**CHAOS TOOLKIT**

**Introduction**:

Chaos Engineering is a discipline that allows you to surface weaknesses, and eventually build confidence, in complex and often distributed systems.

The Chaos Toolkit aims to give you the simplest experience for writing and running your own Chaos Engineering experiments.

The key concepts of the Chaos Toolkit are Experiments, Steady State Hypothesis and the experiment’s Method. The Method contains a combination of Probes and Actions.

## Experiments

A Chaos Toolkit experiment is provided in a single file and is currently expressed in JSON.

## Steady State Hypothesis

A Steady State Hypothesis describes “what normal looks like” for your system in order for the experiment to surface information about weaknesses when compared against the declared “normal” tolerances of what is measured.

The Chaos Toolkit uses the Steady State Hypothesis for two purposes. It is used as a check before an experiment is run that the target system is in a recognised **normal** state. It is also used as the template for comparison of the state of your system **after** the experiment has been run, forming the results provided by the experiment’s report.

## Method

An experiment’s activities are contained within its Method block.

## Probes

A probe is a way of observing a particular set of conditions in the system that is undergoing experimentation.

## Actions

An action is a particular activity that needs to be enacted on the system under experimentation.

## Rollbacks

An experiment may define a sequence of actions that revert what was undone during the experiment.

## Controls

An experiment may declare a set of controls which have an impact over the execution of the experiment itself. Controls are operational elements rather than experimental.

**Requirements:**

**Python Requirements**

The [chaostoolkit CLI](https://github.com/chaostoolkit/chaostoolkit) is implemented in Python 3 and this requires a working Python installation to run. It officially supports Python 3.7+.

**Python Installation:**

Install Python for your system:

* On MacOS X:

brew install python3

* On Debian/Ubuntu:

sudo apt-get install python3 python3-venv

* On Windows:

[Download the latest binary installer](https://www.python.org/downloads/windows/) from the Python website.

**Virtual Environment**

Create a virtual environment[¶](https://chaostoolkit.org/reference/usage/install/#create-a-virtual-environment)

Chaos toolkit can be installed directly on your system via the pip package manager.

But it is advisable to install in a local virtual environment.

python3 -m venv ~/.venvs/chaostk

Make sure to always activate your virtual environment before using it:

source ~/.venvs/chaostk/bin/activate

**Install Chaos Toolkit CLI**

Install chaostoolkit in the virtual environment as follows:

pip install -U chaostoolkit

You can verify the command was installed by running:

chaos --version

Install Extensions

At this stage, you have installed the chaos command line and its core library. To fully enjoy the Chaos Toolkit, you will also want to install [extensions](https://github.com/search?utf8=%E2%9C%93&q=topic%3Achaostoolkit-extension&type=Repositories).

**Run Chaos commands**

Once installed, you can see the options available to you by executing:

* chaos init –help
* chaos run -- help

A tutorial on how to use the chaos init command is available as part of the [Chaos Toolkit’s Getting Started tutorials.](https://www.katacoda.com/chaostoolkit/courses/01-chaostoolkit-getting-started)

**Initialise a new experiment**

To initialise a new experiment based on what has been [discovered](https://chaostoolkit.org/reference/usage/discover/) you simply need to execute the chaos init command:

* chaos init

By default, the chaos init command will look for a ./discovery.json file and use that as the basis of a new experiment’s initialisation.

**Executing an Experiment Plan**

To execute an experiment plan you simply pass it to the chaos run command:

chaos run experiment.json

[2018-01-30 16:35:04 INFO] Validating experiment's syntax

[2018-01-30 16:35:04 INFO] Experiment looks valid

[2018-01-30 16:35:04 INFO] Running experiment: My new experiment

[2018-01-30 16:35:04 INFO] No steady state hypothesis defined. That's ok, just exploring.

[2018-01-30 16:35:04 INFO] Action: kill\_microservice

[2018-01-30 16:35:04 INFO] No steady state hypothesis defined. That's ok, just exploring.

[2018-01-30 16:35:04 INFO] Let's rollback...

[2018-01-30 16:35:04 INFO] No declared rollbacks, let's move on.

[2018-01-30 16:35:04 INFO] Experiment ended with status: completed

Chaos Toolkit will log all the steps it follows from your plan in a journal which by default is called journal.json. You can specify a different the name of this journal output file using the --journal-path option.

Rehearsing an experiment execution

To test that you have a valid experiment you can pass the --dry option.

Run an experiment without validation

You can run an experiment and skip the experiment’s validation using the --no-validation option.

**AWS Extension – chaosaws**

Repo - <https://github.com/chaostoolkit-incubator/chaostoolkit-aws>

This project is a collection of [actions](http://chaostoolkit.org/reference/api/experiment/#action) and [probes](http://chaostoolkit.org/reference/api/experiment/#probe), gathered as an extension to the [Chaos Toolkit](http://chaostoolkit.org/).

Installation

This package requires Python 3.6+

The package must be installed in the Python environment where [chaostoolkit](http://chaostoolkit.org/) is already installed. You can install it by using the pip command:

* pip install -U chaostoolkit-aws

A sample usage with AWS resources:

{

"name": "stop-an-ec2-instance",

"provider": {

"type": "python",

"module": "chaosaws.ec2.actions",

"func": "stop\_instance",

"arguments": {

"instance\_id": "i-123456"

}

}

},

{

"name": "create-a-new-policy",

"provider": {

"type": "python",

"module": "chaosaws.iam.actions",

"func": "create\_policy",

"arguments": {

"name": "mypolicy",

"path": "user/Jane",

"policy": {

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:ListAllMyBuckets",

"s3:GetBucketLocation"

],

"Resource": "arn:aws:s3:::\*"

}

]

}

}

}

}

## We are going to carry out 4 experiments on the following resources:

## Experiment 1 - IAM

## Create a new policy

## json file:

{

"title": "create-a-policy",

"description": "chaos toolkit is used to create an IAM policy",

"tags": [],

"steady-state-hypothesis": {

"title": "IAM policies in approved state",

"probes": []

},

"configuration": {

"aws\_region": "eu-west-2",

"aws\_profile\_name": "bravo"

},

"method": [

{

"name": "create-a-new-policy",

"type": "probe",

"provider": {

"type": "python",

"module": "chaosaws.iam.actions",

"func": "create\_policy",

"arguments": {

"name": "chaos-policy-for-collins",

"path": "/fume/cupboard/",

"policy": {

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:ListAllMyBuckets",

"s3:GetBucketLocation"

],

"Resource": "arn:aws:s3:::\*"

}

]

}

}

}

} ]

}

collinsorighose@Collinss-MacBook-Pro-2 iam % chaos run create-policy.json

[2023-08-31 09:42:26 INFO] Validating the experiment's syntax

[2023-08-31 09:42:26 INFO] Experiment looks valid

[2023-08-31 09:42:26 INFO] Running experiment: create-a-policy

[2023-08-31 09:42:26 INFO] Steady-state strategy: default

[2023-08-31 09:42:26 INFO] Rollbacks strategy: default

[2023-08-31 09:42:26 INFO] No steady state hypothesis defined. That's ok, just exploring.

[2023-08-31 09:42:26 INFO] Playing your experiment's method now...

[2023-08-31 09:42:26 INFO] Probe: create-a-new-policy

[2023-08-31 09:42:27 INFO] Let's rollback...

[2023-08-31 09:42:27 INFO] No declared rollbacks, let's move on.

[2023-08-31 09:42:27 INFO] Experiment ended with status: completed

collinsorighose@Collinss-MacBook-Pro-2 iam %

**Attach a policy to an existing role:**

json file:

{

"title": "attach-a-policy",

"description": "chaos toolkit is used to attach an IAM policy",

"tags": [],

"steady-state-hypothesis": {

"title": "IAM policies in approved state",

"probes": []

},

"configuration": {

"aws\_region": "eu-west-2",

"aws\_profile\_name": "bravo"

},

"method": [

{

"name": "attach-role-policy",

"type": "probe",

"provider": {

"type": "python",

"module": "chaosaws.iam.actions",

"func": "attach\_role\_policy",

"arguments": {

"arn": "arn:aws:iam::684361860346:policy/fume/cupboard/chaos-policy-for-collins",

"role\_name": "chas-ec2-role"

}

}

} ]

## }

collins-MacBook-Pro-2 iam % chaos run attach-policy.json

[2023-08-31 09:46:34 INFO] Validating the experiment's syntax

[2023-08-31 09:46:35 INFO] Experiment looks valid

[2023-08-31 09:46:35 INFO] Running experiment: attach-a-policy

[2023-08-31 09:46:35 INFO] Steady-state strategy: default

[2023-08-31 09:46:35 INFO] Rollbacks strategy: default

[2023-08-31 09:46:35 INFO] No steady state hypothesis defined. That's ok, just exploring.

[2023-08-31 09:46:35 INFO] Playing your experiment's method now...

[2023-08-31 09:46:35 INFO] Probe: attach-role-policy

[2023-08-31 09:46:38 INFO] Let's rollback...

[2023-08-31 09:46:38 INFO] No declared rollbacks, let's move on.

[2023-08-31 09:46:38 INFO] Experiment ended with status: completed

## collins-MacBook-Pro-2 iam %

## Experiment 2 - S3

## Experiment 3 - SSM

## Experiment 4 - CloudWatch

## How to Investigate Issues

When your experiment fails to work as you would expect, you should start looking at the chaostoolkit.log file written to by the chaos command.

This file contains a lot of traces from the Chaos Toolkit core but also any extensions that used the toolkit’s logger.

As new logs are appended to that file, it may grow big. Do not hesitate to wipe it out from time to time.

Please, do make sure to visit our [Slack](https://join.slack.com/t/chaostoolkit/shared_invite/zt-22c5isqi9-3YjYzucVTNFFVIG~Kzns8g) or [GitHub](https://github.com/chaostoolkit) when you have a question around how the toolkit does things. The community will be pleased to help you out.

[Generating a report](https://chaostoolkit.org/reference/usage/report/#generating-a-report)

The chaos report command[¶](https://chaostoolkit.org/reference/usage/report/#the-chaos-report-command)

You use the chaos report command to take the journal produced by the [chaos run](https://chaostoolkit.org/reference/usage/run/) command and produce a report in a specified format.

Due to the many operating system-dependent features that the chaos report command relies upon, the chaos report command is not installed with the Chaos Toolkit CLI. To install the chaos report command you need to [install the chaostoolkit-reporting plugin and the dependencies appropriate to your own operating system](https://github.com/chaostoolkit/chaostoolkit-reporting).

Once the plugin is installed you can see the options available to you by executing:

chaos report --help

Usage: chaos report [OPTIONS] [JOURNAL]... REPORT

Generate a report from the run journal(s).

Options:

--export-format TEXT Format to export the report to: html, markdown, pdf.

--help Show this message and exit.

A tutorial on how to use the chaos report command is available as part of the [Chaos Toolkit’s Getting Started tutorials.](https://www.katacoda.com/chaostoolkit/courses/01-chaostoolkit-getting-started)

Generating a report[¶](https://chaostoolkit.org/reference/usage/report/#generating-a-report)

When an experiment completes after using the chaos run command a journal is generated and stored in the chaos-report.json file. A PDF or HTML report may be generated from this journal using the [chaostoolkit-reporting](https://github.com/chaostoolkit/chaostoolkit-reporting) library.

The chaos report command expects the path to the chaos-report.json file and a path to the actual report file that you require.

You can export various formats of report by specifying what you want using the --export-format option.

For example, to generate a PDF report you can run the following command:

chaos report --export-format=pdf chaos-report.json report.pdf

An HTML report can be produced using:

chaos report --export-format=html5 chaos-report.json report.html