loadsv2 Documentation

Release 0.1

Ben Bangert, Kit Cambridge, Tarek Ziade

Mar 29, 2017

Contents

loads v2 is an AWS-orchestrating load-generation tool, a.k.a load-tester.

Core features:

- Any language can be used to write a load-generator
- Metric collection (Statsd, logfiles) built-in
- Grafana metric dashboards for each run
- Ability to set up the service being load-tested
- Load-generation strategies can combine load-generators
- RESTful API for triggering load-tests

CHAPTER 1

Narrative Documentation

To start learning about `loads`, how to set it up, how to write load- generators, and how to run load-testing strategies – start here.

About Loads

Welcome to `loads` v2, a load-testing tool that strives to provide a powerful and flexible load-testing environment for websites, web services, web applications, and network daemons.

Background

`loads` v2 was created to address a few short-comings in available load- testing tools, including the original loads tool.

Some missing features in existing solutions:

- Inability to specify how many machines to use per load-test
- Load-test client/script required using a specific language
- Unable to effectively test alternate protocols (websockets)
- Inability to spawn the service being load-tested for each load-test

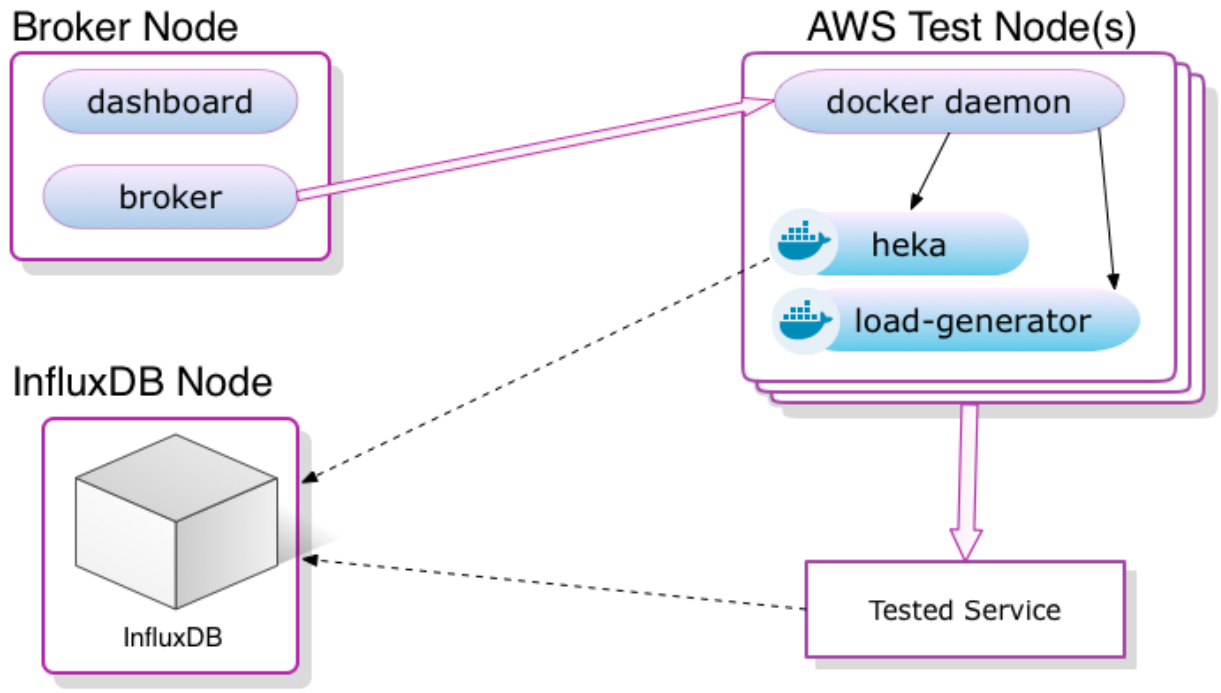
To address these short-comings, `loads` doesn't define how/what a load- tester should be written in or what it can do. Load-testers can be any program packaged in a docker container that can be run solely via environment/command-line arguments.

By focusing solely on orchestrating docker containers across AWS, load-tests become very dynamic, capable of running multiple programs at once.

Since `loads` v2 can orchestrate the running of dockerized programs, if the service to load-test is also dockerized then it can be deployed and run prior to a load-test as well. This makes it easier to tweak configuration parameters to find more robust deployment configurations and see the new load-test results quickly.

Load-Test Architecture

`loads` can be setup in a few different configurations, depending on desired requirements and available equipment. This configuration has InfluxDB on a separate node, but if the node running the loads-broker is sufficient, then it can be run there.



The loads-broker is the orchestration program that coordinates all the running strategies and AWS Test Nodes.

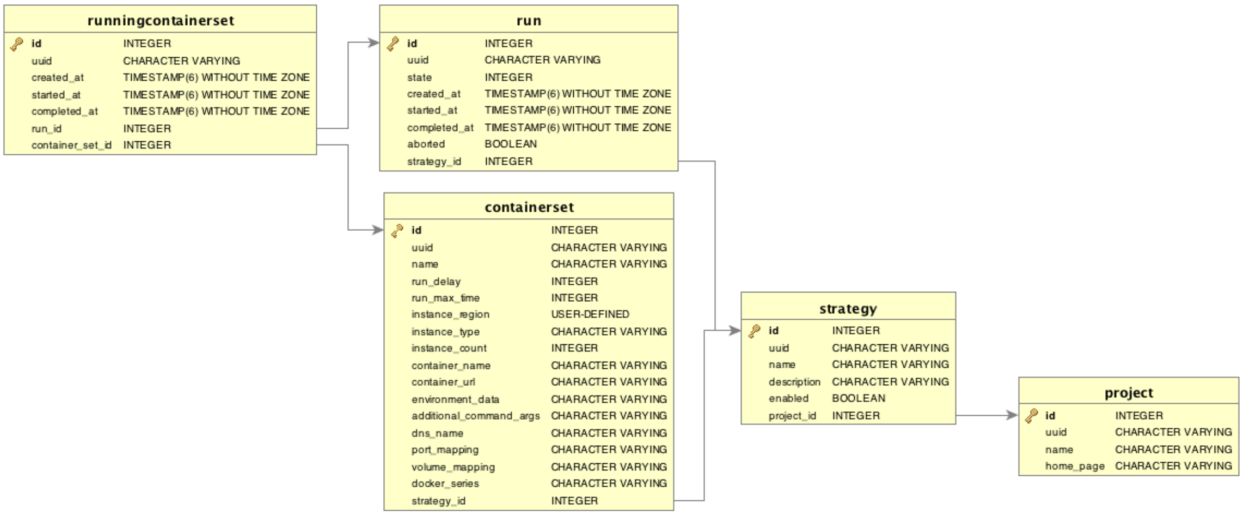
AWS Test Nodes are dynamically created by the loads-broker as needed to fulfill the load-test strategy. Each AWS Test Node has docker installed, which the loads-broker communicates with to have heka and the configured load-generator containers installed.

Each load-tester on each AWS node is supplied with appropriate information to genererate load against the service to test, and sends metrics data to the local heka container which aggregates the data before relaying it to the InfluxDB node.

The service being tested should send metrics data to InfluxDB as well, so that metrics gathered by the load-generators can be easily compared with service metrics.

Database Schema

A high-level database entity-relationship diagram:



The core organization from the top-most down:

Project A project is the top-level organization in loads-broker. Each separate service to test should have a project. Projects may have multiple strategies associated with them.

Plan A load-test plan defines one or more Step's to run, along with when the Step should start/stop.

Step A Step contains all the information needed to allocate AWS instances (instance type/region/count), what docker container to run, how soon after the Plan is started to run it, how long the container should be allowed to run for, and what environment vars and command-line arguments it should receive.

Run Each time a plan is triggered, a Run is created. Runs track when the plan was started/stopped, and its current state. Runs also record the execution of each step as a StepRecord. Each StepRecord records when a step for the plans run was started/stopped.

StepRecord Records a Step for a Run, when it was started/stopped.

Projects, Plans, and Step's need to be created in the database before loads-broker can be run. Run's and StepRecord's are created when a Plan is run by the loads-broker.

Warning:

The Step's for a Plan cannot be changed if the Plan has been run. This is

because a Run reflects a run of the strategy, and the information regarding the run becomes inaccurate if it fails to represent the running of the StepRecord's.

Changing Step configurations for a Plan should be done by forking the Plan

and changing the new one before any Run's are done.

CHAPTER 2

API Documentation

`loads` documentation for developers that wish to work directly with the `loads` code-base and/or create their own custom extensions for load-testing orchestration.

API Documentation

Reference documentation for *loads* code-base:

`loadsbroker.aws`

Core EC2 Classes

Helpers

`loadsbroker.broker`

Core Classes

Utility

loadsbroker.db

Database Tables

Utility

loadsbroker.dockerctrl

Core

Utility

loadsbroker.exceptions

loadsbroker.extensions

Core

loadsbroker.main

loadsbroker.options

loadsbroker.ssh

loadsbroker.util

loadsbroker.webapp.api

API Handlers

Base Class

loadsbroker.webapp.views