Cloud Custodian @ Scale 17x
Your public cloud swiss army knife

Kapil Thangavelu @kapilvt
Cloud Custodian

Custodian is an open source rules engine for account and resource management on AWS, Azure, and GCP.

YAML DSL for policies based on querying resources, filtering them and taking actions.

Intended to scale from mom and pop shops to large enterprise needs.
Cloud Custodian

Policies target a particular resource type, like AWS Elastic Compute Cloud (EC2)

Filter resources, invoke actions on matched.

Vocabularies of actions, and filters for policy construction.

- name: ebs-unused
  resource: ebs
  mode:
    type: config-rule
  filters:
    - Attachments: empty
    - "tag:Retain": absent
  actions:
    - type: mark-for-op
      days: 5
Installation

$ pip install c7n

$ docker pull cloudcustodian/c7n

$ custodian run -m aws --log-group -s output_dir
# Rich Filtering

JMESPath Expressions

Value Type (size, cidr_size, age, expiration, etc)

Operators (>=, <=, list in/not-in, regex, etc)

Arbitrary nesting with and, or blocks

Values from url/s3.

```python
# This filter is to take out EMR instances
- type: event
  key: "detail.userIdentity.invokedBy.serviceName"
  value: "elasticmapreduce.amazonaws.com"
  op: not-equal

- type: value
  # Ignore keys that start with 'aws:' as they don't count towards the limit.
  key: "[length(Tags[?!starts_with(Key,'aws:')]])[0]"
  op: lt
  value: 10

- type: value
  key: "tag:AccountId"
  op: not-in
  values_from:
    expr: "accounts.*.accountNumber"
    url: s3://mybucket/mykey
```
Custodian outputs

Per policy execution outputs.

- AWS X-Ray - Policy api call tracing
- Amazon CloudWatch Metrics - (Resource Counts, API calls, Execution time)
- Amazon CloudWatch Logs - Policy execution logs
- Amazon S3 - JSON resource, metadata, logs archive
- GCP StackDriver Logging & Metrics
- Azure Application Insights & Blob Store

Streaming notifications via Amazon Simple Queue Service and Amazon Simple Notification Service, with delivery channels to Amazon SES, Slack, and SMTP.
Multi Step Workflows

“Poorly tagged instances, should be stopped in 1 day, and then terminated in 3”

- Action: mark-for-op
- Filter: marked-for-op

Chain together multiple policies.

- name: ec2-tag-compliance-mark
  resource: ec2
  description: |
  Find all non-compliant tag instances for stoppage in 1 days.
  mode:
    type: config-rule
  filters:
    - "tag:maid_status": absent
    - or:
      - "tag:App": absent
      - "tag:Env": absent
      - "tag:Owner": absent
  actions:
    - type: mark-for-op
      op: stop
      days: 1

- name: ec2-tag-compliance-stop
  resource: ec2
  description: |
  Stop poorly tagged and schedule Terminate.
  mode:
    type: periodic
  schedule: rate(1 day)
  filters:
    - type: marked-for-op
      op: stop
    - or:
      - "tag:App": absent
      - "tag:Env": absent
      - "tag:Owner": absent
  actions:
    - stop
    - type: mark-for-op
      op: terminate
      days: 2
Custodian Capabilities

Imagination is the limit
- Encryption
- Backups
- Garbage Collection
- Unused resize
- Offhours
- Tag compliance
- SG compliance
- Key rotation
- Perimeter Check

aws:
resources: 152
actions: 107
filters: 119

azure:
resources: 31
actions: 15
filters: 22

gcp:
resources: 24
actions: 9
filters: 7
Custodian Architecture

Stateless, Multiple Execution modes, Integrated Provisioning.

Execution Modes
- Poll (Default)
- AWS Events (Cloud Trail, Health, GuardDuty, etc)
- AWS Config Rule
- GCP Audit Log
- Azure ARM Events
- Azure/AWS/GCP Periodic

Custodian will automatically provision event sources and serverless resources.
Compliance as Code

Policy as Code - Authoring

- Versioning (Rollbacks & Audit)
- Peer Review
- Abstraction and encapsulation
- Code Reuse
- Automation

Policy as Code - Visualize & Index

PolicyStream for event stream of policy changes from git history.

Policy as Code - Delivery

Continuous Integration
- Drone or Jenkins
  - custodian validate mypolicy.yml
  - custodian run --dryrun mypolicy.yml
- LGTM or Github Reviews
  - Multi-person signoff

Continuous Delivery
- git pull & custodian run policy files.
Safety Belts

Three liner to nuke all your instances

```python
import boto3, jmespath
client = boto3.client('ec2')
client.terminate_instances(
    Instancelds=jmespath.search(
        'Reservations.Instances[].InstanceId',
        client.describe_instances() )
)
```

Custodian has built-in safety belts, that can be set on any policy

- max-resources
- max-resources-percent
AWS CloudTrail - Inspection

- **name**: require-rds-auto-tag
  - **resource**: rds
  - **mode**:
    - **type**: cloudtrail
    - **events**:
      - CreateDBInstance
  - **filters**:
    - **or**:
      - Encrypted: false
      - PubliclyAvailable: true
  - **actions**:
    - **type**: delete
      - skip-snapshot: true

Powered by Amazon CloudWatch Events

- Powerful infrastructure observation capabilities
- Enables “realtime” rules enforcement and reaction with wide coverage of AWS product APIs.
- Dozen of event sources.
AWS Config - Rule Support

name: lambda-check
resource: lambda
mode:
  type: config-rule
filters:
  - or:
    - tag:Owner: absent
    - type: cross-account
Amazon Guard Duty - Remediation Support

name: ec2-guard-remediate
resource: ec2
mode:
  type: guard-duty
member-role: arn:aws:iam::{account_id}:role/Duty
filters:
  # Filter for medium and high severity events
  - type: event
    key: detail.severity
    op: gte
    value: 4.5
actions:
  # remove any instance profile on the instance
  - type: set-instance-profile
    profile: null
  # stop the instance
  - stop
  # snapshot the disk for forensics
  - type: snapshot
    copy_tags: [Name]

We also distribute tools like c7n_guardian that automate multi-account, organization enrollment.

name: audit-guard-duty
description: “Alert if guard duty not enabled”
resource: account
mode:
  type: periodic
  schedule: “rate(1 day)”
filters:
  # type: guard-duty
  Detector.Status: ENABLED
  Master.AccountId: "00011001"
  Master.RelationshipStatus: ENABLED
GCP - Cloud Functions

Python Runtime (Beta)

Server Side Dependency Building...

- name: serverless-ftw
  resource: gcp.instance
  mode:
    type: gcp-audit
    methods:
      - "v1.compute.instances.start"
  filters:
    - "tag:Quarantined": present
  actions:
    - stop

$ custodian run gcp.yml -s out
2018-07-31 10:20:03: c7n_gcp.mu:INFO function code uploaded
2018-07-31 10:20:04: c7n_gcp.mu:INFO updating function config
GCP - Auditing API Calls
Azure - Auditing API Calls

Azure Function Python Runtime
Consumption or App Service Plan
Event Grid subscribing to resource group events.

name: tag-key-vault-creator
resource: azure.keyvault
mode:
  type: azure-event-grid
events:
  - resourceProvider: Microsoft.KeyVault/vaults
    event: write
filters:
  - "tag:CreatorEmail": null
actions:
  - type: auto-tag-user
    tag: CreatorEmail
## Custodian Multi Cloud

<table>
<thead>
<tr>
<th>Feature</th>
<th>AWS</th>
<th>GCP</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serverless</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Api Subscriber</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Storage</td>
<td>✔️</td>
<td>Q2</td>
<td>✔️</td>
</tr>
<tr>
<td>Logging</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Metrics</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Resource Query</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Multi Account</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Status</td>
<td>Stable</td>
<td>Alpha</td>
<td>Beta</td>
</tr>
</tbody>
</table>
c7n_org - Multi Account Execution

Run and report custodian policies across multiple accounts.

Bonus run arbitrary scripts across accounts.

```bash
$ c7n-org run-script -s ~/waf-deploy \
  -c accounts/custodian.yml -r us-east-1 \
  -tag "partition:us" \n  -tag "type:dev" \n  aws cloudformation deploy \n    --stack-name default-global-waf \n    --template-file=deploy/waf-default.yml
```

$ c7n-org
Usage: c7n-org [OPTIONS] COMMAND [ARGS]...

custodian organization multi-account runner.

Options:
   --help  Show this message and exit.

Commands:
   report report on a cross account policy execution.
   run    run a custodian policy across accounts
   run-script run an aws script across accounts
c7n_mailer - Notification Delivery

Deliver policy notifications to users.

Jinja2 Message Templates

Supports LDAP Address Lookup

Supported delivery transports

- Slack
- SES
- SMTP
- SendGrid

```bash
c7n-mailer --help
usage: c7n-mailer [-h] [-c CONFIG [-d | --debug]]
               [-t TEMPLATES] [--update-lambda | --run]
optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                         mailer.yml config file
  --debug               sets c7n_mailer logger to debug,
                         for maximum output
  --max-num-processes MAX_NUM_PROCESSES
                         will run the mailer in parallel,
                         integer of max
                         processes allowed
  -t TEMPLATES, --templates TEMPLATES
                         message templates folder location
  --update-lambda       packages your c7n_mailer, uploads
                         the zip to aws
  --run                 run c7n-mailer locally, process sqs
                         messages and send
```
Auto Tag Creators

Who made X, Y, and Z

Just auto tag it.

```yaml
policies:
  - name: auto-tag-buckets
    mode:
      type: cloud-trail
      events: [CreateBucket]
    actions:
      - type: auto-tag-owner
        tag: "ResourceCreator"
```
Cost Savings - Garbage Collection

Find underutilized or unused things and resize or remove them.

Very useful across resources like:

- Amazon Elastic Compute Cloud (EC2)
- Amazon Elastic Block Store (EBS)
- Amazon Elastic Load Balancing (ELB)
- Amazon CloudWatch Logs
- Amazon Redshift
- Amazon Relational Database Service

```yaml
- name: rds-unused-mark
  resource: rds
  description: |
    Take the average connections over 14 days and mark unused RDS instances.
  filters:
    - "tag:custodian_cleanup": absent
    - type: value
      value_type: age
      key: InstanceCreateTime
      value: 14
      op: gt
    - type: metrics
      name: DatabaseConnections
      statistic: Sum
      days: 14
      value: 0
      op: equal
  actions:
    - type: mark-for-op
      tag: custodian_cleanup
      op: delete
      days: 14
```
Cost Savings - Lights Out

Let’s turn off all asg in our dev vpc at night and on weekends and save 64% on instance costs.

Also works for Amazon RDS / DocumentDB / Aurora / Neptune

Docs

```yaml
policies:
  - name: offhours-stop
    resource: ec2
    filters:
      - VpcId: xyz
      - type: offhour
        weekends: false
default_tz: pt
tag: downtime
opt-out: true
onhour: 8
offhour: 20
```
Aws Security Primer
Ensuring Audit Trail - Amazon Cloud Trail

Cloud Trail - Control Plane API calls for services.

Great tools for working with cloud trail logs.

- Amazon Macie, ElasticSearch (ELK), Splunk, Custodian TrailDb, Zalando FullStop, Amazon Athena

```yaml
policies:
- name: aws-cloudtrail-not-enabled
  resource: account
  region: us-east-1
  tags:
    - level: high
  description: |
    Scans for accounts which do not have CloudTrails enabled.

filters:
- type: check-cloudtrail
  global-events: true
  multi-region: true
  running: true
  kms: true
  file-digest: true
```
Ensuring Audit Trail - Amazon Config

Config provides for periodic polling and archiving of current and historical revisions of resources (that it supports).

```yaml
policies:
- name: aws-config-not-enabled
  resource: account
  region: us-east-1
  tags:
    - level: high
  description: |
    Policy scans for accounts which do not have CloudTrails enabled.

filters:
- type: check-config
  all-resources: true
  global-resources: true
  running: true
```
Ensuring Audit Trail - Amazon VPC Flow Logs

Flow logs give basic metadata on connections between ips within a vpc.

```json
policies:
  - name: flow-log-not-enabled
    resource: vpc
    filters:
      - type: flow-log
        state: absent
    actions:
      - <<: *notify_action
        subject: "Missing VPC Flow Logs"
```
AWS Identity and Access Management (AWS)

MFA

- name: user-missing-mfa
  resource: iam-user
  region: us-east-1
  filters:
    - type: credential
      key: password_enabled
    - type: mfa-device
      value: empty
  actions:
    - <<: *notify_action
      subject: "Enable MFA on user"

Password Policy

- name: iam-password-policy
  resource: account
  region: us-east-1
  filters:
    - or:
      - type: password-policy
        key: MaxPasswordAge
        value: 30
        op: gte
      - type: password-policy
        key: MinimumPasswordLength
        value: 8
        op: lte
name: ec2-overprivileged
resource: ec2
mode:
  type: config-rule
filters:
  - type: check-permission
    actions:
      - iam:CreateUser

name: unused-access-keys
resource: iam-user
region: us-east-1
filters:
  - type: credential
    key: access_keys.last_used_date
    value_type: age
    value: 120
    op: greater-than
actions:
  - type: post-finding
    confidence: 100
    compliance_status: FAILED
    recommendation: Rotate Key
    types: [
IAM (Continued)

Detect Root Logins

- name: root-login-detected
  resource: account
  description: |
    Notifies on root user logins
  mode:
    type: cloudtrail
    role: arn:aws:iam::{account_id}:role/Eg
    events:
    - ConsoleLogin
  filters:
    - type: event
      key: "detail.userIdentity.type"
      value: Root
  actions:
  - <<: *notify_action
    subject: "AWS Root Console Login Alert"

Detect Lack of Root Hardware MFA

- name: root-virtual-mfa
  resource: account
  description: |
    Notifies on root user without hw mfa
  filters:
  - type: iam-summary
    key: AccountMFAEnabled
    value: true
  - type: has-virtual-mfa
    value: true
  actions:
  - <<: *notify_action
    subject: "AWS Root Missing MFA"
Resources w/ Embedded IAM

- SNS
- SQS
- Glacier
- Lambda & Layers
- ECR
- KMS
- S3*
- API Gw
- Iam Role (Trust)
- Snapshots & Amis

ECR

vars:
  accounts_url: &aurl s3://mybucket/myaccounts.json

policies:
  - name: ecr-cross-account
    resource: ecr
    filters:
      - type: cross-account
        whitelist_from:
          expr: "accounts.*.accountNumber"
          url: *accounts_url
    actions:
      - <<: *notify_action
        subject: "[custodian] ECR available externally"
      - type: remove-statements
        statement_ids: matched
Ensuring Audit Trail - ALB/ELB/Cloudfront Logs

Capture access logs from all albs

```json
policies:
  - name: appelb-logs
    resource: appelb
    filters:
      - type: value
        key: "Attributes.access_logs.s3.enabled"
        value: False
    actions:
      - type: set-s3-logging
        bucket: my-lb-logs
        prefix: "Logs/{AccountId}/{LoadBalancerName}"
Network Security - AWS Shield

CloudFront

policies:
- name: activate-cloudfront-shield
  resource: distribution
  region: us-east-1
  filters:
    - type: shield-enabled
      state: false
  actions:
    - type: set-shield
      state: true

ELB

policies:
- name: activate-elb-shield
  resource: elb
  filters:
    - type: value
      key: Scheme
      value: internal
      op: not-equal
  actions:
    - type: set-shield
      state: true
Network Security - AWS WAF

CloudFront

policies:
- name: activate-cloudfront-waf
  resource: distribution
  region: us-east-1
  filters:
    - Status: Deployed
    - WebACLId: empty
  actions:
    - type: set-waf
      web-acl: global-default
      state: true

Application Load Balancer

policies:
- name: activate-alb-waf
  resource: app-elb
  filters:
    - type: value
      key: Scheme
      value: internal
      op: not-equal
    - type: waf-enabled
      state: false
  actions:
    - type: set-waf
      web-acl: regional-default
      state: true
Encryption at REST

- SSE Protects against very specific threat models
- Provides ancillary protection around accidental snapshot sharing (image, redshift, ebs, rds).

```yaml
policies:
- name: asg-unencrypted
  resource: asg
  filters:
  - type: not-encrypted
    exclude_images: true

- name: kinesis-unencrypted
  resource: kinesis
  filters:
  - KeyId: absent

- name: ec2-encrypted
  resource: ec2
  filters:
  - type: ebs
    key: Encrypted
    value: false
    skip-devices: ["dev/xvda", "/dev/sda1"]
```
Server Management

With Amazon SSM Agent

You can manage servers with custodian.

See tools/omnissm in the repo.

```
name: ec2-osquery-install
resource: ec2
filters:
  - type: ssm
    key: PingStatus
    value: Online
  - type: ssm
    key: PlatformName
    value: Ubuntu
  - type: ssm
    key: PlatformVersion
    value: 18.04
actions:
  - type: send-command
    command:
      DocumentName: AWS-RunShellScript
      Parameters:
      commands:
        - wget https://pkg.osquery.io/deb/osquery
        - dpkg -i osquery_3.3.0_1.linux.amd64.deb
```
Roadmap

Pycon 2019 (Ohio) Development Sprint
- Documentation
- Kubernetes Provider

Stronger AWS IAM Management Capabilities

GCP parity

Policy Author Experience Improvements

Project Stats

1m+ downloads
170+ contributors
2100+ pull requests
1500+ tests ~ 90% cov
~700 users in chatroom
Community

Home Page [http://cloudcustodian.io](http://cloudcustodian.io)

Gitter [https://gitter.im/capitalone/cloud-custodian](https://gitter.im/capitalone/cloud-custodian)

Github [https://github.com/cloudcustodian/cloud-custodian](https://github.com/cloudcustodian/cloud-custodian)

Reddit [https://reddit.com/r/cloudcustodian](https://reddit.com/r/cloudcustodian)

Mailing List - [https://groups.google.com/forum/#!forum/cloud-custodian](https://groups.google.com/forum/#!forum/cloud-custodian)
Any Questions