)
```

Tags

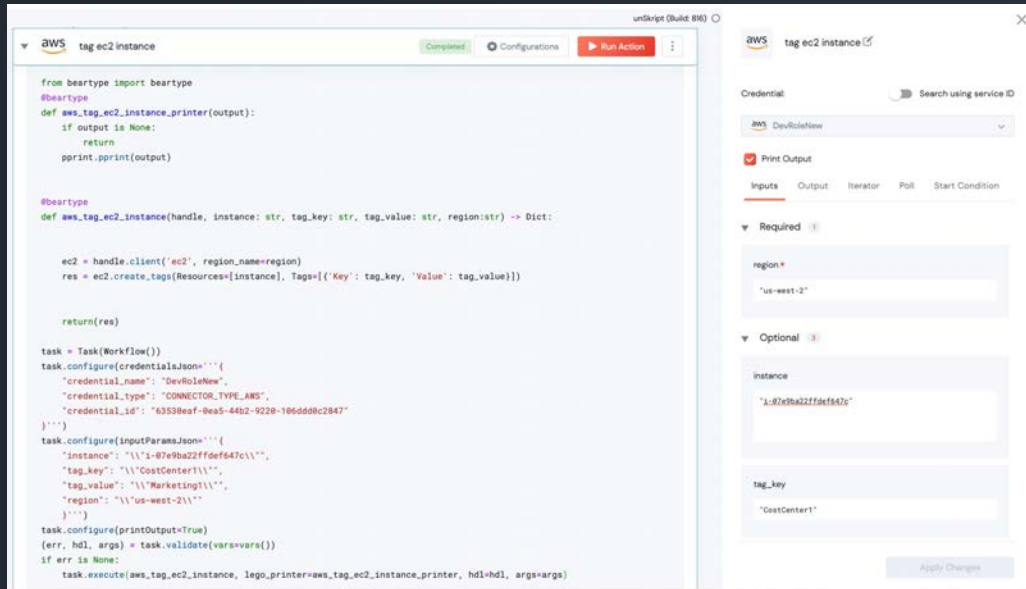
Manage tags



< 1 >

Key	Value
CostCenter1	Marketing1
CostCenter	Marketing
Name	doug_test

Tag an AWS EC2 instance



The screenshot displays the AWS CloudFormation console for a task definition named "tag ec2 instance". The left pane shows the task definition code, and the right pane shows the configuration options.

```
from beartype import beartype
@beartype
def aws_tag_ec2_instance_printer(output):
    if output is None:
        return
    pprint.pprint(output)

@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region: str) -> Dict:

    ec2 = handle.client('ec2', region_name=region)
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)

task = Task(Workflow())
task.configure(credentials_json="""
{
  "credential_name": "DevRoleNew",
  "credential_type": "CONNECTOR_TYPE_AWS",
  "credential_id": "63538eaf-8e85-44b2-9228-106d99c2847"
}
""")
task.configure(input_params_json="""
{
  "instance": "\\i-07e9ba22ffdef647c\\",
  "tag_key": "\\CostCenter1\\",
  "tag_value": "\\Marketing1\\",
  "region": "\\us-west-2\\"
}
""")
task.configure(print_output=True)
(err, hdl, args) = task.validate(vars=vars())
if err is None:
    task.execute(aws_tag_ec2_instance, lego_printer=aws_tag_ec2_instance_printer, hdl=hdl, args=args)
```

The configuration options on the right include:

- Credential:** AWS DevRoleNew
- Print Output:** Checked
- Required:** region* (us-west-2)
- Optional:** instance (i-07e9ba22ffdef647c), tag_key (CostCenter1)



Tag an AWS EC2 instance

```
from beartype import beartype
@beartype
def aws_tag_ec2_instance_printer(output):
    if output is None:
        return
    pprint.pprint(output)

@beartype
def aws_tag_ec2_instance(handle, instance: str, tag_key: str, tag_value: str, region: str) -> dict:
    ec2 = handle.client('ec2', region_name=region)
    res = ec2.create_tags(Resources=[instance], Tags=[{'Key': tag_key, 'Value': tag_value}])

    return(res)

task = Task(Workflow())
task.configure(credentialsJson'''{
    "credential_name": "DevRoleNew",
    "credential_type": "CONNECTOR_TYPE_AWS",
    "credential_id": "65538eaf-8e45-44b2-9228-106d99c2847"
}''')
task.configure(inputParamsJson'''{
    "instance": "\\i-07e9ba22ffdef647c\\",
    "tag_key": "\\CostCenter1\\",
    "tag_value": "\\Marketing1\\",
    "region": "\\us-west-2\\",
}''')
task.configure(printOutput=True)
(err, hd1, args) = task.validate(vars=vars())
if err is None:
    task.execute(aws_tag_ec2_instance, lego_printer=aws_tag_ec2_instance_printer, hd1=hd1, args=args)
```

Credential: Search using service ID

aws DevRoleNew

Print Output

Inputs Output Iterator Poll Start Condition

Required 1

region *

"us-west-2"

Optional 3

instance

"i-07e9ba22ffdef647c"

tag_key

"CostCenter1"

Apply Changes

Required 1

region *

"us-west-2"

Optional 3

instance

"i-07e9ba22ffdef647c"

tag_key

"CostCenter1"

tag_value

"Marketing1"

Credential:



Search using service ID

aws DevRoleNew



Print Output

Get all GCP VMs

And tell me if they are publicly available



Get all GCP VMs

And tell me if they are publicly available



Can you please use the google cloud project library to write python code that creates a list of all GCP VMs and whether are not they are publicly available for project 'unskript-dev' in region 'us-west1-a'?



Get all GCP VMs

And tell me if they are publicly available

The image shows a screenshot of the ChatGPT web interface. On the left is a dark sidebar with white text links: "Script Python", "Instances", "Script for Checking C...", "Parabola About X-A...", "Revolution Calcula...", "py Birthday Song", "mentation Recipe", "versations", "e", "scored", and "FAQ". The main content area has a dark background with white text. At the top center is the "ChatGPT" logo. Below it are three columns: "Examples" (with a sun icon), "Capabilities" (with a lightning bolt icon), and "Limitations" (with a warning triangle icon). Each column contains three rows of text. At the bottom is a text input field with a question and a send button.

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

Can you please use the google cloud project library to write python code that creates a list of all GCP VMs and whether are not they are publicly available for project 'unskript-dev' in region 'us-west1-a'?

ChatGPT_Jan.9.Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Get all GCP VMs

And tell me if they are publicly available

```
from google.cloud import compute_engine

client = compute_engine.Client()
project = 'unskript-dev'
region = 'us-west1-a'

vms = client.list_instances(project=project, zone=region)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

print(vm_list)
```

```
compute_client = compute.InstancesClient(credentials=handle)

vms = compute_client.list(project=project, zone=zone)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

return(vm_list)
```

Get all GCP VMs

And tell me if they are publicly available

```
from google.cloud import compute_engine

client = compute_engine.Client()
project = 'unskript-dev'
region = 'us-west1-a'

vms = client.list_instances(project=project, zone=region)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

print(vm_list)
```

```
compute_client = compute.InstancesClient(credentials=handle)

vms = compute_client.list(project=project, zone=zone)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

return(vm_list)
```

Get all GCP VMs

And tell me if they are publicly available

```
from google.cloud import compute_engine

client = compute_engine.Client()
project = 'unskript-dev'
region = 'us-west1-a'

vms = client.list_instances(project=project, zone=region)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

print(vm_list)
```

```
compute_client = compute.InstancesClient(credentials=handle)

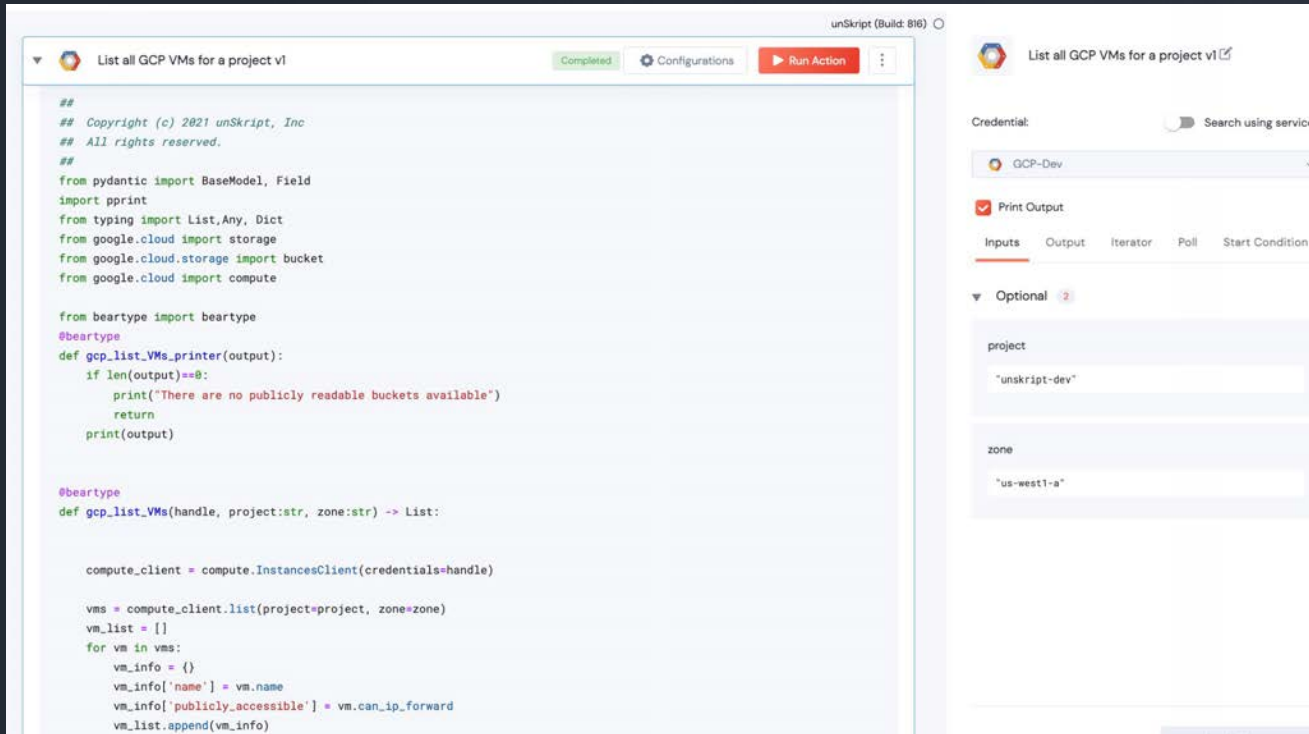
vms = compute_client.list(project=project, zone=zone)
vm_list = []
for vm in vms:
    vm_info = {}
    vm_info['name'] = vm.name
    vm_info['publicly_accessible'] = vm.can_ip_forward
    vm_list.append(vm_info)

return(vm_list)
```

```
{'name': 'gke-trace-e2-standard-2-saas-<clip>',
'publicly_accessible': False},
{'name': 'gke-trace-e2-standard-2-saas-<clip>',
'publicly_accessible': False},
{'name': 'gke-trace-github-runner-<clip>',
'publicly_accessible': False}}
```

Get all GCP VMs

And tell me if they are publicly available



The image shows a screenshot of the unSkript interface. The left pane displays a Python script for listing GCP VMs. The right pane shows the configuration for the script, including a credential dropdown set to 'GCP-Dev', a checked 'Print Output' option, and input fields for 'project' (set to 'unskript-dev') and 'zone' (set to 'us-west1-a').

```
##  
## Copyright (c) 2021 unSkript, Inc  
## All rights reserved.  
##  
from pydantic import BaseModel, Field  
import pprint  
from typing import List, Any, Dict  
from google.cloud import storage  
from google.cloud.storage import bucket  
from google.cloud import compute  
  
from beartype import beartype  
@beartype  
def gcp_list_VMs_printer(output):  
    if len(output)==0:  
        print("There are no publicly readable buckets available")  
        return  
    print(output)  
  
@beartype  
def gcp_list_VMs(handle, project:str, zone:str) -> List:  
  
    compute_client = compute.InstancesClient(credentials=handle)  
  
    vms = compute_client.list(project=project, zone=zone)  
    vm_list = []  
    for vm in vms:  
        vm_info = {}  
        vm_info['name'] = vm.name  
        vm_info['publicly_accessible'] = vm.can_ip_forward  
        vm_list.append(vm_info)
```

unSkript (Build: 816)

List all GCP VMs for a project v1 Completed Configurations Run Action

List all GCP VMs for a project v1 [↗](#)

Credential: Search using service I

GCP-Dev

Print Output

Inputs Output Iterator Poll Start Condition

Optional 2

project

unskript-dev

zone

us-west1-a

Get all GCP VMs

And tell me if they are publicly available

```
unSkript (Build: 816)
List all GCP VMs for a project v1 [Completed] [Configurations] [Run Action]

##
## Copyright (c) 2021 unSkript, Inc
## All rights reserved.
##
from pydantic import BaseModel, Field
import pprint
from typing import List, Any, Dict
from google.cloud import storage
from google.cloud.storage import bucket
from google.cloud import compute

from beartype import beartype
@beartype
def gcp_list_VMs_printer(output):
    if len(output)==0:
        print("There are no publicly readable buckets available")
        return
    print(output)

@beartype
def gcp_list_VMs(handle, project:str, zone:str) -> List:

    compute_client = compute.InstancesClient(credentials=handle)

    vms = compute_client.list(project=project, zone=zone)
    vm_list = []
    for vm in vms:
        vm_info = {}
        vm_info['name'] = vm.name
        vm_info['publicly_accessible'] = vm.can_ip_forward
        vm_list.append(vm_info)
```

unSkript (Build: 816) List all GCP VMs for a project v1

Credential: Search using service ID

GCP-Dev

Print Output

Inputs Output Iterator Poll Start Condition

Optional 2

project

"unskript-dev"

zone

"us-west1-a"

Credential: Search using service ID

GCP-Dev

project

"unskript-dev"

zone

"us-west1-a"

Demo

- **Live demo**
- **Any suggestions to create a Automation Action using Chat GPT?**

Summary

- **RunBooks: Internal Documentation**
 - **Improve outcomes**
 - **Lower MTTR**
 - **Improve team collaboration**
 - **Can be Automated**



Summary

- **RunBooks: Internal Documentation**
 - **Improve outcomes**
 - **Lower MTTR**
 - **Improve team collaboration**
 - **Can be Automated**
- **Automation**
 - **Reduces Manual DevOps 'toil'**
 - **Auto remediations**
 - **Increase observability**
 - **Etc.**



Summary

- **RunBooks: Internal Documentation**
 - **Improve outcomes**
 - **Lower MTTR**
 - **Improve team collaboration**
 - **Can be Automated**
- **Automation**
 - **Reduces Manual DevOps 'toil'**
 - **Auto remediations**
 - **Increase observability**
 - **Etc.**
- **unSkript**
 - **Open Source**
 - **Jupyter Notebooks**
 - **Hundreds of automations built in**

Summary

- **RunBooks: Internal Documentation**
 - **Improve outcomes**
 - **Lower MTTR**
 - **Improve team collaboration**
 - **Can be Automated**
- **Automation**
 - **Reduces Manual DevOps 'toil'**
 - **Auto remediations**
 - **Increase observability**
 - **Etc.**
- **unSkript**
 - **Open Source**
 - **Jupyter Notebooks**
 - **Hundreds of automations built in**
- **ChatGPT**
 - **Prototyping of automation -> almost working demo**

Resources

- **unSkript**

- <https://runbooks.sh> 



★ Starred 111

- <https://unskript.com>

- **ChatGPT**

- <https://chat.openai.com/chat>

- **Things I Learned Managing Site Reliability for Some of the World's Busiest Gambling Sites**

- <https://zwischenzugs.com/2017/04/04/things-i-learned-managing-site-reliability-for-some-of-the-worlds-busiest-gambling-sites/>

- **Do Nothing Scripting**

- <https://blog.danslimmon.com/2019/07/15/do-nothing-scripting-the-key-to-gradual-automation/>

Thank You

